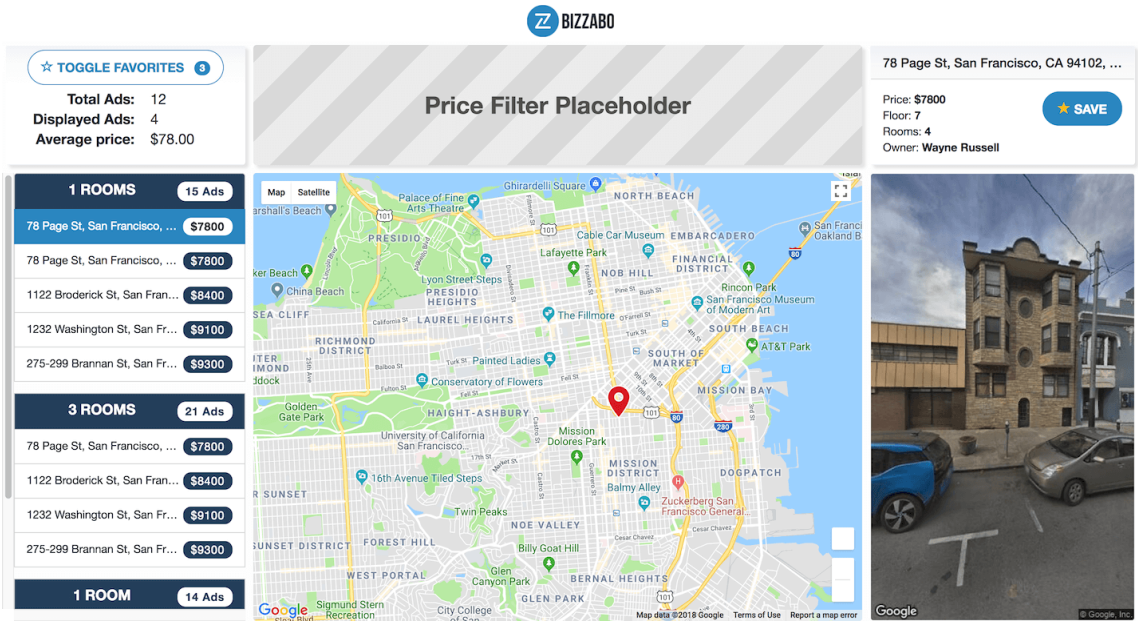
Table of Contents:



What do you need to do?

You're given a partially implemented web app that allows users to browse ads of apartments for rent in San Francisco.

You're given a Features Spec you'll need follow and add all the missing functionality.
The project's client app, contains the UI components without any of the logic.
You can implement it any way you see fit, including adding any 3rd party library you want.

> *It's up to you to connect everything and make it all come together.*
> *Of course you're free to change or add whatever you see fit.*

Your tasks are:

- **Implement all the missing logic in the app according to the spec**
- **Create a component from scratch**
- **Bonus Task - Favorites Feature**

Implement all the missing logic in the app

Your task will be to implement different features for this app, according to the Features Spec.
Let's briefly go over them:

- **Rent Ads Map**
  - Shows a marker for each rent ad.
  - Clicking on a marker shows the ad's details on the right **Info Panel**.
- **Rent Ad Info Panel**
  - Shows the ad's details and a Street View image of the ad.
- **Rent Ads List**
  - Shows an ordered list of rent ads.
  - Clicking on an item in the list shows the ad's details on the right **Info Panel**.
- **Summary Panel**
  - Shows a few stats about the ads
- **Price Range Filter**
  - Filters the price range of the displayed ads on the map and the ads list.
- **Data Fetching**
  - The app shows the `latest 50` ads in `active` status.
  - Use the **RentAdsAPI** to load the rent ads data.

Create the Price Range Filter

Unlike the other components, the Price Range Filter doesn't have a basic UI - only a placeholder for it should be, above the map.
Your task will be to create this component from scratch.

It should work as follows:

- Allow the user to filter the ads by min/max price range.
- Changing the price range will change the `Displayed Ads` on the map and list.
- The minimum price is **$500**.
- The maximum price is **$10000**.
- For simplicity, you can assume the filter only filters the `50` ads you're currently viewing.

Other then that, you can implement it any way you see fit, including using any 3rd party component you want.

Bonus Task - Favorites Feature

Your task will be to implement the Favorites feature - users can mark an ad as a favorite and also view all favorites.
Your task will be to implement the following components:

- **Save/Remove Favorite Button**
  - The **Rent Ad Info Panel** also includes a `Save/Remove Favorite` button which should save/remove the ad from the user's favorites.
- **Toggle Favorites Button**

- The **Summary Panel** includes the `Toggle Favorites` button.
- It switches between showing only the favorites or showing all ads.

- **Total Favorites Counter**
  - The `Toggle Favorites` button includes the `Total Favorites` counter.
  - It shows the count of all the users favorite ads.

Features Spec

This is the main page in our app. It contains a map and list which lets users browse apartments for rent in San Francisco.
The app shows the `latest` `50` ads in `active` status.

Rent Ads Map

This is a **GoogleMap** which displays a marker for each ad.
A map marker has 3 states: `Regular / Selected / Hovered` - each with its own icon.
Hovering over a marker changes its state to `Hovered`.
Clicking a marker on the map:

- Shows the ad's details on the **Rent Ad Info Panel**.
- Changes the marker to its `Selected` state.
- Changes the ad's list item to its `Selected` state.

Rent Ads List

Shows a list of rent ads grouped by number of rooms, in ascending order.
Each group is sorted by rent price also in ascending order.
Clicking on an item in the list:

- Shows the ad's details on the **Rent Ad Info Panel**.
- Changes the list item to its `Selected` state.
- Changes the ad's marker on the map to its `Selected` state.

Hovering on an item in the list changes the ad's marker on the map to `Hovered` state.

Rent Ad Info Panel

The side info panel shows the currently selected ad info along with a **Google StreetView** image of the address.

> *Note, the `Save/Remove Favorite` button is part of the Bonus Task*

Summary

In the top left there's the summary panel which contains:

- `Total Ads` counter - the count of all the `latest` `50` ads in `active` status.
- `Displayed Ads` counter - the count of all of the ads that should be displayed on the map and in the list after applying the filter.
- `Average price` counter - the average price of all `Displayed Ads`.
  > Note, the `Toggle Favorites` button and `Total Favorites` counter is part of the [Bonus Task](#bonus-task)

Application Overview

There are code examples in each React component which demonstrates how to use the existing UI.
The example loads the map with an example marker, displays an example ad on the info panel and example data on the ads list.

> *It's up to you to connect everything and make it all come together.*
> *Of course you're free to change or add whatever you see fit.*

Server

The server is a simple `Express` app which loads the bundled `React` app on every un-mapped route (i.e `APP_URL/*`)

> *You can start the server by running `npm run start` from your **Kimbr Console**.*

Client

The client is a `React` + `Redux` application, with the minimal UI components already in place.
The main client application is located at the `src/app/` folder.

Using `Redux` is **optional** and you can use it in any way that suits you, including adding any 3rd party you want.
Also, `react-redux` is included in advance, but you're not required to use it.

> *You can build the client app by running `npm run build` from your **Kimbr Console**.*

Fetching Data - RentAdsAPI

To fetch the `RentAds` you'll need to use `RentAdsAPI`, located at `src/app/services/rent-ads-api.js`.
The `RentAdsAPI` is a **fake** API which simply loads mock data and returns a list of RentAd objects.
For simplicity, you can use it directly from the client, without going through the server.

Example: ```js RentAdsAPI.fetchAds().then((rentAds)=> { //... }); ```

> *Note, the soon-to-be-deprecated `RentAdsAPI` returns a list of **unsorted** and **unfiltered** ads.*

RentAd Model

Represents a single ad of an apartment for rent.
`RentAd` is a simple data wrapper class, located at `models/rent-ad.js`.
A `RentAd`'s status can be either `active` or `closed`.

Example data:

```
{
  "id": 321,
  "status": "active",
  "rentPrice": 7800,
```

```
  "latitude": 37.774345,
  "longitude": -122.42203,
  "rooms": 4,
  "squareMetersSize": 95,
  "floor": 7,
  "numberOfFloors": 9,
  "hasParking": true,
  "owner": "Wayne Russell",
  "phone": "1-(862)153-3335",
  "address": "78 Page St, San Francisco, CA 94102, USA",
  "message": "Amazing apartment, call now!",
  "createdAt": "2016-09-22T20:00:14Z"
}
```


***

References

- ES6 Features
- Express
- Lodash
- React
- Redux
- Redux Ecosystem
- React-Redux
- React Google Maps
- Google Maps JavaScript API
- Bootstrap