COMP.SE.110 Software Design

Group Assignment Design Document

Link to the prototype

GROUP JMSK
Jani Ilkka, jani.ilkka@tuni.fi Markus Siitonen, markus.siitonen@tuni.fi Sviatoslav Vasev, sviatoslav.vasev@tuni.fi Kian Moloney, kian.moloney@tuni.fi

Document version history	
Document created	14.9.2024 (Jani Ilkka)
Added movie functionality-related documentation and a diagram	19.9.2024 (Jani Ilkka)
Updated Movies-table and the diagram	29.9.2024 (Markus Siitonen)
Numerous updates for mid-term	23.-24.2024 (Kian Moloney, Markus Siitonen, Jani Ilkka)

Table of Contents

Introduction

Application Idea

Our application is called Movie Night Planner.

Movie Night Planner allows users to select a movie to watch based on search criteria. The application then tells which streaming service(s) has/have the movie in their catalog.

BONUS: As a potential addition to the aforementioned functionality, we have an idea about using a third-party API to get recommendations as to which snacks go with the theme of the movie.

Schedule and Division of Work

Course week	Tasks	Deadline	Responsible
1	Post a group formation message.	27.8.	Jani Ilkka, Markus Siitonen
2	Choose application idea.	10.9.	Whole team
3	Pivot the application idea.	18.9.	Whole team
4	First Prototype Views	20.9.	Markus Siitonen, Sviatoslav Vasev, Kian Moloney
5	JavaFX application creation, TMDB-related Java code, including relevant documentation.	20.9.	Jani Ilkka
6	Additional TMDB java code added. Documentation updated.	25.9.	Jani Ilkka
7	TA Meeting	1.10.	Whole team

Due to the hectic nature of the project implementation, after the first TA-meeting, this table is no longer being updated. You can find the final “division of work” table at the end of this document.-

Central Features

Movie Night Planner recommends movies to watch.

We have included 10 services, which can be used to stream or by a movie, or simply to watch it based on a subscription. Movies can be filtered according to individual services.

In addition to services, movie suggestions can be narrowed down according to genres, spoken languages, and/or subtitle languages.

Preferences can be used to store the aforementioned filters.

Graphs and diagrams provide the user with information about, e.g., favourite genres and watch history.

User Guide

After the application has started, the user is taken first to the search view.

The search view has three main sections. The one on the top is the filter area. The main area is taken by a tab view. The first tab displays two horizontally scrollable lists, containing the currently most popular and top-rated movies. The other tab shows the search results based on the selection in the filter area.

If the default lists already show an interesting movie, click it.

If you wish to use the filters:

1. Select the checkboxes that correspond to the streaming services you wish to use.
2. Choose one or multiple genres from the Genre menu.
3. Choose spoken languages from the Spoken language menu.
4. Choose subtitles from the Subtitle language menu.
5. Click Show results.
6. Select a movie by clicking it.

Rating a movie

1. Navigate to the Movie Details view.
2. Choose Rate.
3. Select an appropriate number of stars.
4. Choose Rate or Cancel, if you do not wish to rate the movie.

Saving preferences

1. Click the profile picture.
2. Choose the streaming services you want to be selected by default.
3. Select the genres you want to be selected by default.
4. Select the languages you want to use by default.
5. Choose Save preferences.

Check statistics

1. Click the profile picture.
2. See statistical information about your watch history in a graphical form.

APIs Utilized

API	Functionality
The Movie Database (Getting Started (themoviedb.org)	Provides movie and tv-show data.
Movie of the Night (Shows - Streaming Availability API Reference (movieofthenight.com))	Lists streaming services that can be used to watch the movie/tv-show selected in the first phase. We use this to get language-area-specific streaming service and subtitle-information.
BONUS: OpenAI API	Provides snack and recipe information.

Classes, Interfaces, and Their Responsibilities

Below you'll find a general level description of the technical implementation.

A more detailed documentation is available in the project Javadoc files.

Movies

Movies (TMDB API)		
Views	SearchView	Start view, initially showing a list of most popular movies in the database. Enables searching for movies based on search criteria.
	MovieDetailsView	Displays detailed data about a selected movie.
Controllers	MovieDataController	Houses Application logic for all movie-related functionalities.
	MovieDetailsController	Handles all things related to the movie details page.
	ImageController	Handles all image-related features.
	RateViewController	Initializes rate view and handles rate button press with appState.
Models	Genre	Represents movie genre.
	GenresResponse	Response data from the "get all genres" API. (https://api.themoviedb.org/3/genre/movie/list)
	Movie	Non-detailed data about a single movie. Used by PopularMoviesResponse and MovieSearchByTitleRespose.
	MovieDetails	Detailed data about a single movie.
	MovieDetailQueryResponse	Response data for a data request about a single movie. (https://api.themoviedb.org/3/movie/{movie_id})
	Cast	Detailed data about a member of the cast.
	Crew	Detailed data about a crew member of the movie (writers etc.)
	Credits	Has both the crew and credits of the movie.

	PopularMoviesResponse	Response data for the API returning data about the currently most popular movies. (https://api.themoviedb.org/3/movie/popular)
	~~ProductionCompany~~	~~Represents production company information. (Inside MovieDetails.)~~
	~~ProductionCountry~~	~~Represents production country information. (Inside MovieDetails.)~~
	SpokenLanguage	Represents spoken language information. (Inside MovieDetails.)
	MovieSearchByTitleResponse	Response data for movie title-based search. (https://api.themoviedb.org/3/search/movie)
	StreamingProvider	Holds the information of a streaming provider
	StreamingResponse	Response data for a List of Streaming Providers (https://api.themoviedb.org/3/watch/providers/movie?language=en-US&watch_region=FI)

MovieDataController

Method	**Functionality**
getPopularMovies()	Returns a list of currently most popular movies.
getMovieDetails(movieID)	Returns detailed movie data.
searchMoviesByTitle(title)	Returns a movie list based on a title search string.
getAllGenres()	Returns a list of all available genres.

MovieDetailsController

Method	**Functionality**
setMovie()	Loads all of the movie details into the view.
clearMovieDetails()	Empties labels, images and elements.
setProviderImages(movie)	Sets the two provider images. Will be changed to use TilePane later.
setCast(movie)	Sets the relevant cast info including profile images.
handleRateButtonAction()	Passes the appState to the rate view controller and sets the scene.

ImageController

Method	**Functionality**
loadImageIntoView(path, fxId, width, scene)	Loads images from the TMDB API in to view based on fx:id and crops the image if necessary.
loadLogosIntoMovieLabel(path, tilepane)	Loads provider logos into TilePane in the movie label.
loadPosterIntoMovieLabel(path, imageview)	Loads movie poster into the movie label.

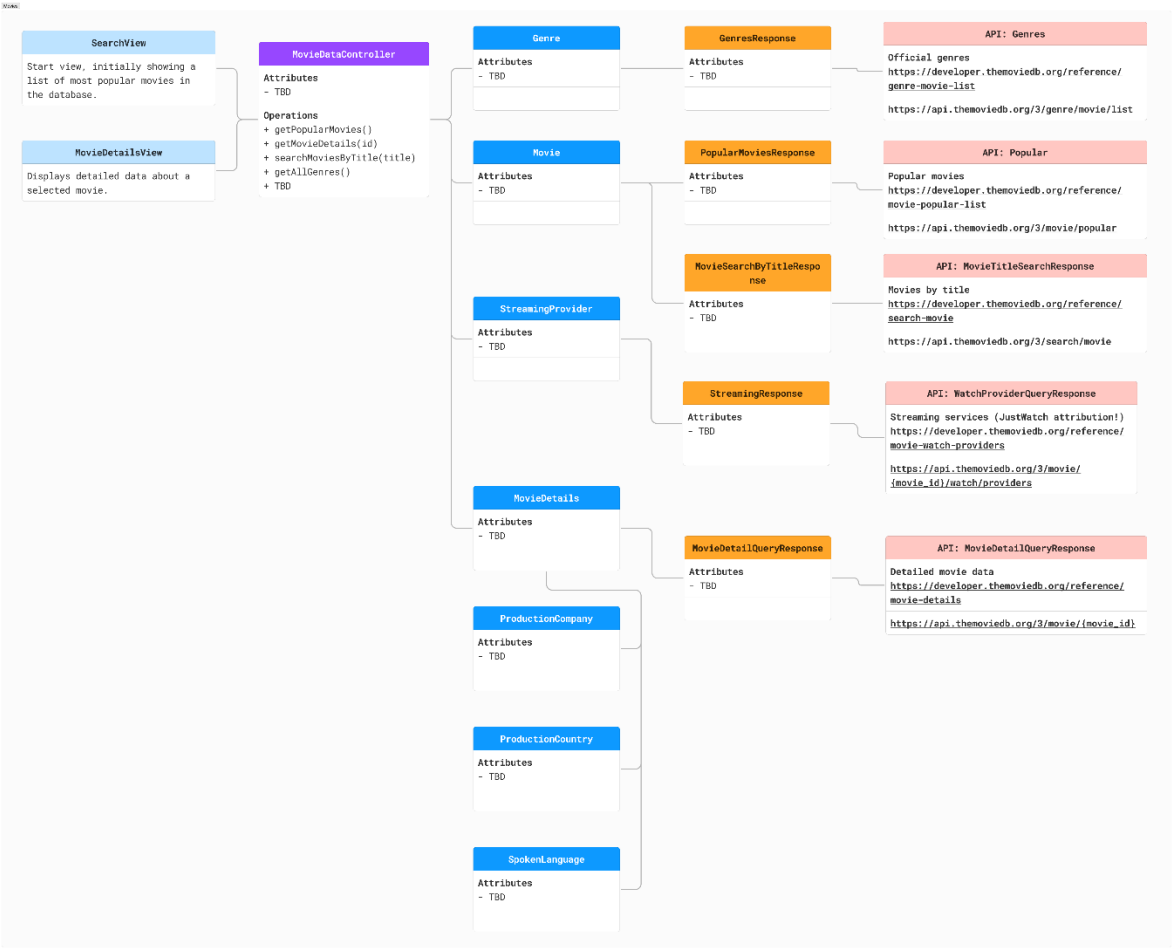


Figure 1 Movie functionality information flow

Subtitles

Subtitles (Movie of The Night API)		
Controllers	ShowDataController	Application logic for all subtitle-related features.
Models	SubtitleService	Represents a json object that contains the language and subtitle information, in addition to the rental/purchasing/subscription services information.

ShowDataController

Method	Functionality

areFinnishSubtitlesAvailableAtAll (int movieId)	Returns true, if Finnish subtitles are available for any streaming service for the movie.
areFinnishSubtitlesAvailableForMovieInService(String streamingServiceId, int movieId)	Returns true, if Finnish subtitles are available for the chosen streaming service for the movie.
getServicesWithFinnishSubtitles(int movieId)	Returns a list of streaming services that offer Finnish subtitles for the selected movie.
IsFinnishAreaSupported(int movieID)	Returns true, if the movie is available on the “fi”-area.
CheckRequiredSubtitlesForMovieInService(int movieId, String streamingServiceName, List<String> languageList)	Checks if the selected subtitles are provided by the selected service for a movie.

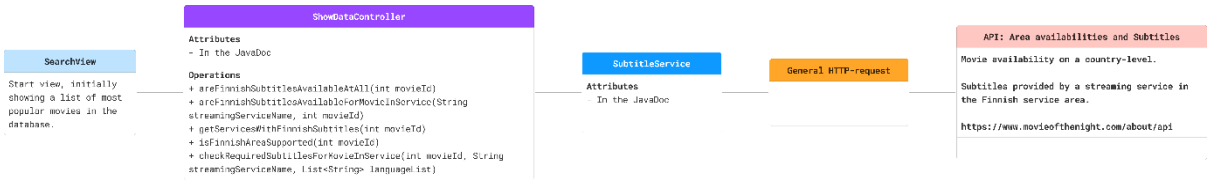


Figure 2 Subtitle functionality information flow

Other

Application State and User Data		
Views	ProfileView	Displays user-specific data, including view history and genres of movies rated by the user. Previous viewing data and ratings are used to display a collection of charts and figures.
Controllers	AppState	Application logic for user information, search history, movie ratings and application state management.
	ProfileViewController	Application logic for the ProfileView.
	Constants	Provides access to constant-based information, such as language codes.
	FileDataController	Handles reading data from the disk and writing data to the disk.
	GSONTools	Utility class for JSON-operations.
	HTTPTools	Utility class for HTTP-operations.
	LanguageCodes	Utility class to translate between two language code types.
Models	ErrorModel	Utility class to handle TMDB errors.

AppState

Method	Functionality
getSearchHistory	Returns true, if Finnish subtitles are available for any streaming service for the movie.
addSearchedMovie(Movie movieSearch)	Returns the user's movie search history.
saveMovieRating(int movieId, int rating)	Adds a movie to the user's search history.
getMovieRating(int movieId)	Gets the rating for a movie.
getMovieRatings()	Gets all movie ratings.

ProfileViewController

Method	Functionality
setStreamingProviders()	Sets the streaming providers by iterating over the streaming provider nodes and assigning unique IDs from the provider list.
setStreamingProviderLogos()	Sets the streaming provider logos in the search view by fetching and displaying them in the UI.

Constants

Method	Functionality
createLanguageList()	Creates a list of language pairs containing language codes and their corresponding names.
getProviderIds()	Retrieves a list of streaming provider IDs by using reflection to access the static fields in the {@code PROVIDERS} class.
getProvidersString()	Encodes the provider IDs into a URL-friendly string, where IDs are concatenated and separated by the '|' character.
getProvidersString(List<Integer> list)	Encodes a given list of provider IDs into a URL-friendly string, where IDs are concatenated and separated by the '|' character.
getLanguages()	Retrieves the list of supported languages.
getPopularMoviesUrl()	Generates the URL for retrieving popular movies from TMDb, with the list of providers encoded as a URL parameter.
getTopRatedMoviesUrl()	Generates the URL for retrieving top-rated movies from TMDb, sorted by vote average and including only providers supported by the app.
getGenresUrl()	Retrieves the URL for fetching movie genres from TMDb.
getFilteredUrl(List<Integer> list)	Generates a filtered URL for discovering movies from TMDb based on a specific list of providers.

FileDataController

Method	Functionality
readStateFromFile()	Reads app state data from a JSON file.
writeStateToFile(AppState appState)	Writes state data to a JSON file.

GSONTools

Method	Functionality
convertJSONToObjects(String JSONString, Class<?> targetClass)	Converts a JSON string to an object of the specified class.
convertJSONToObjects(String JSONString, Type type)	Converts a JSON string to an object of the specified type.

HTTPTools

Method	Functionality
makeGenericHttpRequest(String url)	Handles HTTP requests and responses asynchronously.
makeTMOTNHttpRequest(String url)	Handles HTTP requests and responses asynchronously. Only works with The Movie of the Night due to the mandatory headers.

LanguageCodes

Method	Functionality
getLanguageCode(String key)	Gets the language code for the specified key.

BONUS: Snacks (TBD)

Snacks (ChatGPT API)		
Views	RecipeView	Displays a recipe for a movie-theme-appropriate snack.
Controllers	RecipeDataController	Application logic for all recipe-related features.
Models	GenAIPrompt	Message to the generative AI service.
	GenAIResponse	Response from the generative AI service (the recipe).

Architecture

Architecture Patterns

The selected architecture pattern is MVC.

We spent considerable time researching[iiiiiiiv] and discussing this choice. As JavaFX does not dictate using a single pattern, it boils down to what suits the application best and what are the preferences of the development team.

The application itself is rather simple. A classic, clear division between the view, application logic, and the model was deemed sufficient. We opted for a version where the models are as lightweight as possible. In fact, they only represent data with appropriate getters and setters.

The length of previous experience and amount of knowledge of using architecture patterns varied greatly within the development team – starting with no experience and ending with professional work experience with multiple patterns. This impacted the selection process in favor of a simple pattern.

Design Patterns

Singleton

All helper-features, such as HTTP requests and GSON-operations were implemented as objects adhering to the singleton design pattern. Such features do not require creating multiple instances.

Application state is another object that uses the singleton approach. The state object is a property of SceneController (more about this below). In order to persist the state, it is saved on the disk as a JSON file each time the application is shut down. The state information is automatically loaded each time the application starts.

Navigation

When we were thinking about how to solve the flow of information from one view to another and how to store application state data, we decided to implement a top-level navigation structure to tackle several of these questions on one go.

In addition to aforementioned issues, we analyzed what happens from the JavaFX application perspective, when a user navigates from one view to another. In practice, we thought about optimizing the application functioning under runtime based on processing power or memory usage. We decided to save processing power and take the penalty in the form of reserved memory.

SceneController creates a navigation stack that keeps already created views alive in the memory. Therefore, they are not recreated every time a user navigates to a view. This also enables us to keep state information alive in the views, saving HTTP requests.

Implementation Technologies

Below you'll find explanations for a few key technological selections. If we opted for using a third-party library, the library is listed below in a table with a source link.

Views

The user interface is implemented using JavaFX. More specifically, we opted for the XML-based approach whereby a baseline implementation of each view is created using Gluon Scene Builder. This will be later enhanced using the declarative Java approach, as needed due to view dynamics.

This makes view design quick and as effortless as possible.

ControlsFX

ControlsFX is a library that has several neat looking UI controls for JavaFX.

The rationale for this is simply to save development time. Also, the controls look really nice visually.

We use a rating control and a multi-select drop-down menu from this library.

HTTP Client

We use Apache HttpClient (version 5). (Apache HttpComponents – HttpClient Quick Start)-

There are centralized, but separate clients for different services. The reason is that each client requires different headers, etc. We thought about creating a single general client, that would be configured per request, but we came to a conclusion that it would be absolutely ludicrous. Thus, a single client per service, preconfigured with appropriate settings.

The rationale for choosing this client is based on multiple criteria. The popularity of a library is always a good thing when doing technology selection. Additionally, it is flexible and comes with several advanced features. We based our selection rationale also on this comparison: Which Java HTTP client should I use in 2024?

AI Disclosure

Various AI services were used with several tasks during the project, as detailed below.

Idea

We used several prompts to create dozens of application ideas. The idea that was ultimately chosen was from ChatGPT. The chosen prompt-idea combination is:

"Suggest ideas for a programming group project in an university software design class. The application should use and combine data from at least two open REST application programming interfaces."

Answer from ChatGPT:

Movie Night Planner

- **Description**: A collaborative movie recommendation platform where users can select a movie and receive recommendations for streaming services and snacks.
- **APIs to Use**:
    - **The Movie Database (TMDb) API**: For fetching movie information and ratings.
    - **Utelly API**: For determining which streaming platforms have the selected movie.

- **Spoonacular API**: For suggesting themed snacks and drinks based on the movie genre.

It was later revealed that we do not need Utelly API at all, but we actually do need to get subtitle information from somewhere. Thus, we dropped Utelly and started to use the Movie of the Night API.

Code

All initial Java model class files were created by Copilot. For each API, we requested Copilot to create a Java class that corresponds to the JSON data that each individual API returns. The Author-tag in such Java classes include "Copilot", in addition to at least one human name.

The draft level Java code was refined by team members. This means adjusting property visibility settings and, e.g., refactoring the Movie-class out of two API responses into a class of its own.

Graphic Design Resources

All app logos, background images etc. Were originally created using DALL-E.

Javadoc

The majority of all comments in the code were created with the help of AI (Copilot and/or ChatGPT).

External Libraries

Library	Source
Apache HttpClient	Apache HttpComponents – HttpClient Overview
ControlsFX	ControlsFX | ControlsFX
GSON	Google - GSON

Division of Work

Below you'll find a table showing the division of work to a rather detailed level. This is based on the GitLab issues and Teams disussions. We exported the issues as a CVS file and proceeded from there. Some really small individual issues may have been combined to form a larger, more sensible item.

Task	Responsible
temp	n.n.

Self-evaluation

This section is divided into two separate sections, one pertaining to the design and another to the technical aspects.

Design

We were initially a bit worried about the potential limiting factors of JavaFX when compared with what we had done with Figma. As is evident from the final UI, our fears did not come true. With enough patience and tinkering, JavaFX can be used to create visually impressive user interfaces with, e.g., transparency and blur.

We do not expect this situation to change in any way.

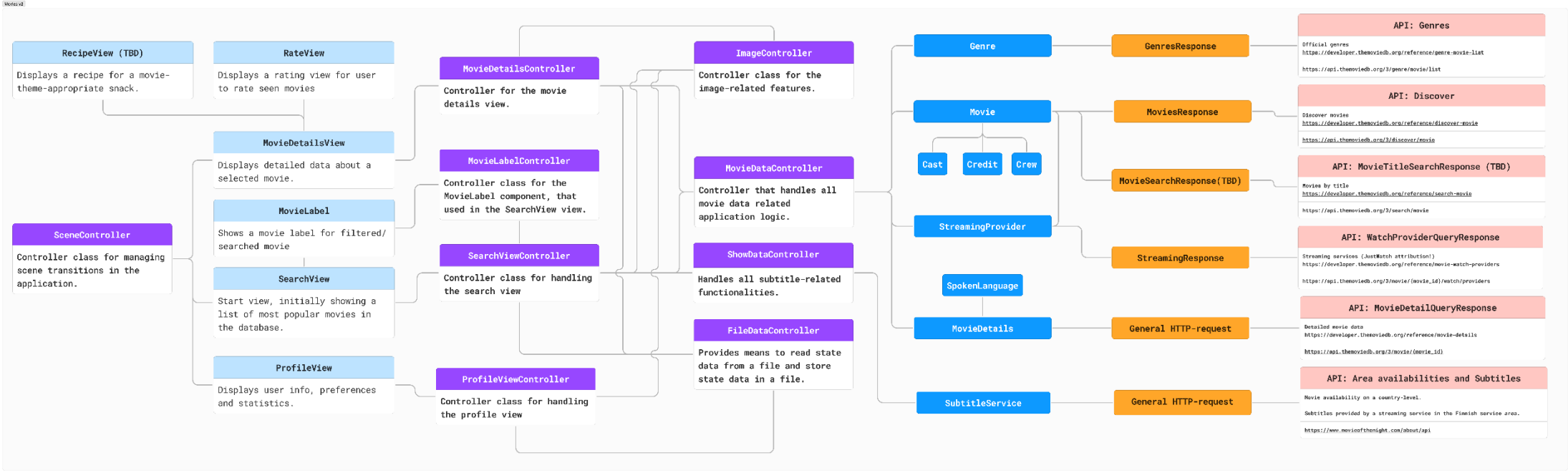
Technical aspects

After the programming 3 course, three quarters of our team are seasoned JavaFX veterans. We had a pretty good idea as to what is possible and how things should be implemented.

The largest changes were related to the APIs we thought we would be using. For streaming services and subtitles we dropped Utelly and went with Movie of the Night. Regarding snacks or food, we noticed that Spoonacular (obviously) has U.S.-specific products. A generative AI-based approach was deemed better.

Reuse of components

We were originally planning to divide the UI into separate components and reuse those in multiple places. On a high level, this was deemed unfeasible. What we noticed in practice was that while the reuse of FXML was effortless, when connecting controllers and trying to reference UI controls in an imported view, we ran across fatal problems. Obviously, on a low level, there is a lot of reuse going on. For example, both movie lists repeat a single movie-specific design.


Movies v3
RecipeView (TBD)
Displays a recipe for a movie-theme-appropriate snack.
RateView
Displays a rating view for user to rate seen movies
MovieDetailsView
Displays detailed data about a selected movie.
MovieLabel
Shows a movie label for filtered/ searched movie
SearchView
Start view, initially showing a list of most popular movies in the database.
ProfileView
Displays user info, preferences and statistics.
SceneController
Controller class for managing scene transitions in the application.
MovieDetailsController
Controller for the movie details view.
MovieLabelController
Controller class for the MovieLabel component, that used in the SearchView view.
SearchViewController
Controller class for handling the search view
ProfileViewController
Controller class for handling the profile view
ImageController
Controller class for the image-related features.
MovieDataController
Controller that handles all movie data related application logic.
ShowDataController
Handles all subtitle-related functionalities.
FileDataController
Provides means to read state data from a file and store state data in a file.
Genre
Movie
Cast
Credit
Crew
StreamingProvider
SpokenLanguage
MovieDetails
SubtitleService
GenresResponse
MoviesResponse
MovieSearchResponse(TBD)
StreamingResponse
General HTTP-request
General HTTP-request
API: Genres
Official genres
https://developer.themoviedb.org/reference/genre-movie-list
https://api.themoviedb.org/3/genre/movie/list
API: Discover
Discover movies
https://developer.themoviedb.org/reference/discover-movie
https://api.themoviedb.org/3/discover/movie
API: MovieTitleSearchResponse (TBD)
Movies by title
https://developer.themoviedb.org/reference/search-movie
https://api.themoviedb.org/3/search/movie
API: WatchProviderQueryResponse
Streaming services (JustWatch attribution!)
https://developer.themoviedb.org/reference/movie-watch-providers
https://api.themoviedb.org/3/movie/{movie_id}/watch/providers
API: MovieDetailQueryResponse
Detailed movie data
https://developer.themoviedb.org/reference/movie-details
https://api.themoviedb.org/3/movie/{movie_id}
API: Area availabilities and Subtitles
Movie availability on a country-level.
Subtitles provided by a streaming service in the Finnish service area.
https://www.movieofthenight.com/about/api

References

[i] Design pattern to use : r/JavaFX (reddit.com)
[ii] Implementing JavaFX Best Practices | JavaFX 2 Tutorials and Documentation (oracle.com)
[iii] Best practices for JavaFX architecture and patterns? - Oracle Forums
[iv] Implementing MVC in JavaFX - PragmaticCoding