Web-school

Home task

React -Front Camp

Contact Us

Some text with an important information

Phone: [Telephone]
Email: [Email address]

<epam>

React task

Definition

The idea is to create a single page application, which will allow users to search the Movies DB database.

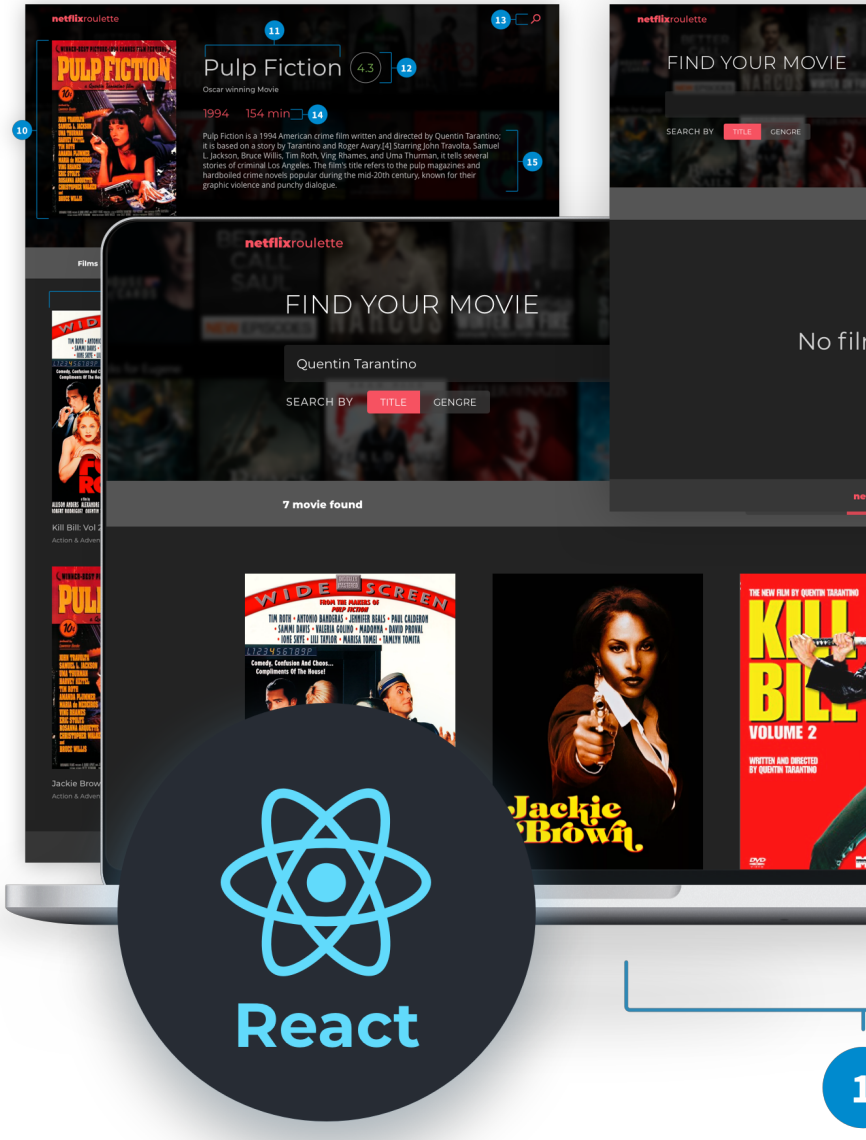
Api swagger documentation:

https://reactjs-cdp.herokuapp.com/api-docs

There will be two pages: main page with search and a page with a particular film.

Interface

Here are two screens presented with break down. There is also an additional screen with empty state of the main page. Design and captions are available below.



Materials

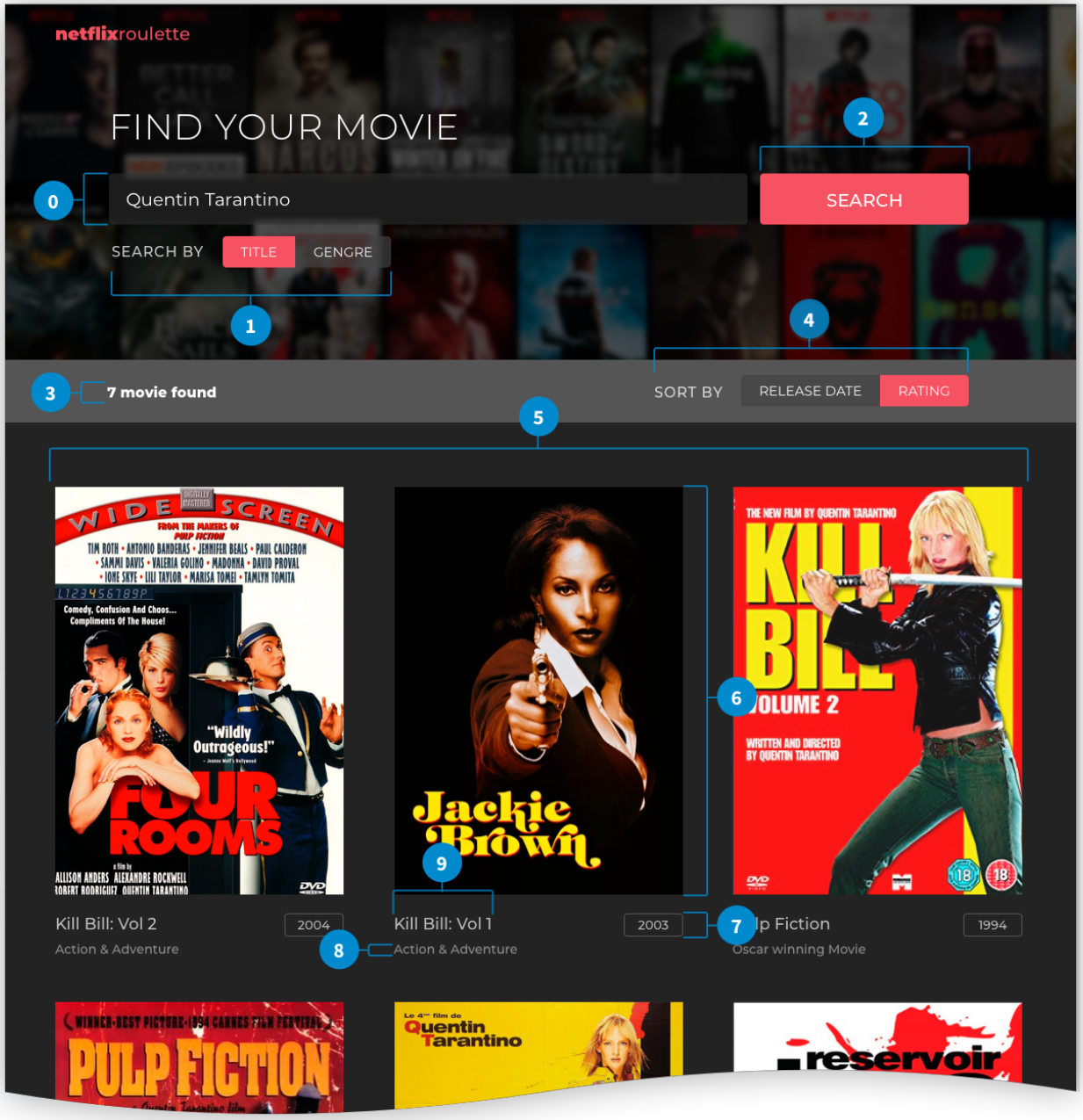
InVision prototype

https://epam.invisionapp.com/share/HCRLBL4465S

All mockups

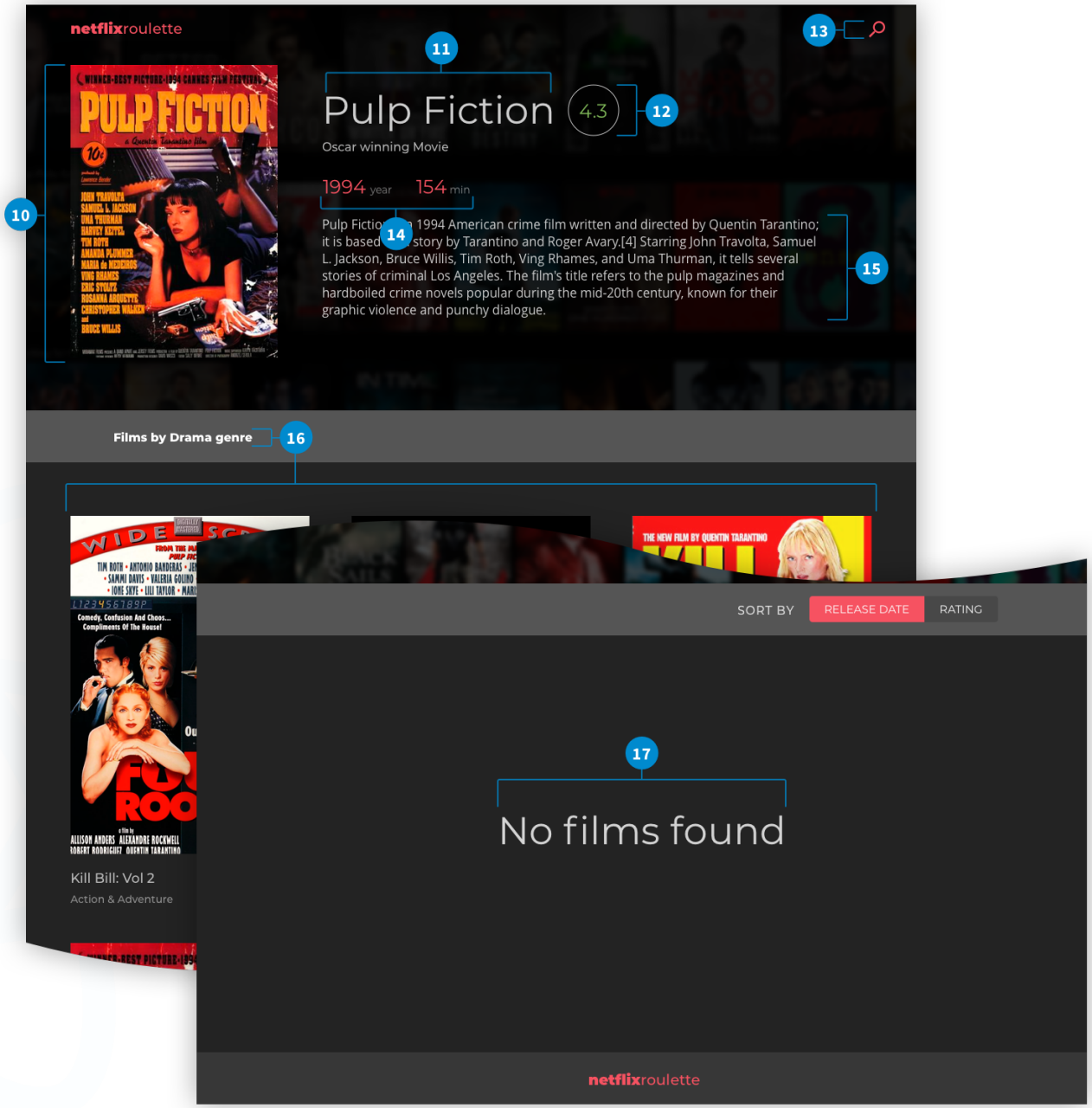
https://www.dropbox.com/sh/twi7jwikg32pyll/AABwgADCi1dpu8vWuV_BCO6Oa?dl=0

Captions



Details

0. **Search field**
   Enter button should work as well
1. **Search filter**
   By default “Title” is selected.
   Click to switch option.
2. **Search button**
3. **Results count**
4. **Results sort**
   “release date” is selected by default. Click to switch option.
5. **Results body**
   All found items, without pagination
6. **Item image**
   URL for the image you will get from the server
7. **Item release date**
8. **Item genre**
   Enter button should work as well
9. **Item title**
   By default “Title” is selected.
   Click to switch option.



Details

10. **Film cover**
11. **Film title**
12. **Film rating**
13. **Search button**
    Returns user to the main page with search
14. **Film duration and release year**
15. **Description**
16. **Film by the same genre**
17. **Empty result state**

Task 1

Create `package.json` file and install `React, Redux, React-Redux, React-Router, Jest.` Install and configure webpack & babel to get build artifact by running `npm` command.

Write components implementing `HTML` markup for required design (see images at the beginning of the document). For this part, no need to implement API calls and routing, the task can be done with mocked data.

Use `<ErrorBoundary>` component for catching and displaying errors https://reactjs.org/docs/error-boundaries.html. You could create one component and wrap all your **application or** use several components.

Use smart/dumb components approach

100% decomposition (estimated by mentor)

Evaluation criteria*

2 — Installed Webpack, Implement all the required markup, Markup is done with `React Components`

3 — Use `<ErrorBoundary>` component for catching and displaying errors

4 — Use `smart/dumb components` approach

5 — 100% decomposition (evaluated by mentor)

* each mark includes previous mark criteria

Task 2

1. Go through API docs in swagger: https://reactjs-cdp.herokuapp.com/api-docs
2. API Endpoint: https://reactjs-cdp.herokuapp.com/
3. Make your components perform real AJAX requests.
4. Move data fetches to actions and pass data to your components with `redux`.
5. Create routing for your application. Link app states between each other with React router. By default user lands on a new page with empty results state (caption 17). When user clicks on a film item, redirect him to:

```
localhost/film/id
```

6. Handle invalid URLs, display create a 404 page, where user will be redirected in case of invalid URL. On switching search type or sorting type you shouldn't switch any routes.
7. When user performs a new search, you should redirect him to:

```
localhost/search/Search%20Query
```

8. When a new user lands on the page with such URL, you should perform search and display results.

Evaluation criteria*

2 All data fetches are implemented with real requests

3 All data fetches moved to redux actions & received from store by components

4 Filtering and sorting is done as redux actions

5 Actions and reducers covered with unit tests (~60%+, can be amended by mentor)

* each mark includes previous mark criteria