# Movies Storefront Documentation

## Summary

The app\_movies cartridge features a movie trailer search functionality on a SFCC website using IMDb and YouTube APIs.

## Features

1. **Movie trailer search**: The cartridge consists of a website where users can search for movie trailers.
2. **Advanced search:** Advanced search includes smart search features to filter down the search results by language, genre and release year. Language and genre options are configurable via custom preferences.
3. **Predictive search:** When the number of characters entered on the search field has reached 3, a search is triggered to show results corresponding to the entered search term.
4. **Top 25 IMDb Movies:** A carousel is shown below the search section where the top 25 IMDb movies are shown.
5. **Request Trailer Form:** A movie trailer request form is provided so that users can request for trailers that are unavailable on search results.
6. **Social Share:** Social share option is provided for trailer video which enables users to share to Facebook and Twitter.
7. **Client and Server side validations:** Client and server side basic validations are integrated for security.
8. **Accessibility:** Basic accessibility features implemented and verified using chrome lighthouse extension.

## Key Decisions

* *YouTube API is called when Play Trailer link is clicked*:

On search of a movie title, the IMDb search movie API is fired to get search results corresponding a keyword. For getting the embedHtml to render a YouTube video on the website, the YouTube API is used. The input to this API is a video id which is not received as part of the IMDb search movie API or IMDb Advanced search API. To get this video id, the IMDb YouTubeTrailer API is fired using the imdbID, as input, which is received from IMDb search movie API initially. If all these API were being called on a single backend call, it could have constrained the number of search results to a maximum of 5 and would drastically increase the response time of the search API. Thus, the YouTube API had to be separated out to be fired only on link is clicked.

* *Custom object is used to cache / store the aggregated data for each play trailer link click:* Custom objects are added as well as updated for each API call so that for next play trailer link click, third party API call could be avoided, and data is taken from the custom object to show on the UI.
* *For the advanced search, filter options are made configurable via the custom preferences so that new language and genres could be added in future.*
* *Top 25 IMDb Movies are shown as a carousel on the website and is stored in custom cache:*

The data used to show carousel is from the IMDb Top250Movies API. The details of 250 movies are received as response out of which only the first 25 are taken. This is to reduce the load time of the page. The result is stored in a custom cache to again increase the performance and decrease initial page load time.

* *Predictive Search and most recently searched terms are show:*

This is done to provide easiness for the user to search for results.

* *Request Trailer form:*

This form is integrated in a way that on submission, the data from the form are saved to a custom object. This data can later be synced to a 3rd party system by means of a scheduled job.

## API Details

|  |  |
| --- | --- |
| **IMDb** | |
| Visit <https://imdb-api.com/> and register to get an API KEY. | |
| API details | <https://imdb-api.com/api>  <https://imdb-api.com/api/#SearchMovie-header>  <https://imdb-api.com/api/#AdvancedSearch-header>  <https://imdb-api.com/api/#YouTubeTrailer-header>  <https://imdb-api.com/api/#Top250Movies-header> |
| Swagger | <https://imdb-api.com/swagger/index.html> |

|  |  |
| --- | --- |
| **YouTube** | |
| Register on or login to google developer console: https://console.cloud.google.com/apis/dashboard. After registering, create a project and generate API KEY. | |
| Documentation | <https://developers.google.com/youtube/v3/docs> |
| API details | <https://developers.google.com/youtube/v3/docs/videos/list> |

## Implementation Guide

1. Import the cartridge into UX studio and associate them with a Server Connection.
2. Add the cartridge app\_movies to site path, in front of app\_storefront\_base
3. Import metadata/Site-template/meta/system-objecttype-extensions.xml and metadata/Site-template/meta/ custom-objecttype-definitions.xml
4. Import service

## Configurations

After importing metadata, navigate to BM > Administration > Global Preferences > Custom Preferences > **MovieSearchSetting**

|  |  |
| --- | --- |
| Preference Name | Purpose |
| IMDB API Key | IMDb API key |
| Youtube API Key | YouTube API key |
| Maximum Results | Number of response objects to be updated in custom objects |
| Supported Languages | Configure search languages |
| Supported Genres | Configure search genres |

## Testing

After completing the implementation guide and configurations above, navigate to SearchMovies-Show pipeline path and search for a movie name.