# Beer Catalog App “SRS”

## 1 Source of information

### 1.1 Punk API

The main source of information is [Punk API](https://punkapi.com/documentation/v2). We will need the following feeds:

1. Get beers with pagination
2. Get single beer

It’s free RESTful API which provides JSON data.

There is a [punkapi-lib](https://www.npmjs.com/package/punkapi-lib) in NPM. Using this is prohibited as you’ll need to build your own proxy layer.

### 1.2 Twitter API

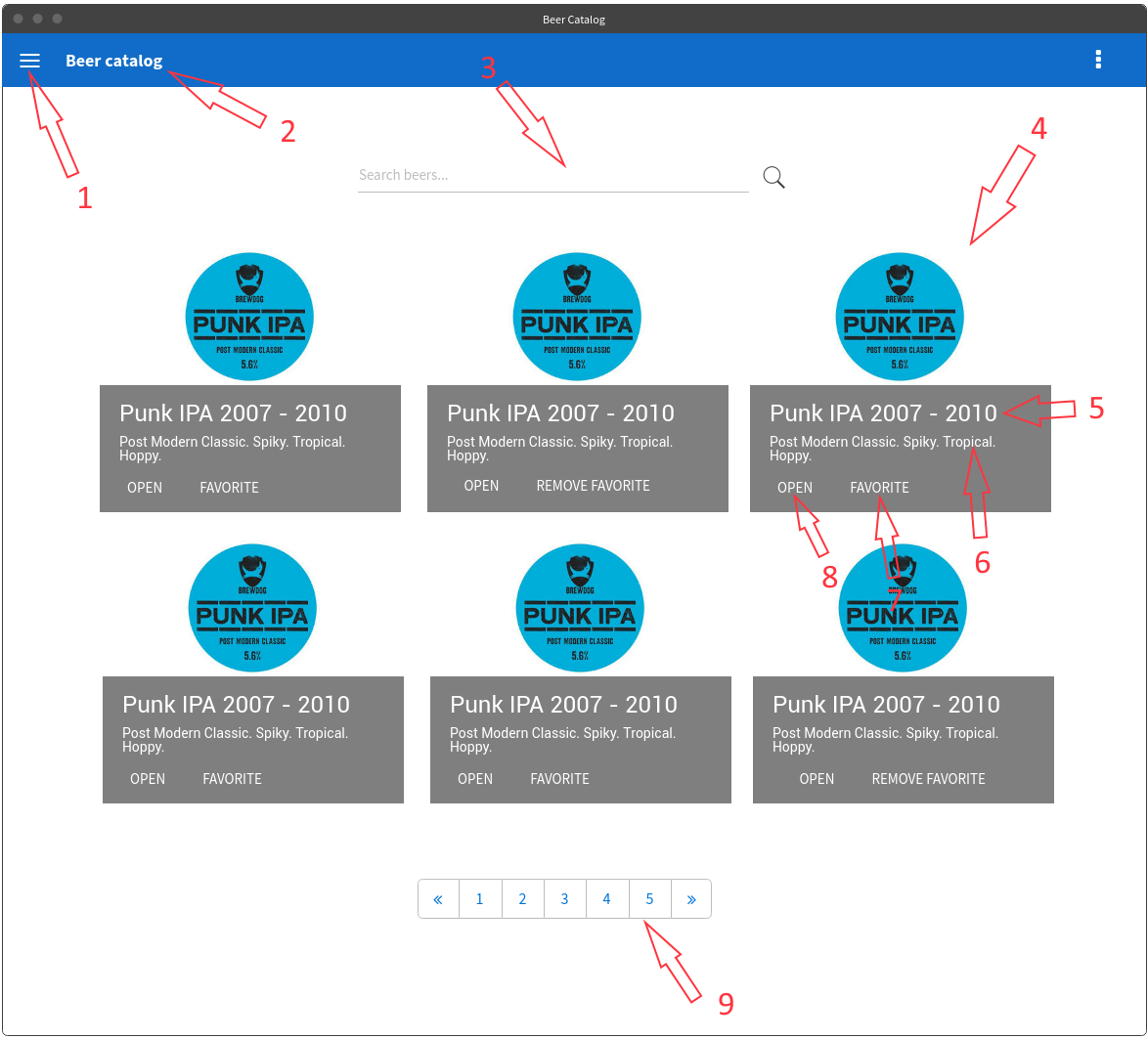
[Twitter API](https://dev.twitter.com/rest/public) will be used for retrieving tweets by club name. So the Search API will be used. To authenticate in this API you’ll need to use your own Twitter account. If you don’t have one, you’ll need to create one.

Twitter has [embedded timelines functionality](https://dev.twitter.com/web/embedded-timelines/search). Using this functionality is **strongly prohibited**. You’ll need to fetch API data and render it in some way.

## 2 Pages

### 2.1 Landing page

Its mockup is presented on pic. 1:



Pic. 1. Landing page mockup

*Figure 1* is Menu button. It opens a drawer similar to [this one](http://material-components-web.appspot.com/drawer/temporary-drawer.html). The drawer contains links to two pages: [2.1 Landing page](#_2.1_Landing_page) and [2.3 Favorites page](#_2.3_Favorites_page).

*Figure 2* is page header. It’s present on every page. Styling may vary, as the picture is a mockup.

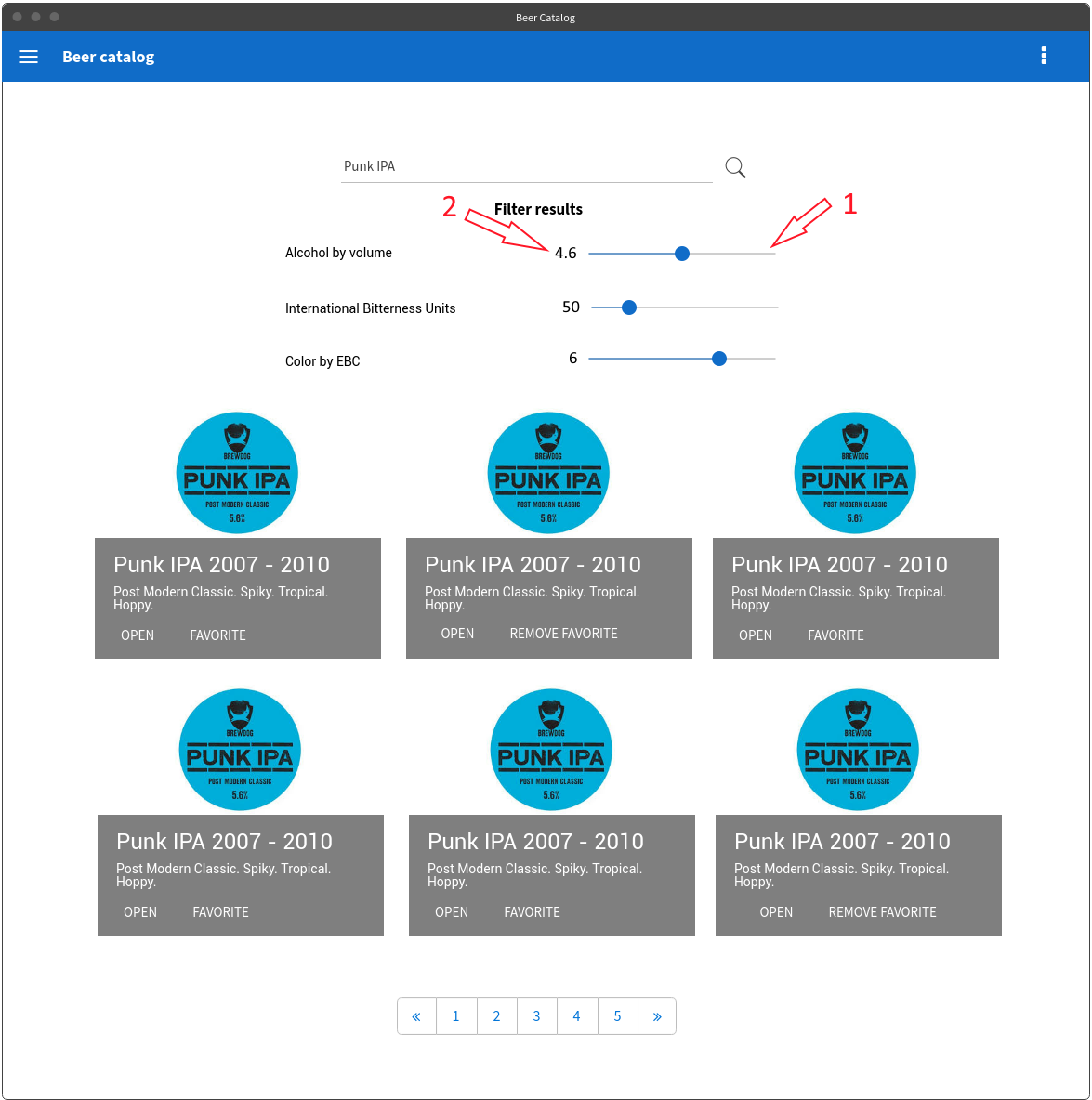
*Figure 3* is search box. It consists of an input with “Search beers…” placeholder and an icon button which performs the search. The search can also be performed by pressing “Enter” key.

*Figure 4* is a search result card. It consists of an image and overlapping panel with information about the beer:

* *Figure 5* is beer’s title;
* *Figure 6* is beer’s tagline;
* *Figure 7* is a “Favorite button”. If the beer is already in favorites list it should say “Remove from Favorites”.
* *Figure 8* is “Open” button. It leads to [2.2 Beer details page](#_2.2_Beer_details) opened for the beer.

*Figure 9* is paging panel. Note that every page should contain 9 elements (pic.1 contains 6).

After search is performed, an “Advanced Filters” panel should appear below the search box as depicted on pic. 2.



Pic. 2. Advanced Filters

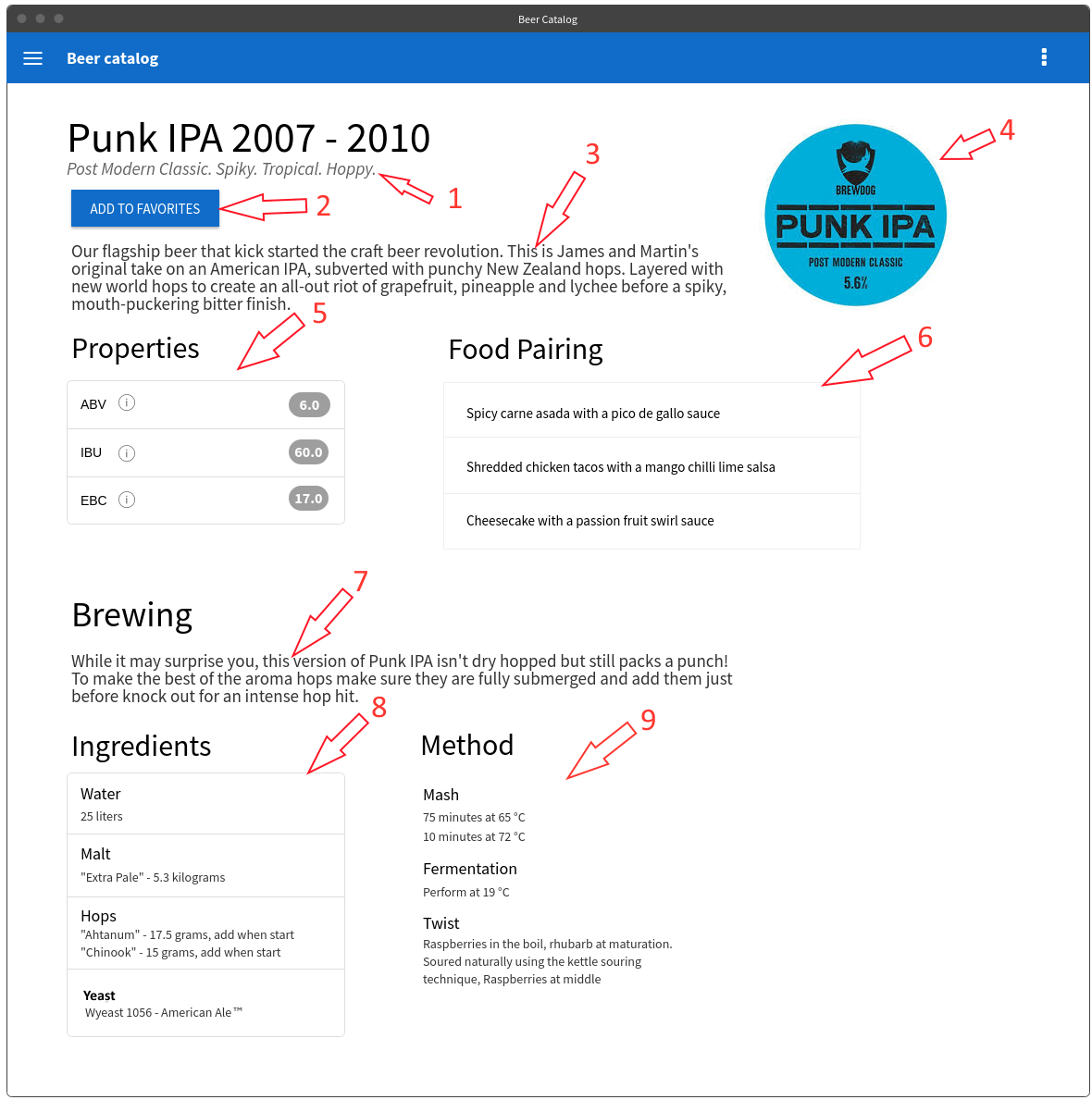
Advanced filters have three filters:

1. **Alcohol by volume** with range from 2 to 14;
2. **International bitterness units** with range from 0 to 120;
3. **Color by EBC** with range from 4 to 80.

Each filter has a slider (*Figure 1*) and a label that shows its current value (*Figure 2*).

### 2.2 Beer details page

Beer details page shows detailed information about a single beer as shown on Pic. 3:



Pic. 3. Teams page mockup

*Figure 1* is the page header. It has the beer’

*Figure 2* is Favorites button. It says “Add to favorites” if the beer is not favored and “Remove from favorites” otherwise.

*Figure 3* is the beers description.

*Figure 4* is the beer’s image taken from the feed response.

*Figure 5* is beer’s properties section. It shows only ABV, IBU and EBC. Every acronym has an info icon near. Hovering that icon leads to showing the tooltip which contains explanations on what does it mean.

*Figure 6* is food pairing list.

Beer details page has also a “Brewing” section. It consists of the following:

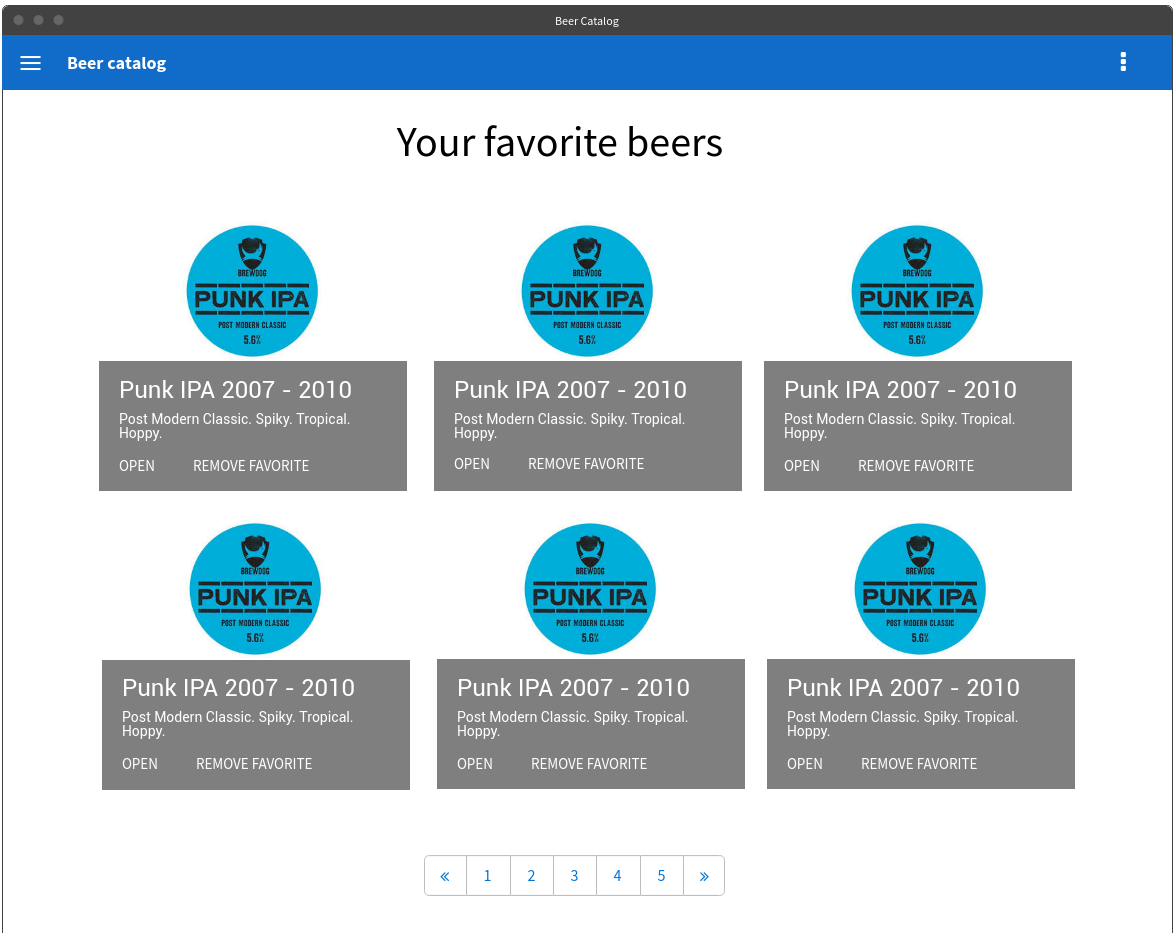
*Figure 7* is brewing tips paragraph.

*Figure 8* is Ingredients list as described in the API response.

*Figure 9* is “Method” section which describes Mash, Fermentation and Twist recommendations from the API response.

### 2.3 Favorites page

Favorites page presents a list of beers that were marked as favorites. It’s shown on pic. 4:



Pic. 4. Team page mockup 1.

This page is very similar to [2.1 Landing page](#_2.1_League_table). It has the same items and paging but doesn’t contain any search. After clicking “Remove Favorite” button the item should be immediately removed from the list.

## 3 General requirements

1. Use [Knockout JS](http://knockoutjs.com/) for interacting with the view. It has an [excellent tutorial](http://learn.knockoutjs.com/). Alternatively, use [ReactJS](https://reactjs.org/) as a view framework.
2. Use [Require.js](http://requirejs.org/) for module loading. Stick to CommonJS syntax. *In case of using ReactJS ignore this point.*
3. Create a single-page application. Provide routing (preferably your own) for it.
4. Use ES2015+ syntax, but opt for classical syntax for classes and inheritance (with prototypes and Object.create) instead of ES2015 classes – they are just syntax sugar over constructor functions*. In case of ReactJS you’re allowed to use classes.*
5. Use *XmlHttpRequest* for AJAX calls on the first phase, and *fetch API* on the second.
6. Provided pages are not designs and should not be implemented in pixel-perfect manner. Use any CSS library to style your pages. The main purpose of this task is to learn JS, not CSS.
7. Use [BEM](https://ru.bem.info/) methodology to create reusable and component-oriented CSS.
8. Use **local storage** to save user’s favorite beers.
9. Pay attention to keeping code clean and sticking to following principles: separation of concerns, single responsibility, KISS, YAGNI. The main goal of this exercise is to develop your skills in app architecture and code structuring.
10. Follow company [coding standards](https://wiki.itechart-group.com/display/codstandarts/Coding+Standard+JavaScript).
11. Do not overcomplicate your codebase so that you’ll need to build your front-end code on the first phase. *Of course, you’ll need to build your front-end in case you’re using ReactJS.*

## 4 Phases

### 4.1 Phase I

1. Implement mockups [2.1 Landing page](#_2.1_League_table) and [2.3 Favorites page](#_2.3_Team_page).
2. Use *XmlHttpRequest* and classical inheritance pattern.

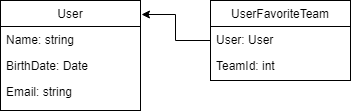
### 4.2 Phase II

1. Implement [2.2 Beer details page](#_2.2_Teams_page).
2. Switch to *fetch API* for AJAX calls. It should as easy as possible to switch to another implementation.
3. Add front-end build process to create JS and CSS bundles.
4. Switch to ES2015 classes syntax.

### 4.3 Phase III

The main goal of phase III is to practice server-side Node.js development. You will need to add Users to the application. It implies adding sign up and sign in forms, creating a profile page and moving “favorite” teams to server-side DB.

UML diagram of entities is presented on pic. 5:



Pic. 5. User UML diagram.

You will also have to get acquainted with cloud hosting environments.

Requirements for server-side are:

1. Use *express* for creating a web server (if you haven’t used it before).
2. Use [PostgreSQL](https://www.postgresql.org/) database.
3. Get acquainted with ORM concept and use any like [sequelize](https://www.npmjs.com/package/sequelize).
4. Use any free email-sending service to send confirmation emails to users when signing up.
5. Use [Heroku](https://www.heroku.com) or [Azure](https://azure.microsoft.com) for application and database hosting.

### 4.4 Phase IV. Complex application

Now that we have football data and Users in the same place, we need to add prediction tournaments and Docker. TBD.