Movi: movie & tv recommendation service

Janae, Ryan, Dom

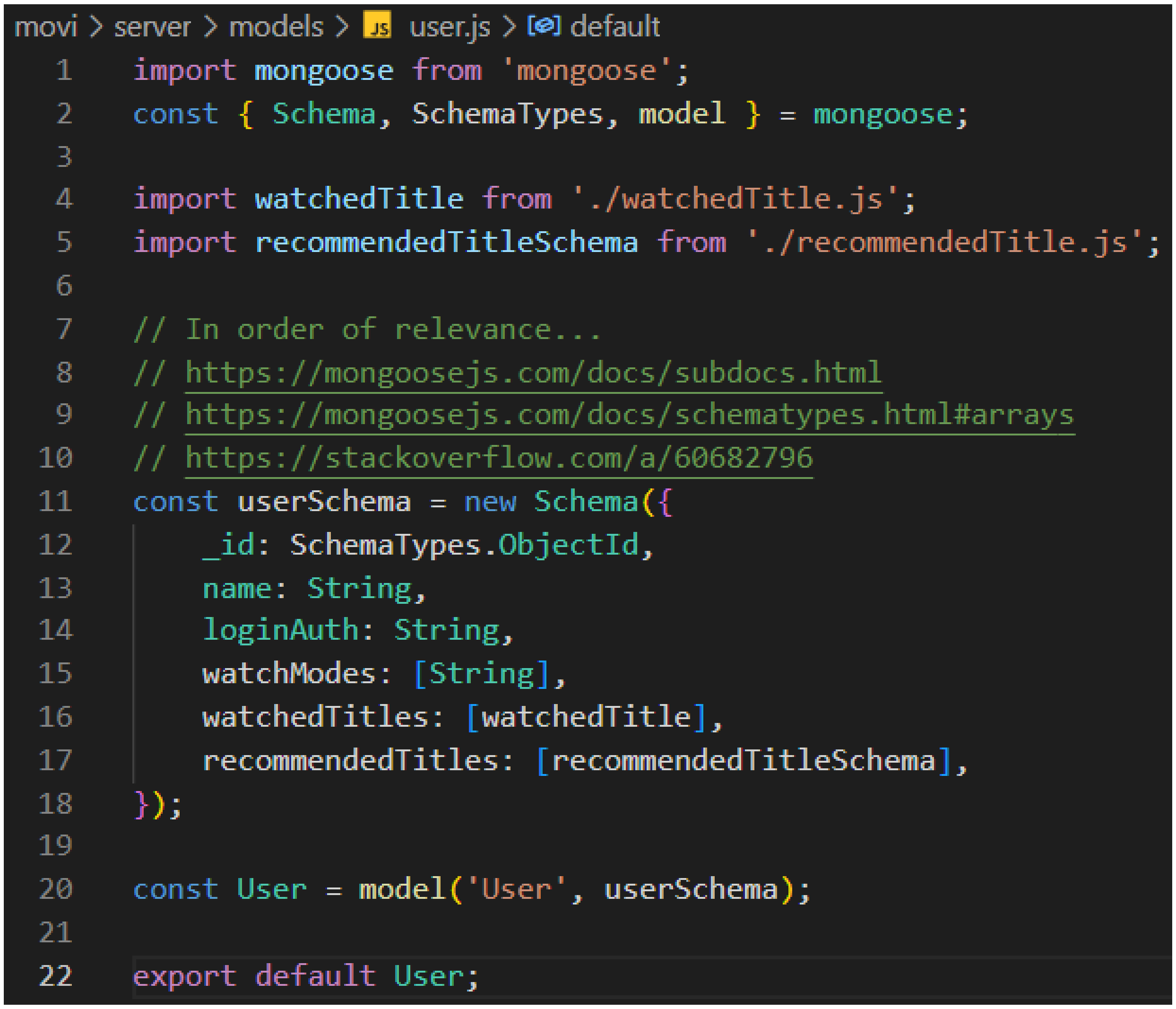
Update 1

GitHub: https://github.com/jjm8759/MOVI

MongoDB

One of the core technologies in our project is the NoSQL MongoDB database. For update 1, we have created the database and hooked it up to our node project.

We currently have a single document collection titled “Users” with the following schema and sub-schema definitions using mongoose:



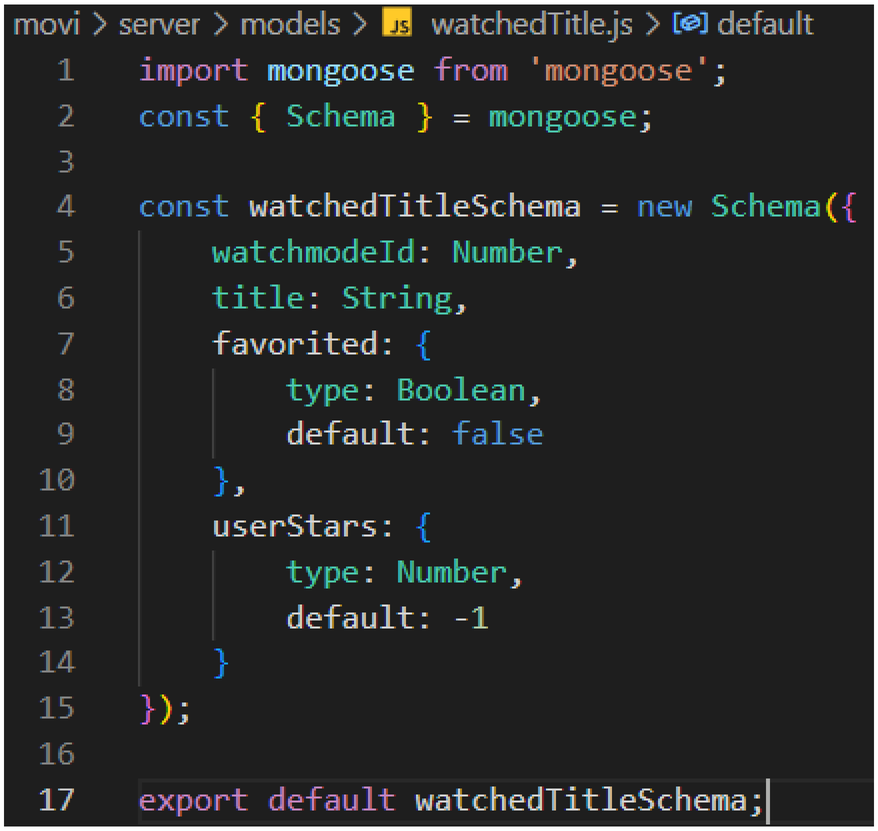
```
import mongoose from 'mongoose';
const { Schema, SchemaTypes, model } = mongoose;

import watchedTitle from './watchedTitle.js';
import recommendedTitleSchema from './recommendedTitle.js';

// In order of relevance...
// https://mongoosejs.com/docs/subdocs.html
// https://mongoosejs.com/docs/schematypes.html#arrays
// https://stackoverflow.com/a/60682796
const userSchema = new Schema({
    _id: SchemaTypes.ObjectId,
    name: String,
    loginAuth: String,
    watchModes: [String],
    watchedTitles: [watchedTitle],
    recommendedTitles: [recommendedTitleSchema],
});

const User = model('User', userSchema);

export default User;
```

Each **User document** has the following:

- An objectId reference
- A name associated with the user
- A login_auth string (may or may not be needed)
- An array of strings holding which streaming services the user has access to
- An array of *watchedTitle* sub-schemas representing titles they have watched
- An array of *recommendedTitles* sub-schemas representing which titles have been recommended to them

Login auth: this field will store the user's login authentication data.

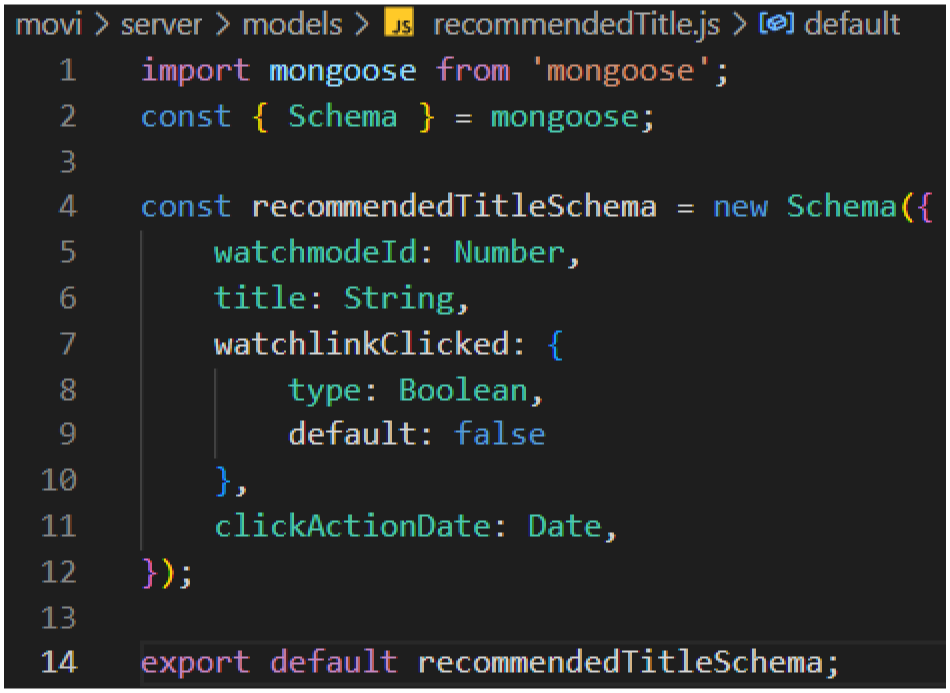
- Depending on our authentication method, this may not be needed


movi > server > models > watchedTitle.js > default

```javascript
import mongoose from 'mongoose';
const { Schema } = mongoose;

const watchedTitleSchema = new Schema({
    watchmodeId: Number,
    title: String,
    favorited: {
        type: Boolean,
        default: false
    },
    userStars: {
        type: Number,
        default: -1
    }
});

export default watchedTitleSchema;
```

A Watched Title has the following information related to a movie the user has marked as watched:

- **The integer id of the movie in the WatchMode API** (the WatchMode API stores them as integers)
- **The title of the movie** (for view convenience)
- **Bool for whether the user has favorited the movie** (giving it more weight towards future recommendations, defaults to false)
- **The star rating the user has given the movie** (also used in the recommendation algorithm, defaults to -1, or unrated)



```
import mongoose from 'mongoose';
const { Schema } = mongoose;

const recommendedTitleSchema = new Schema({
    watchmodeId: Number,
    title: String,
    watchlinkClicked: {
        type: Boolean,
        default: false
    },
    clickActionDate: Date,
});

export default recommendedTitleSchema;
```

Recommended titles array: each array element references an object containing the following information related to a movie that has been recommended to a user:

- **The integer id of the movie** (the WatchMode API stores them as integers)
- **The title of the movie** (for convenience)
- **Bool for whether the user has clicked a watch link** (to prompt them later to give the movie a rating, adding it to their watched collection. Defaults to false)
- **The Date object of when they clicked the watch link**

WatchMode API

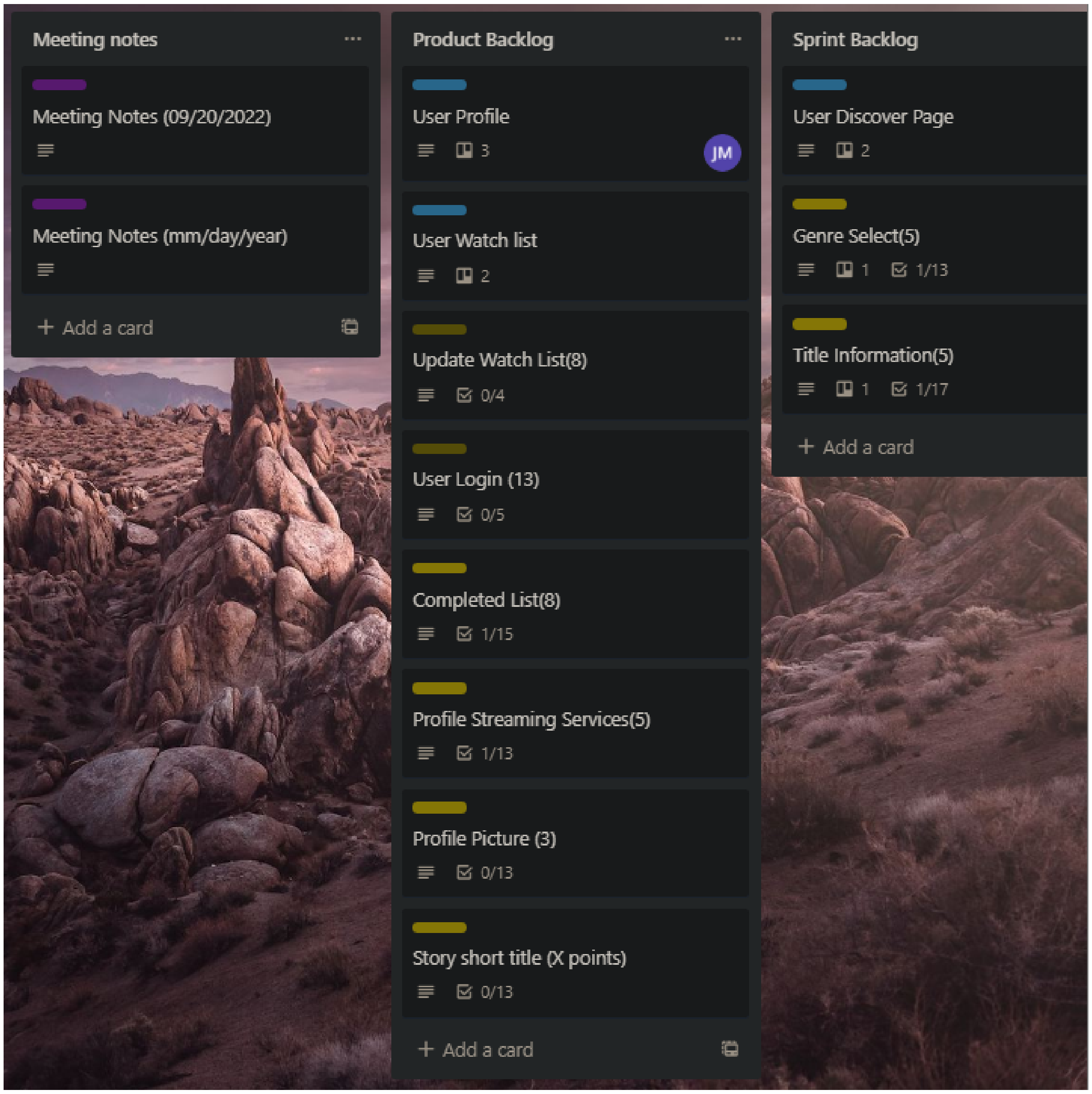
We have chosen to use the movie API *WatchMode* which serves standard title metadata, API-hosted permalinks to poster images, and comprehensive data relating to the streaming platforms the title is available on. The API has clean and clear documentation, as well as free live chat support. We have secured our API key, downloaded a copy of the database for good measure, and have begun pulling the API into the backend for a taste of how it will work with the project.

Back End

We have implemented a basic Node project back end and react project front end and installed the immediately necessary dependencies on both sides. We have also set up the MongoDB database connection using our URI. We are still generating class diagrams and a more detailed domain analysis diagram.

Trello

We have a Trello set up with sprint stories and epics. We have lists for meeting notes, product backlog, sprint backlog, in development, in need of testing, being tested, and done.



Meeting notes

Meeting Notes (09/20/2022)

Meeting Notes (mm/day/year)

+ Add a card

Product Backlog

User Profile
3
JM

User Watch list
2

Update Watch List(8)
0/4

User Login (13)
0/5

Completed List(8)
1/15

Profile Streaming Services(5)
1/13

Profile Picture (3)
0/13

Story short title (X points)
0/13

+ Add a card

Sprint Backlog

User Discover Page
2

Genre Select(5)
1 1/13

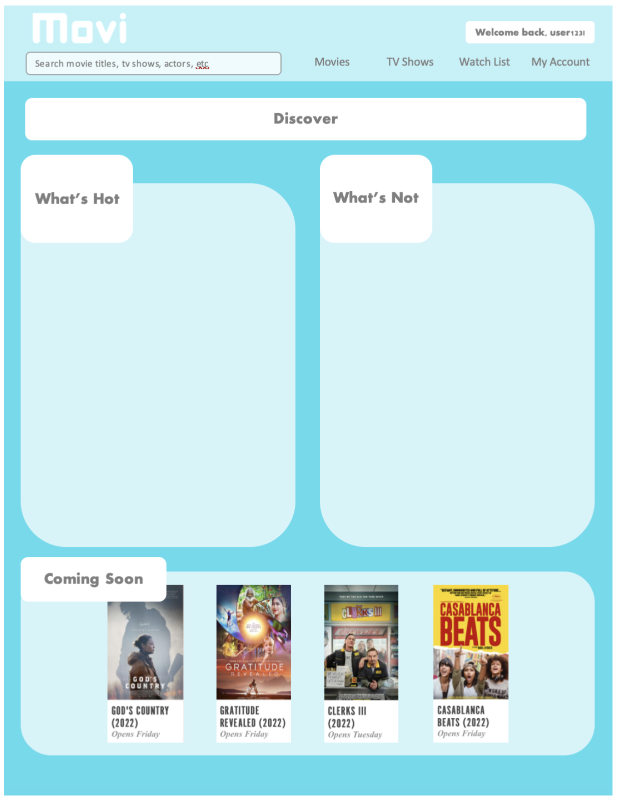
Title Information(5)
1 1/17

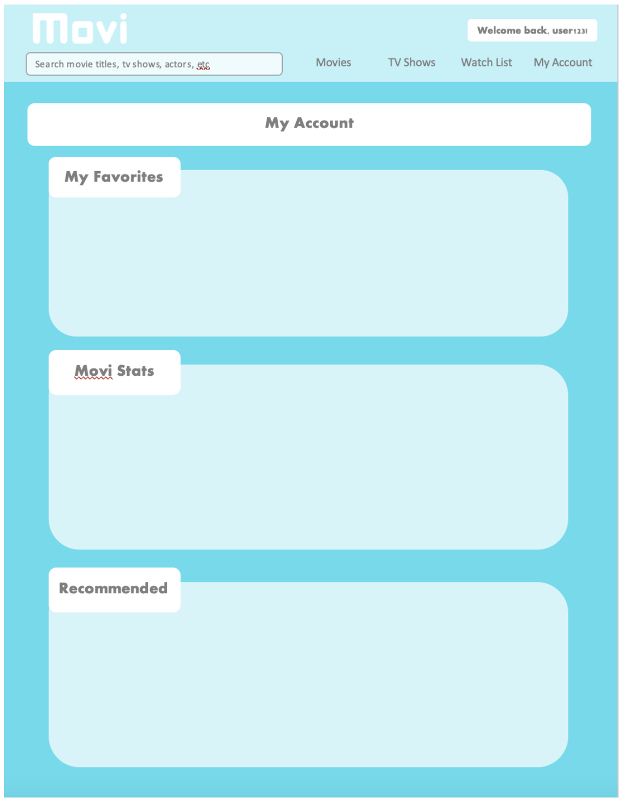
+ Add a card

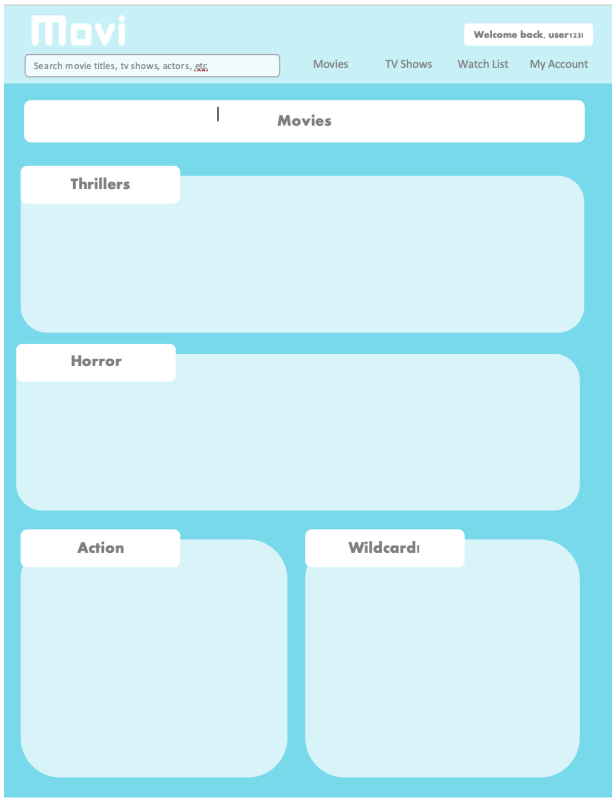
Front End

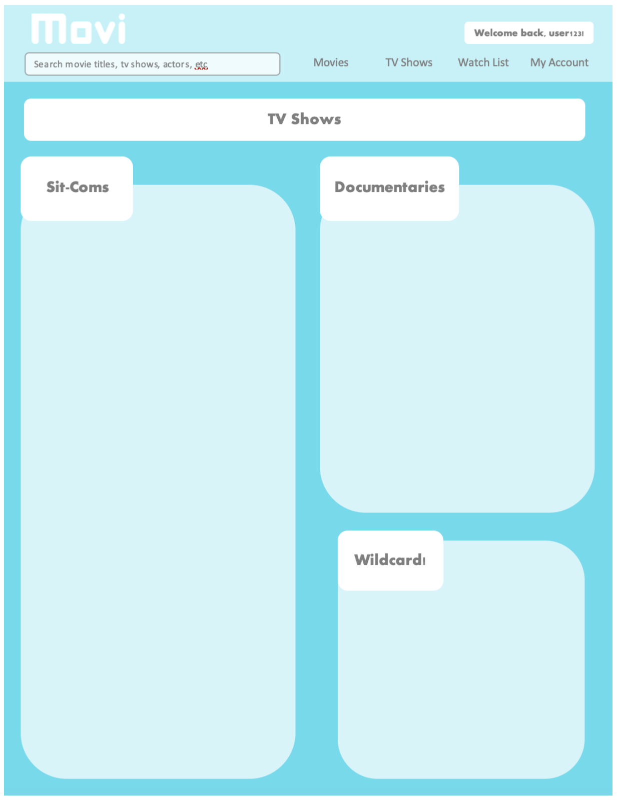
We have begun working on a basic implementation of the front-end wireframe, including a watched section and recommended section, as well as the start of a login system using Google OAuth.

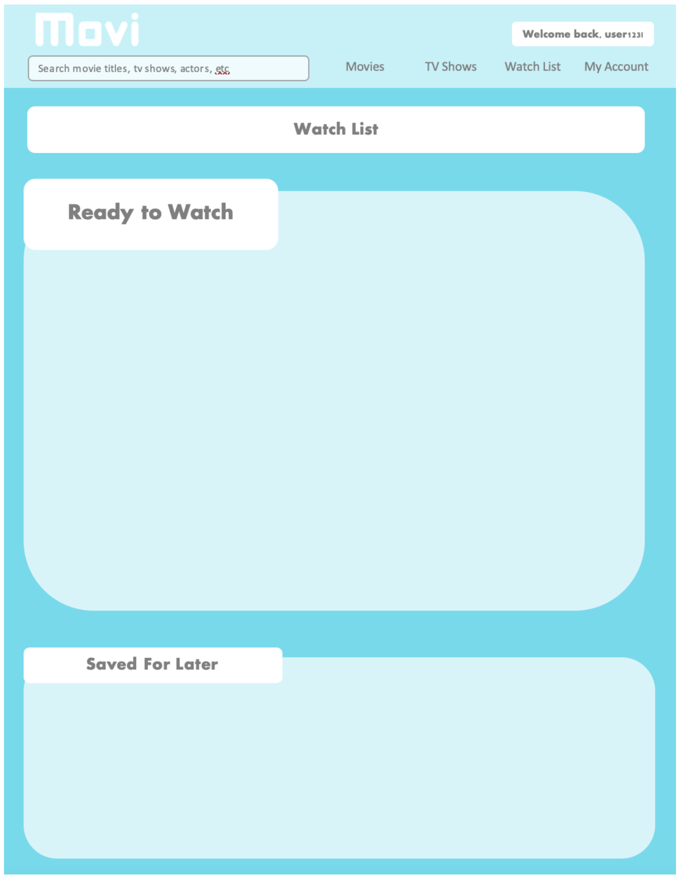
Current wireframe guide


Movi
Welcome back, user1231
Search movie titles, tv shows, actors, etc
Movies
TV Shows
Watch List
My Account
Discover
What's Hot
What's Not
Coming Soon
GOD'S COUNTRY (2022) Opens Friday
GRATITUDE REVEALED (2022) Opens Friday
CLERKS III (2022) Opens Tuesday
CASABLANCA BEATS (2022) Opens Friday


Movi
Welcome back, user1231
Search movie titles, tv shows, actors, etc
Movies
TV Shows
Watch List
My Account
My Account
My Favorites
Movi Stats
Recommended


Movi
Welcome back, user1231
Search movie titles, tv shows, actors, etc
Movies
TV Shows
Watch List
My Account
Movies
Thrillers
Horror
Action
Wildcard1


Movi
Welcome back, user1231
Search movie titles, tv shows, actors, etc
Movies
TV Shows
Watch List
My Account
TV Shows
Sit-Coms
Documentaries
Wildcard1


Movi
Welcome back, user123!
Search movie titles, tv shows, actors, etc.
Movies
TV Shows
Watch List
My Account
Watch List
Ready to Watch
Saved For Later