Mock-up: Netflix-Clone

Learning Units:

- Vue-Directives
- Binding Classes
- Binding Images
- Binding elements to properties
- Computed Properties
- Methods
- Lifecycle Hooks
- Objects - state management, updating
- Arrays - filtering, sorting
- Storage API- localStorage, JSON

Brief:

As the popularity of Netflix has been soaring during the last few years, everyone is aware of video streaming apps and the demand for such kinds of apps at present times. In this project you are required to showcase all of the frontend skills you have learned up until now, and build a media-type app similar to Netflix, a “Netflix-Clone” if

you will. These types of applications are growing in popularity and require most of the fundamental frontend developer skills in order to be built, thus making them great choices for Capstone Projects.

Have a look at the provided wireframe and technical requirements needed to pass this project. There a list of additional features that can be implemented in order to improve your project, however you are not limited to any of these specifications and if you wish to add your own features you definitely encouraged to do so, as this will reflect well on you as a prospective developer and strengthen your portfolio of projects.

Overview:

Create an application that has a similar look and feel, colour theme, layout, etc. to the actual Netflix application. Your core functionality will revolve around displaying movie objects on the Landing page, adding movies to your watch-list and displaying them on your Watch-List page, and finally displaying movies the thumbnails of movies that are “coming-soon” in the hero of the Landing page, not being allowed to display them with the available movies, or add them to the watch-list until they are made available.

Each page should have its own Vue instance and data should be shared across pages using localStorage. If you are planning to create an SPA you can engineer your own project/file structure, but a recommended one would be to have one main Vue instance and use functions and import/export syntax to load your components in.

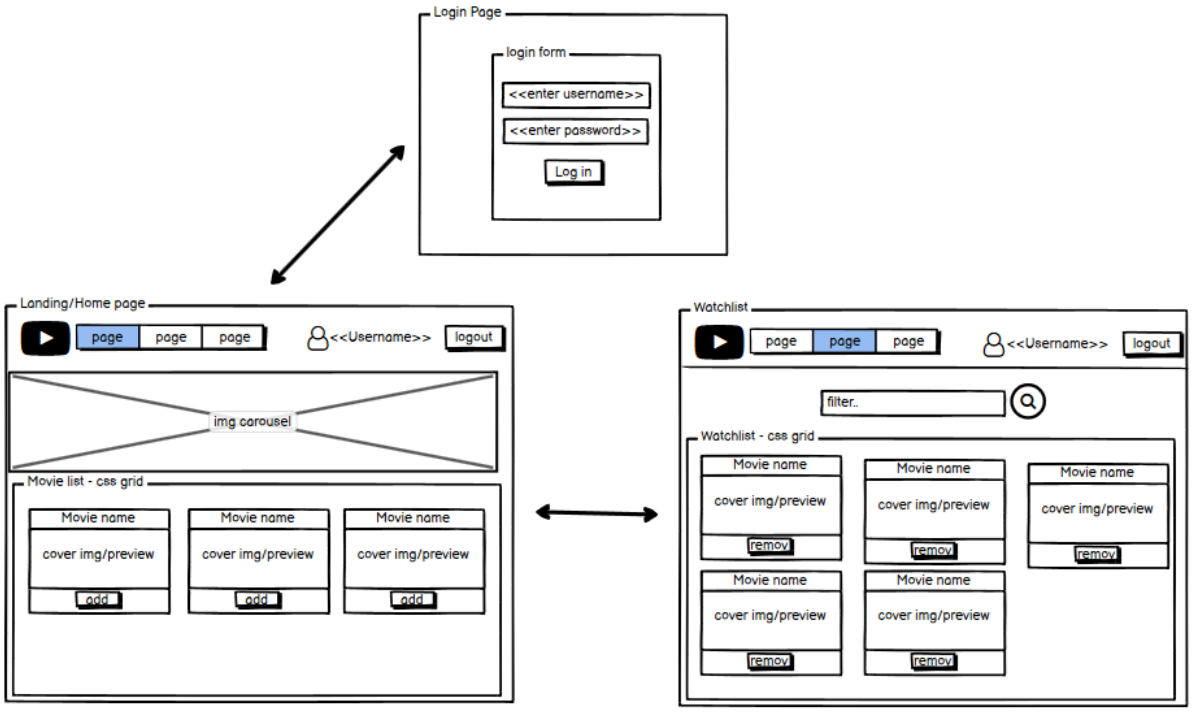
Remember to make extensive use of localStorage to move data across pages, and to ensure that watchlist movies remain consistent even after the application closes. If you choose to create a single-page application you will most likely need to update and call the localStorage a lot more frequently.

Wireframe:

This wireframe and flow-diagram is a basic layout guide. You do not have to follow it strictly if you are confidant in your own abilities, or want to expand the number of

pages in your application.

Standard Website Format:



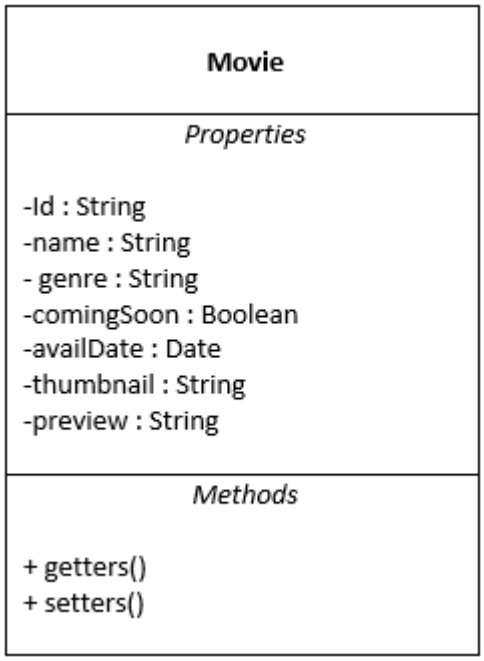
Object Specification:

Below is the required members and properties for the Movie objects. The UML class diagram serves as a sample of what a more object-orientated approach would look like.

Movie Properties

- id : String
- name : String
- genre : String
- comingSoon : boolean
- availDate : Date
- thumbnail : String (url for image)
- preview : String(YT link for IFrame)

UML Class Diagram



Technical Requirements:

Login Page

1. This is the only compulsory page in the site that does not require a Vue instance. When a user clicks the login button, your app must save their username to localStorage so it can be displayed on other pages.
2. You will need to JavaScript to redirect the user to the home page when clicking on the login button as the <a> tag won't be a viable option (since it will change pages before the data can be saved to localStorage).
3. When a user click on any logout button they must be redirected to the login page and once this page loads the logged-in user's details must be cleared.

Landing Page

1. **Populate your application with Movie objects**

   Create an object Array that will be used for storing all of your Movie objects

Populate this array with at least 20 Movies (look at the Object Specification for the list of members/properties)

You also are not required to use classes for this project, object literals are accepted as well.

2. **Create a computed property for Movies that are coming-soon**

   Out of the movies that you created inside the movies list, have at least 4 of them have a comingSoon value of “true”.

   Create a computed property that will return all movies that coming soon if their member/property of comingSoon === true.

3. **Create a computed property for Movies that are available**

   Create another computed property that returns a list of movies that are available if their member/property of comingSoon is === false.

   Only these movies should be allowed to be added to a user’s watchlist.

4. **Presentation Concerns**

   In the hero of your page create an image carousel that displays the thumbnails of all movies that are coming soon. Over the whole carousel display a banner with the text “Coming Soon..”. Try to style the banner so it looks like a watermark, but if you are unable at least have the banner display over the carousel.

   Inside the main area of the landing page display all of the movies that are currently available to checkout and add to the watch-list. Have a look at the wireframe for basic inspiration on how to present your movies, but you can use any CSS framework you prefer, or use vanilla CSS for layouts. Try to account for mobile devices and include media queries, flexbox, smaller fonts, etc. When a movie’s thumbnail is hovered over you need to display a button that when clicked adds that movie to the watch-list (you will need a method for this watch-list adding). The button must not display if the user is not hovering over the movie’s thumbnail.

5. OPTIONAL:

   a. Before adding an item to the watch-list, write some logic that first checks whether the item already exists in the watch-list so we can ensure we don’t have any duplicate movies being displayed.

   b. Write a function that updates the comingSoon property of an unreleased movie when the availDate of the movie is the same as the present day in

real life.

Watch-List

1. When the Watch-List page loads ensure to read all the movies stored in the watch-List localStorage object and display them in a similar fashion as you displayed the available movies on the landing page.
2. Give each movie on the watch-list a delete button that when clicked removes and item from the watch-list. This should reflect on your localStorage as well.
3. Write a function that takes in keywords in the search bar, filters through the watch-list, and only displays the movies that includes the string typed into the search bar.
4. OPTIONAL:
    a. Sort all the movies in the watch-list by their names alphabetically