# 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

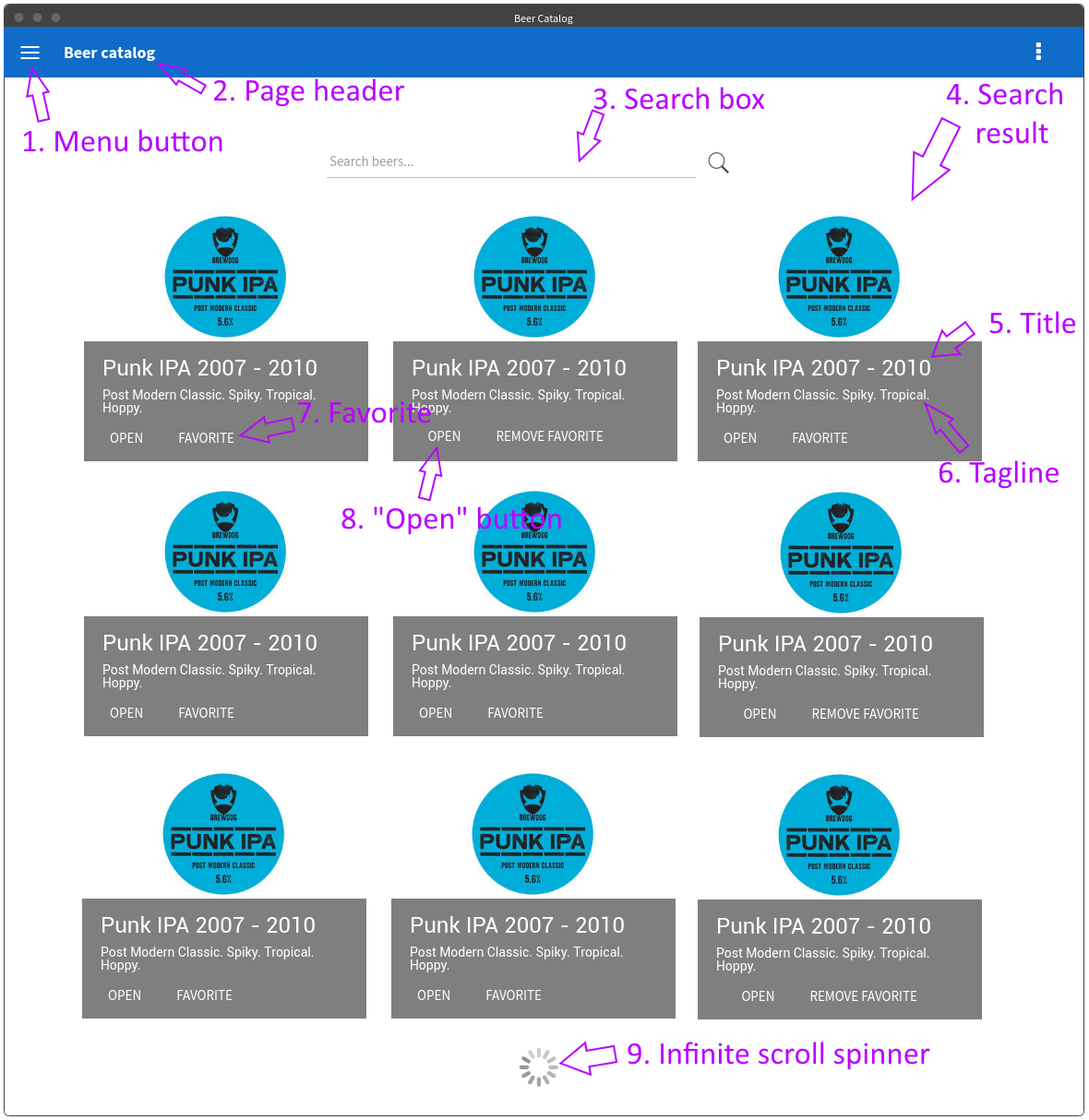
[Twitter API](https://dev.twitter.com/rest/public) will be used for retrieving tweets by club name. So the Search API will be used. To authenticate in this API you’ll need to use your own Twitter account. If you don’t have one, you’ll need to create one.

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

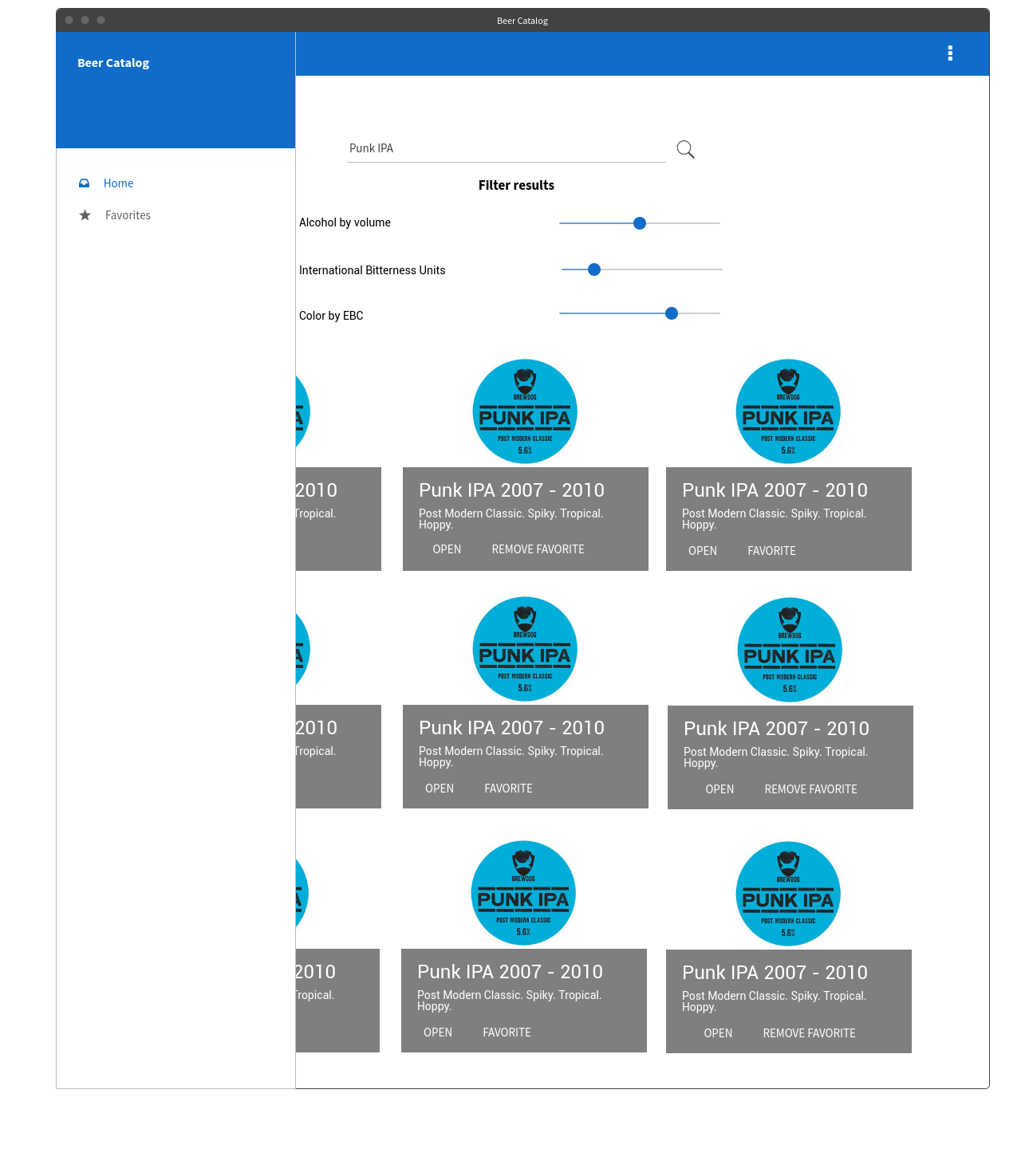
### 2.1 Landing page

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 and 2.3 Favorites page. It should look similar to this:



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

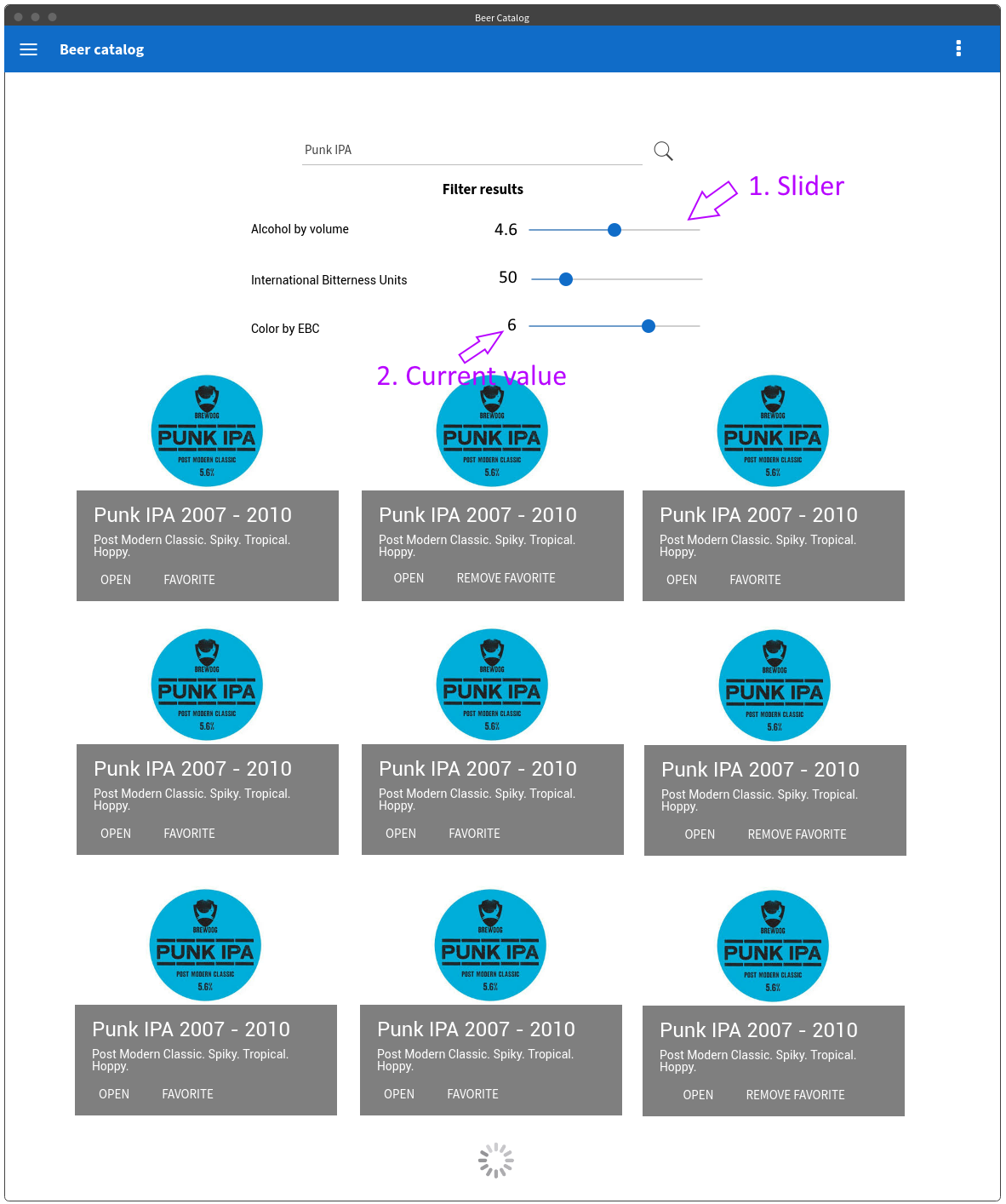
*Figure 3* is search box. It consists of an input with “Search beers…” placeholder and an icon button which performs the search. The search can also be performed by pressing “Enter” key.

*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 infinite scroll spinner. The feed doesn’t support paging, so infinite scroll (your own) should be used instead.

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



Pic. 2. Advanced Filters

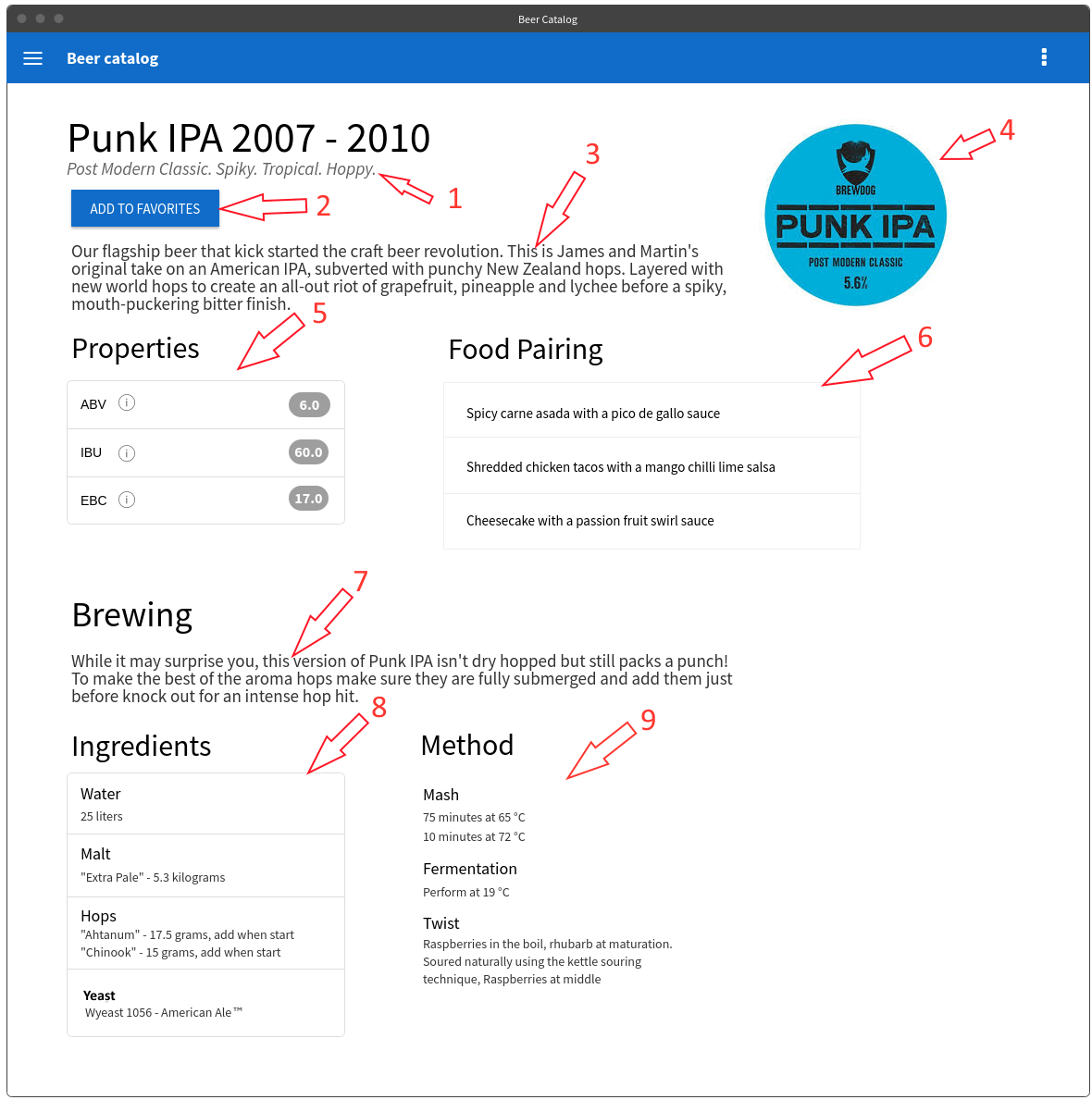
Advanced filters have three filters:

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. Beer 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. Favorites page mockup.

*Figure 1* and *Figure 2* are Beer's Title and Tagline, the same as on the 2.1 Landing page.

*Figure 3* is Beer's description.

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

*Figure 5* is action buttons panel. It contains "Open" and "Remove favorite" buttons. "Open" button leads to 2.2 Beer details page. "Remove favorite" removes the current beer from the favorites list.

*Figure 6* is paging panel. Since this list is something we have control over, it should be properly paged. Default page size is 5. If the number of items is less than one page, the panel should be hidden.

## 3 General requirements

1. Use [ReactJS](https://reactjs.org/) or [Vue.js](https://vuejs.org/) as a view framework.
2. Use [Webpack](https://webpack.js.org/) or [Rollup](https://rollupjs.org/guide/en)*.*
3. Create a single-page application. Provide routing for it.
4. Use ES2015+ syntax.
5. Use *XmlHttpRequest* for AJAX calls on the first phase, and *fetch API* on the second and later on.
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](https://en.wikipedia.org/wiki/Separation_of_concerns), [SOLID](https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)), [KISS](http://enterprisecraftsmanship.com/2015/06/15/kiss-revisited/), [YAGNI](http://enterprisecraftsmanship.com/2015/06/11/yagni-revisited/), [DRY](http://enterprisecraftsmanship.com/2015/09/11/dry-revisited/). The main goal of this exercise is to develop your skills in app architecture and code structuring.
10. Use [ESLint](https://eslint.org/) and [Airbnb guidelines](https://github.com/airbnb/javascript).
11. **Do not use** components libraries.
12. **Do not use** JQuery.

## 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*.

### 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.

### 4.3 Phase III

The main goal of phase III is to practice server-side 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 beers to server-side DB.

Users have names, birth dates, emails and a profile picture.

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

Requirements for server-side are:

1. Use [express.js](http://expressjs.com/ru/)  or [.NET Core](https://docs.microsoft.com/en-us/dotnet/core/) for creating a web server.
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 [Cloudinary](https://cloudinary.com/) for storing images.
5. Use any free email-sending service to send confirmation emails to users when signing up. *(optional)*
6. Use [Heroku](https://www.heroku.com) or [Azure](https://azure.microsoft.com) or [AWS](https://aws.amazon.com/ru/) or [AppHarbor](https://appharbor.com/) for application and database hosting.

### 4.4 Phase IV. Complex application

As we have users and their favorite beers in the DB, we are free to start logging brews.

A new entity is added: **Brew**. A brew is a log of an event of brewing one of the beers. It has a direct relation to the beer entity. A brew entry has:

1. Date and time;
2. Location (optional);
3. Ingredients used (pre-filled with data from the beer entity from the feed);
4. Brewing method deployed (again, pre-filled with the data from the feed);
5. Brew name (pre-filled with the beer name from the feed);
6. *Optional*: images collection that depict the process and the result;
7. Impressions: this is a free text field.
8. Beer type

All brew entries can be **commented and rated** by users (using +/- system like 9gag). Comments should be updated using [WebSockets](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API). But updates shouldn't be shown immediately – instead, a "load" button is shown. Clicking this button triggers loading new comments.

Users should be able to enter their beer type preferences using an autocomplete field with a possibility of free text enter. That means that the autocomplete suggests suitable values based on the existing ones, but the user should be able to enter new ones.

Users are able to view their brew feeds. The feed consists of the latest top-rated brews that can be interesting for the user based on their beer preferences. This means that brews are *filtered by user preferences*, *grouped by date* and *ordered by rating* inside those groups. Each list item displays brew name, date and location, rating and one of the photos (the first).

Default beer types list:

* Ale
* Lager
* Stout
* Porter
* Lambic
* Pilsner
* Pale Ale
* Weissbier
* Belgian Ale