TECH ASSIGNMENT

The goal of this assignment is to implementing a toy app to show how the candidate copes with:

- Managing specifications and requirements
- Interacting with JSON REST APIs
- Building possibly long scrolling lists with complex structure
- Managing live data/UI updates
- Code architecture skills

To take into account the time availability, which could be limited based on a current occupation, we split the assignment into two parts and ask to implement mandatorily at least only the first one.

Moreover, a few options will be given in terms of what is optional and appreciated but not required.

The candidate will be evaluated more on HOW he/she will implement the solution rather than on HOW MUCH he/she will do.

The candidate will be asked to:

- present the app developed and show that it matches the requirements
- Identify the more interesting parts of the app and discuss them in terms of the challenges they offered, how they have been coped with and what alternatives there could have been

IMPORTANT: Do not focus on fancy UI, make the app do what it's expected to do.

IMPORTANT: Do not "over do", do what you can in the time you have, do it at your best, you can take note of anything else you would have liked to do and then you can tell us about it, it will be valued positively. It's better to have something less but working fine, than some fancy solution that does not work!

IMPORTANT: We do not expect that the candidate understands the betting domain, and not even all the details of the requirements and specifications, so for any question the candidate shall feel free to ask via email for more details or clarifications.

The assignment

The assignment in its core is quite simple:

- Show a welcome page
- Give access to a menu with a navigation tree
- Show the items related to a node in the navigation tree

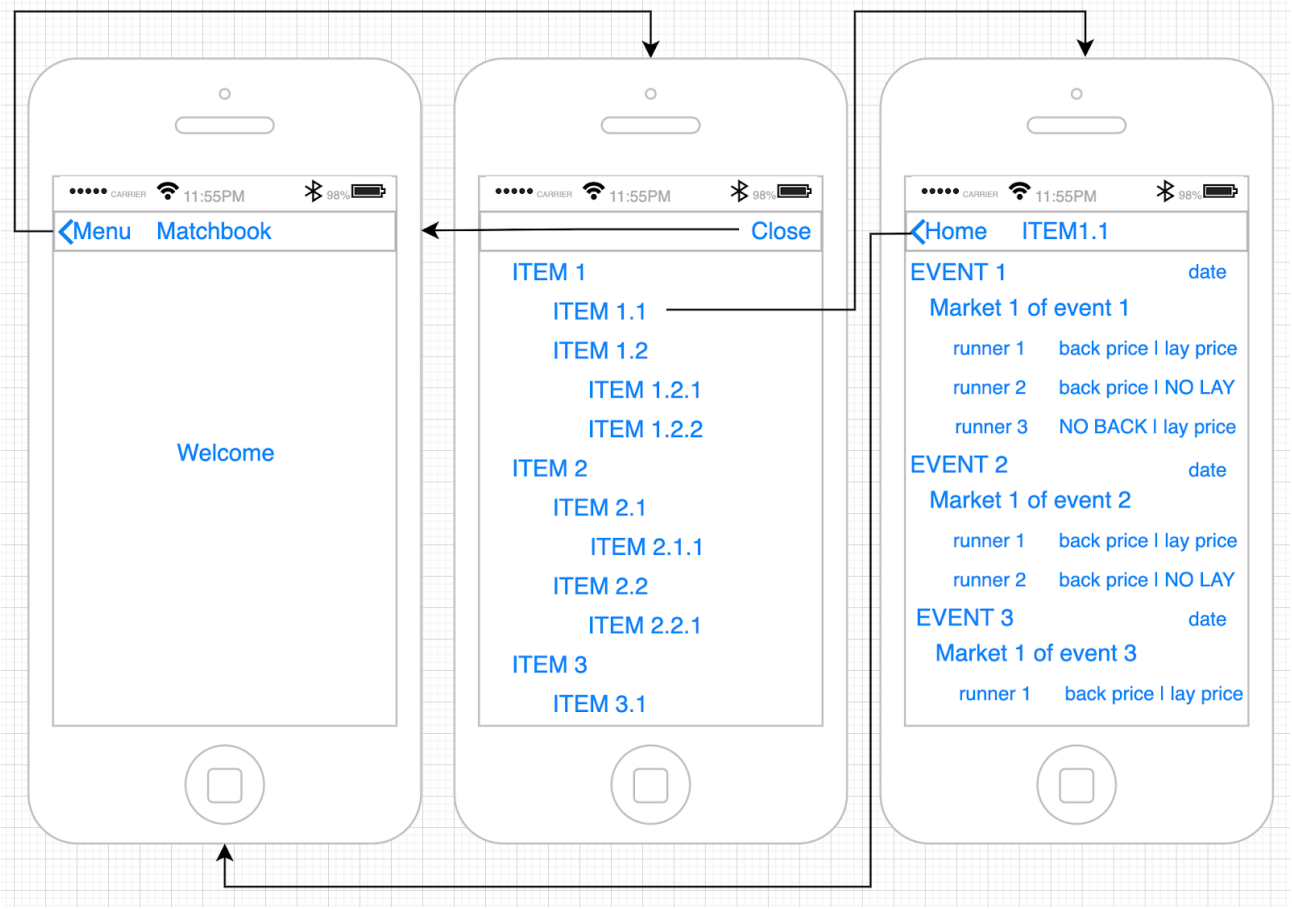
Some context

Do not worry about understanding the domain, it's not needed for this assignment, but some basic terminology can help you in reading the requirements and understanding the APIs:

- By **event** we mean matches between teams or players: Juventus vs Milan, Federer vs Nadal…
- By **market** of an event we mean a possible point of view on an event: which team will win, whether the teams will score in total more than 5 goals…
- By **runner** of a market we mean a specific outcome of the market: Juventus win, the two teams in total scored more than 5 goals…
- By **price** of a runner we mean a "number" that identifies the probability for the runner to win (back price) or lose (lay price), the price determines how much you win if you bet on the right runner
- The **Matchbook navigation tree** is a structure that organizes the events by sport, country, competitions, etc… for example
  SOCCER
  ITALY
  SERIE A
  SERIE B
  Do not worry about the semantics of each level, just take the tree as a data structure.

You are asked to

Implement an app that shows the first three levels of the Matchbook navigation, and allows users to select the leaves of the resulting tree to show the related events.
The following figure illustrates the "pages" of the app, the contents shown and the transitions between pages: WELCOME PAGE, NAVIGATION PAGE, EVENTS PAGE



IMPORTANT:

- We split the assignment into two parts: NAVIGATION, EVENTS
- NAVIGATION: covers only the welcome page and the navigation page
- EVENTS: includes NAVIGATION and adds the events page

Requirements:

NOTE: **don't worry for now about how to fetch the data**, just focus on the interaction with the app, the API specifications follow.

NAVIGATION

1. The app will start showing
    a. a navigation bar with:
        i. "Menu" link
        ii. "Matchbook" title
    b. a page with a "Welcome" text
2. Tapping on the menu link will open a panel (entering from the left) called the navigation drawer

3. The navigation drawer will be full page and will show in the top right a “Close” link to close it and show again the home page of the app
4. The navigation drawer will show the first three levels of the Matchbook navigation tree (Navigation API);
   NOTE: that the Matchbook navigation tree is not guaranteed to have at least 3 levels for each node, so you can have cases like the ones in the wireframe where you have only 2 levels (you could have even just one, so “ITEM 1” for example could have no descendants)

EVENTS

5. Tapping on a leaf (any node with no descendants, like ITEM 1.1) of the navigation tree would
   a. Close the navigation menu
   b. Update the navigation bar to show:
      i. “Home” button to navigate back to the WELCOME page
      ii. The name of the tapped leaf
   c. In the body of the page, show the list of events related to the tapped leaf (Events API)
      Events should sorted by date (most recent first) and for each one the app would show
      i. Name of the event (hyphenate if needed)
      ii. Date of the event (formatted as dd/mm/yy hh/mm)
      iii. Name of the first market (hyphenate if needed)
      iv. List of all runners names for the first market (hyphenate if needed);
      NOTE: the number of runners can vary from market to market
      v. For each runner the app must show the first “back” price (or “NO BACK” if not present) and the first “lay” prive (or *NO LAY” if no present) rounded to two decimals
      NOTE in Events API we explain how a back and a lay price are identified
6. When a list of events is shown, the data must be kept uptodate by polling the same API every 5 seconds and updating the UI accordingly

Implement the app adopting any architecture of choice, but be ready to justify it and possibly discuss alternatives.

OPTIONALS

a. implement the update of the events data using mockup data to simulate changes to the data and be able to show how the UI updates even if the real data are not changing
b. Adopt any form of reactive programming of choice
c. Implement (some) Unit tests
d. Implement (some) UI tests

APIs Specifications

Here we present the APIs with response examples and explain the data they return.

IMPORTANT: in order to be able to connect to Matchbook APIs (and access the Maatchbook website), it is necessary to use the GOOGLE DNS (8.8.8.8).

IMPORTANT: in order to receive the data in JSON format, you need to add to each API request the headers

```
"Accept": "application/json; charset=utf-8"
"Content-type": "application/json; charset=utf-8"
```

Navigation API

- https://www.matchbook.com/edge/rest/navigation?include-tags=true
- Example of a response.
  For the sake of clarity we show a response that contains only two sports and only a few data for each sport, in particular we show only the fields needed for this assignment, in **bold** the fields that you will need to show the navigation tree and to request the events data

  The `name` field is the one you must show in the navigation tree

  The `url-name` field is the one you will use to compose the request for the events data (Events API)

  This is how the complete navigation tree would look based on the example result

  Soccer
  FIFA International Friendlies
  UEFA Euro 2020
  Group A
  Group B
  Australia
  Hyundai A-League
  Tennis
  ATP French Open
  Outrights
  ATP Stuttgart

Response example:

```
[
    {
        "name": "Sport",
        "url-name": "sport",
        "meta-tags": [
            {
                "name": "Soccer",
                "url-name": "soccer",
```

```
        "meta-tags": [
            {
                "name": "FIFA International Friendlies",
                "url-name": "fifa-international-friendlies",
                "meta-tags": []
            },
            {
                "name": "UEFA Euro 2020 ",
                "url-name": "uefa-euro-2020",
                "meta-tags": [
                    {
                        "name": "Group A",
                        "url-name": "group-a",
                        "meta-tags": []
                    },
                    {
                        "name": "Group B",
                        "url-name": "group-b",
                        "meta-tags": []
                    }
                ]
            },
            {
                "name": "Australia",
                "url-name": "australia",
                "meta-tags": [
                    {
                        "name": "Hyundai A-League",
                        "url-name": "hyundai-a-league",
                        "meta-tags": []
                    }
                ]
            }
        ]
    },
    {
        "name": "Tennis",
        "url-name": "tennis",
        "meta-tags": [
            {
                "name": "ATP French Open",
                "url-name": "atp-french-open",
                "meta-tags": [
                    {
                        "name": "Outrights",
                        "url-name": "outrights",
                        "meta-tags": []
                    }
                ]
            },
            {
                "name": "ATP Stuttgart",
                "url-name": "atp-stuttgart",
                "meta-tags": []
```

```
                    }                ]
                }
            ]
        },
        {
            "name": "Country",
            THIS YOU CAN IGNORE, IT'S OF NO USE FOR THIS ASSIGNMENT
        },
        {
            "name": "Date",
            THIS YOU CAN IGNORE, IT'S OF NO USE FOR THIS ASSIGNMENT
        }
]
```

Events API

- https://www.matchbook.com/edge/rest/events?language=en&currency=EUR&exchange-type=back-lay&odds-type=DECIMAL&price-depth=1&per-page=100&market-states=open&runner-states=open&market-auto-sequence=true
- Take the above as a base for your API calls, it contains a few parameters that are NOT of interest for this assignment but are needed to make things smoother.

  Just note the `per-page=100` parameter that you can use to reduce the number of events returned by the API, during development it may be useful to have `per-page=1` or `2` to reduce the noise.

  Suppose you have a navigation tree like

  ITEM 1 (`name="ITEM 1", url-name="url-name-level1"`)
  ITEM 1.1 (`name="ITEM 1.1", url-name="url-name-level2"`)
  ITEM 1.1.1 (`name="ITEM 1.1.1", url-name="url-name-level3"`)

  The leaf is ITEM 1.1.1 and to make the API call to fetch the events after clicking on it you would add to the base URL
  `&tag-url-names=url-name-level1,url-name-level2,url-name-level3`

  If the tree was
  ITEM 1 (`name="ITEM 1", url-name="url-name-level1"`)
  ITEM 1.1 (`name="ITEM 1.1", url-name="url-name-level2"`)

  The leaf would be ITEM 1.1 and you'd add
  `&tag-url-names=url-name-level1,url-name-level2`

- Example of a response, for the sake of clarity we show a response that contains only two events and only a few data for each event, in particular we are NOT showing fields that are not needed for this assignment, in **bold** the fields that you will need to show in the list of events

The `name` fields in events, markets and runners are self explanatory.
The `start` field in the event is self explanatory.
Remember you have to show at most ONE (the first one) marker for each event.

The `odds` field in the prices is what you must show for each runner.
Each runner contains an array of prices (possibly empty), given the base API URL you will get AT MOST one black price and one lay price, and you must show them. If a price is missing you show NO BACK or NO LAY, so you could even have both NO BACK and NO LAY if the runner has no prices.


```
{
    "events": [
        {
            "name": "Turkey vs Italy",
            "start": "2021-06-11T19:00:00.000Z",
            "markets": [
                {
                    "name": "Match Odds",
                    "runners": [
                        {
                            "withdrawn": false,
                            "prices": [
                                {
                                    "odds": 8.80000,
                                    "side": "back",
                                },
                                {
                                    "odds": 9.60000,
                                    "side": "lay",
                                }
                            ],
                            "name": "Turkey",
                        },
                        {
                            "withdrawn": false,
                            "prices": [
                                {
                                    "odds": 1.54054,
                                    "side": "back",
                                },
                                {
                                    "odds": 1.57143,
                                    "side": "lay",
                                }
                            ],
                            "name": "Italy",
                        },
                        {
                            "withdrawn": false,
                            "prices": [
                                {
                                    "odds": 3.96000,
```

```
                            "side": "back",
                        },
                        {
                            "odds": 4.15000,
                            "side": "lay",
                        }
                    ],
                    "name": "DRAW (Tur/Ita)",
                }
            ]
        },
        {
            "name": "Handicap",
            WE ARE NOT INTERESTED IN THIS MARKET BECAUSE IT IS THE
            SECOND OF THE ENVENT, AND WE CONSIDER ONLY THE FIRST ONE
        }
    ]
},
{
    "name": "UEFA Euro 2020 - Winner",
    "start": "2021-06-11T19:00:00.000Z",
    "markets": [
        {
            "name": "Winner",
            "start": "2021-06-11T19:00:00.000Z",
            "runners": [
                {
                    "withdrawn": false,
                    "prices": [
                        {
                            "odds": 5.80000,
                            "side": "back",
                        },
                        {
                            "odds": 6.20000,
                            "side": "lay",
                        },
                    ],
                    "name": "France",
                },
                {
                    "withdrawn": false,
                    "prices": [
                        {
                            "odds": 6.80000,
                            "side": "back",
                        },
                        {
                            "odds": 7.50000,
                            "side": "lay",
                        }
                    ],
                    "name": "England",
                },
```

```
                        {
                            "withdrawn": false,
                            "prices": [
                                {
                                    "odds": 7.10000,
                                    "side": "back",
                                },
                                {
                                    "odds": 8.60000,
                                    "side": "lay",
                                },
                            ],
                            "name": "Belgium",
                        }
                    ]
                }
            ]
        }
    ]
}
```