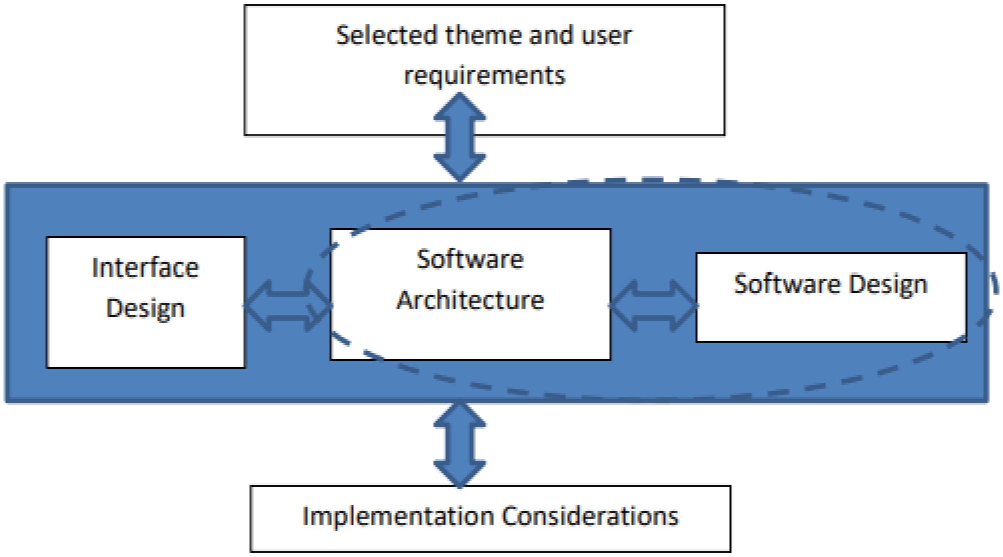
1.0 Introduction

"This report is the second step towards developing your design. The figure below shows the scope of this intermediate report in relation to the D1 and the final report that needs to be delivered in week 10."


Selected theme and user requirements
Interface Design
Software Architecture
Software Design
Implementation Considerations

The following deliverable is separated into sections to respond to two parts. This report will identify and justify the project's chosen software architecture and the team's initial software design.

Group members of Jesse & Co.

Jesse Merhi, z5312498

Jay Patel, z5309776

Jack Whaling,

Azren Snow,

Weiting Han, z521058

1.1 Table of Contents

2.0 Software Architecture

The following section is a response to Part 1, Software Architecture.

“Given the current user stories of what the team has decided to do, what would be the resulting software architecture?”

Each subsection is a response to correlating elements listed in the Deliverable 2 specification. These elements follow as:

1. What external data sources will your system be accessing?

2. Software components: the selected Web stack showing major software components that comprise your solution. These will include both components that need to be developed and third-party components (e.g. web browser).

3. Relating choices to components: decide which language should be used for which component of the software architecture. This will be largely determined by the Web stack but at the same time you can make variations.

4. The choice of a platform: decide on machine or machines requirements (Linux, Windows etc.) for the final system Selected theme and user requirements Software Design Interface Design Implementation Considerations Software Architecture.

5. Make a summary of the key benefits/achievements of your architectural choices.

2.1 External Data Sources

This section is a response to question 1, “What external data sources will your system be accessing?”

Our Web-Application provides data on 4 major types of online content, those being Movies, TV shows, Music and podcasts. As a result of this it is necessary to use several different APIs in order to obtain this data on demand.

1. TMDB API:

   This API is from the website “The Movie Database” and essentially allows the access of expansive data in relation to (nearly) all movies and tv shows that have been released. This API not only provides information about content, such as titles, ratings, descriptions and actors, but also provides support for places to watch the content from all locations around the world. The best part is the API is completely free and has very detailed documentation. This API will provide us with the majority of the data we require for movies and tv shows.

2. Spotify API

   The Spotify API is an extensive data store that provides access to all information in regard to music that is accessible on Spotify. This API provides access to artist names, song names, hits and more. There are also several podcasts on Spotify which also have their information provided via the use of the API which allows quite easy access for that specific information. This API is also completely free and will provide a lot of the information for the music tracks and podcasts on our web application.

3. Apple Music API

   Finally, the last API is the Apple music API. Once again this is a professional API that provides information on music that is available to be streamed on the Apple Music app. Whilst this API is not as powerful as the Spotify API, only containing music and no podcast information. This will still help in providing us plenty of data so that it becomes quite easy for us to give several streaming options, especially when music streaming platforms are quite limited right now. This API is also free and as mentioned before, will help support the music collection on our website.

2.2 Software Components

This section is a response in regards to our software components. It is related to element 2, “the selected Web stack showing major software components that comprise your solution. These will include both components that need to be developed and third-party components (e.g. web browser).”

When discussing the web stack that we will be utilising, the stack is relatively simple. Most of the developers working on the project are working on windows machines. So during the development the operating system of our stack will most likely be on consumer windows 10 versions. That being said, when the web-app is actually launched, the operating system will most likely be a basic form of Linux using one of the Amazon or Google cloud services machines to allow for almost 24/7 uptime.

When it comes to the web-app itself, we will be using a ReactJS frontend alongside a Python Flask backend to process information. The main reason behind using React is because most of the team has little to no experience in front end programming, and after doing research it seemed that React is one of the easier and more beginner friendly languages to create website frontends. As for using Python and Flask, this is mainly due to the extensive experience that the team has due to completing the COMP1531 (backend programming) course at UNSW, which uses Python and Flask.

There will also be a requirement to use a database, primarily for user information, but also perhaps to store the details about movies, tv shows, music or podcasts. As groups are encouraged to practise social distancing, the team decided to work with Google’s online database system called ‘Firebase’. This should provide all members easy access to all the data we store and using the free-tier of Firebase it also comes at no cost. Whilst it would be nice to store content details as well, this is lower priority as all this would realistically provide is faster search speeds. So at the bare minimum the database would most likely just be used to store user information.

2.3 Relating Choices to Components

This section is a response to how our team will relate development and design choices to components. This section is related to element 3, “decide which language should be used for which component of the software architecture. This will be largely determined by the Web stack but at the same time you can make variations.”

To ensure the components used in our software architecture contain a high level of cohesion and low need of coupling our team has selected a particular architecture pattern, modules, and components.

The languages primarily used for implementation will be ReactJS and Python Flask. As ReactJS is primarily used to build user interfaces, it will be used to provide a stable and user-friendly frontend. Since it will not be used in the backend implementation, the use of ReactJS requires a lower level of coupling and high cohesion as it is able to communicate with various backend languages. Python Flask will be used primarily in the backend implementation as information can be easily communicated between the frontend and backend through Flask. Moreover, as Python is able to communicate with various data types such as JSON, and YAML. As information from API sources are presented with a JSON format, Python provides the ability to work with this data type. Additionally, if another API or data source becomes available in the future, Python will be able to work with it. As each element of our components do not rely on each other to operate, the team’s selection of components contain a high level of cohesion and very low level of coupling.

To further employ a high level of cohesion, our team will implement a Façade software architecture pattern. As specific API’s are being employed to gain information, the need for coupling can occur if the frontend is developed to communicate exclusively with these APIs. Hence a Façade pattern will be employed to allow the communication between the APIs selected, and the ability to communicate with other APIs if necessary.

To store particular user data, the team has decided to utilize Google's online database system, 'Firebase'. The use of this system aligns with the languages used in the frontend and backend implementation as it requires a low level of coupling and is cohesive between the languages our team has selected.

2.4 Platform Choice

This section is a response in regards to the choice of a platform. This section is related to element 4, "decide on machine or machines requirements (Linux, Windows etc.) for the final system".

For the final system, the project will rely on a basic form of Linux using one of the Amazon or Google cloud services machines to allow for almost 24/7 uptime. It will likely utilise this form as it is a web-based service. As such, to allow its consistent activity, using Amazon or Google cloud service machines will provide the opportunity to ensure the application is live without the need for a dedicated server.

As the final product will be a web based application, any machine which supports an up-to-date web browser will be able to access the website. As the frontend of our project will be completed using ReactJS, newer web browsers will be able to meet the requirements to run the application. In general, versions of web browsers which existed before 2017 will be unable to support the application. The latest versions of web browsers which cannot support ReactJS includes Google 49, Firefox 50, Safari 10, IE 9, and Edge 14. As such, a machine requirement to run the application is a machine which can support a newer model of these web browsers. This includes Windows and Linux systems.

2.5 Summary of Software Architecture

This section is related to the final element of Part 1. It outlines a summary of the key benefits/achievements of your architectural choices.

We are using three different APIs to get data of movies and music. Since those APIs are independent and we need to provide convenient service to users, we choose to use a façade pattern for our software architecture.

Façade patterns can make a complex subsystem much easier to use. When the users want to get information about movies and music, we will collect those info from others' APIs, arrange them tidily and show them to the users on our website. So the users can get information from a simple interface instead of learning how to get them from different APIs.

Façade patterns carry out low coupling between the participating components. The users will not direct using the subsystems, which will protect the database and subsystems, also increasing the security of the whole system.

3.0 Initial Software Design

The following section is a response to Part 2, the Initial Software Design.

Each subsection is a response to correlating elements listed in the Deliverable 2 specification. These elements follow as:

1. The updated list of stories or use cases must first be presented.

2. Then include one sequence or interaction diagram for each use case. You can use UML sequence diagram definition: http://en.wikipedia.org/wiki/Sequence_diagram. Each box in a sequence diagram should correspond to a component in your architecture.

3.1 Updated List of Stories

The following user stories feature an updated list of stories provided in Deliverable 1.

Feature: Create account

As a: viewer

So that: I don’t have to re-enter information every time I wish to use the service

I want to: create an account to keep track of important information

Scenario: Creating an account

GIVEN I have navigated to the homepage

WHEN I click ‘Create Account’

THEN I should be prompted to enter contact information, such as email, phone, account name, password and region

WHEN I select ‘Submit’

AND I have entered suitable information (username is not taken, email exists and is not being used by another user, password and phone number meet the given requirements)

THEN An account with the details I enter will be created and I will automatically be signed in for the first time

* * *

Feature: Log In/Out of created account (see above)

As a: viewer

So that: I can manage my account across various devices, and remove access/change user when necessary

I want to: authorise and remove active devices from the account through a login portal

Scenario: Logging in then out

GIVEN I have created an account

WHEN I click ‘Sign In’

THEN I should be prompted to enter a username and a password

IF the username exists, and the password correctly matches

THEN I should be signed into the account and have a summary of the services available to me provided, otherwise, I should be prompted to re-enter correct information until a correct set is inputted.

WHEN I select ‘Log Out’

AND I click ‘Yes’ when prompted

THEN The services I have available on the device should become inaccessible on that device until I choose to log back in, and I should be navigated back to the homepage.

* * *

Feature: Organise video media from various services into a playlist to schedule viewing. In this case, a playlist can be composed of video media such as (TV shows, screenplays, movies) to create a ‘Watch List’, or audio (podcasts, music) to create a ‘Track List’. The semantics of managing the two are identical.

As a: viewer

So that: I can keep track of and schedule content I plan to consume

I want to: create a playlist of media I’m interested in, in one place to minimise confusion

Scenario: Creating and editing a Playlist

GIVEN I have an account I have signed into

AND I am on the user control panel

WHEN I click ‘Organise Playlists’

THEN I should be transferred to the page with the playlists I have compiled

WHEN I select ‘Add Show’

OR I select ‘Add Track’

THEN I should search for the media I want to add, and place it wherever in the list I see fit

* * *

Feature: Remove content from the above playlist

As a: viewer

So that: I can remove content I’ve completed or no longer hold interest in

I want: an intuitive means of removing arbitrary elements of a playlist

GIVEN I have an active account

AND I have navigated to the playlist

WHEN I select the Remove button on the movie (most likely represented by a - sign)

THEN the site will ask for confirmation, before removing the movie from the watch list display

* * *

Feature: Change privacy settings on playlist to moderate who sees the list

As a: viewer

So that: I can share media with friends, as well as conveniently discover new content

I want to: publicise watch lists that I make, as well as view other members’ playlists.

Scenario: Sharing a playlist

GIVEN I have an active account I’ve signed into

AND I have navigated to an existing playlist that belongs to me

WHEN I select ‘Playlist Settings’

THEN I should be able to set a playlist to Private (viewable only by people with a link or invite) or Public (viewable by anyone)

WHEN I select ‘Share Playlist’

THEN I should be able to select another member of the service to send the playlist to, or the service should generate a link to the created playlist

* * *

Feature: Inform user which regions media is available and how to access it, if the content is restricted by location. Content which is available in a user’s specified region will be prioritised in search results before this feature is invoked.

As a: viewer

So that: I know how to access region-locked content

I want to: be informed of which region the movie may be hosted in, so I can access content through VPN or other streaming services if possible.

Scenario: Gather region information from a movie

GIVEN I have signed into an existing account

WHEN I search for a movie not available in my region

THEN In order:

- Display content from my region

- Return results which include available outside the region if there are no results within my region, or I choose to override the original search.

WHEN A search hit is selected

THEN In the ‘Where to Find’ field, the region the film is available in will be provided.

* * *

Feature: The ability to ‘like’ media, and view these statistics

As a: viewer

So that: I can quickly assess media I haven’t consumed, as well as give approval for content I have previously enjoyed

I want to: like and rate movies that I’ve watched, as well as view these statistics on movies I search for

GIVEN I have an active account I’ve signed into

AND I am on the page of the movie I wish to rate

WHEN I select the ‘Like’ button

THEN the media I liked will be added to my Liked collection, and the total number of viewable likes for the content will increase by 1

WHEN I select the ‘Rate’ button

THEN I should be able to give the content an appropriate score, and this will influence the average rating on the movie

* * *

Feature: Search movies and shows by various filters. These filters include :

- Rating: Filter search results by rating to ensure results are age-appropriate. If audio is being searched for where ratings don't apply, you can choose to filter by explicit language.
- Popularity: in this context, popularity refers to media which receives the most searches and likes, this can help quickly find good content. This will typically be on by default.
- Genre: Filter movies such that they are in a specific genre, such as horror, action, comedy for movies/shows, classic, rap etc for audio, informative, comedy for things like podcasts. Used to narrow down content to suit specific interests for the user.
- Creator: Search for media made by a specific creator. Generally useful in locating exciting new content if you have previously enjoyed the creator's work.
- Release date: ensure returned results conform to specific time window
- Title: used to search for content with a specific title. While the other filters have more general use, this one is used to find specific content.

If no filter is selected, the search will just look for any hit which matches whatever is inputted into the asic search field. Additionally, there will be multiple search forms that can be navigated to separately for video and audio media, since there are different filters applicable to each. In any search where an input is expected, searches will not be case sensitive.

As a: viewer

So that: I can find content more readily suited to my preferences

I want to: search for content through a series of preferred filters

Scenario: Selecting content with specific filters - note each filter will have its own separate case

Title

GIVEN I am signed into an active account

AND I am in the ‘Advanced Search’ for the media type I’m after (in this case Video)

WHEN I type in the specific name of content I want to watch, for example ‘home alone’

AND I select ‘Start Search’

THEN I will be returned a list of movies whose titles contain the phrase ‘home alone’, the search is not case sensitive.

Rating

GIVEN I am signed into an active account

AND I have navigated to the ‘Advances Search’ for the media I am after (in this case Video)

WHEN I select ‘Filter by Rating’

THEN A drop down box of applicable ratings will be presented

WHEN I choose the M15+ rating

THEN the search will return a list of movies that are M15+ rating and below

Creator

-video

GIVEN I have signed into an eligible account

AND I have navigated to the advanced search

WHEN I type a Director's name e.g. 'Spielberg' into the designated input field

AND I select Find to commence the search

THEN I am returned a list of available content with only films directed by Spielberg

-audio

GIVEN I have signed into an eligible account

AND I have navigated to the advanced search

WHEN I type a Artist's name e.g. 'michael jackson' into the designated input field

AND I select Find to commence the search

THEN I am returned a list of available content created only by Michael Jackson, case sensitivity is not important

Genre

GIVEN I have signed into an eligible account

AND I am in the 'Advanced Search' for the media type I'm after (in this case, music)

WHEN I select a genre from a labeled drop down box e.g. 'Classic'

AND I select Find to commence the search

THEN I am returned a list of music classified as Classic

Popularity

GIVEN I have signed into an eligible account

AND I am in the ‘Advanced Search’ for the media type I’m after (in this case, music)

WHEN I select Sort By -> Most Popular

AND I select Start Search

THEN if I have inputted other search filters, the returned results will be sorted from most to least popular, if no other filters or search criteria had been specified, simply return the most searched-for content

* * *

Feature: Add or remove streaming services to a profile to automatically refine searches for you

As a: viewer

So that: searches consistently return media readily accessible to me

I want to: add or remove streaming services that I am currently subscribed to

GIVEN I am in my User Settings

WHEN I select My Subscriptions

THEN I can view all my current subscriptions to services for example Netflix, Stan, Disney+

WHEN I select the remove buttons next to their associated service

THEN the website will prompt me if I’m sure I want to remove the selected services

WHEN I select ‘Yes’

THEN the services will be removed from my account, and my search history will reflect this by lowering search hit priority of these services, or removing them entirely

WHEN I select the ‘Add Service’ button (symbolised by a +)

THEN I will be given a list of compatible services I can add to my account

WHEN I select one or multiple services using checkboxes

THEN Future searches will include this new service

* * *

Feature: Display information about media returned from initial search

As a: viewer

So that: I can get a brief summary of content to evaluate whether it captures my interest

I want: the search results to return a summary of all the relevant information, such as title, length, creator and synopsis

Scenario: gathering more information about content from the search result

GIVEN I have an active account

AND I have conducted a search on the media platform I’m after (in this case, video)

THEN I can see a series of movie and show posters, their titles, directors and length

WHEN I click ‘More Info’ on a movie that captures my attention

THEN I am navigated to a page which contains a more detailed abstract, including plot synopsis, main actors, language, release date and services the movie is available to watch on

* * *

Feature: record history of all content that has been visited through the website.

As a: Viewer

So that: I can conveniently re-access shows that I liked, be it to continue watching or share with others

I want to: keep a private record of all content I have accessed through the site

Scenario: Revisiting old content

GIVEN I am signed into an active account

AND I have initiated a search

WHEN I choose media through the website

THEN this activity is recorded in my history

WHEN I navigate to ‘My Profile’

AND I select ‘History’

THEN I will be given a list of content that I have engaged with on the site. This data is not accessible to anyone else.

* * *

Feature: change password

As a: viewer

So that: my private data on the website is less liable to be compromised

I want: the ability to change a password

Scenario: changing a password

GIVEN I have signed into the website

AND I have navigated to my User Profile

WHEN I select ‘Change Password’

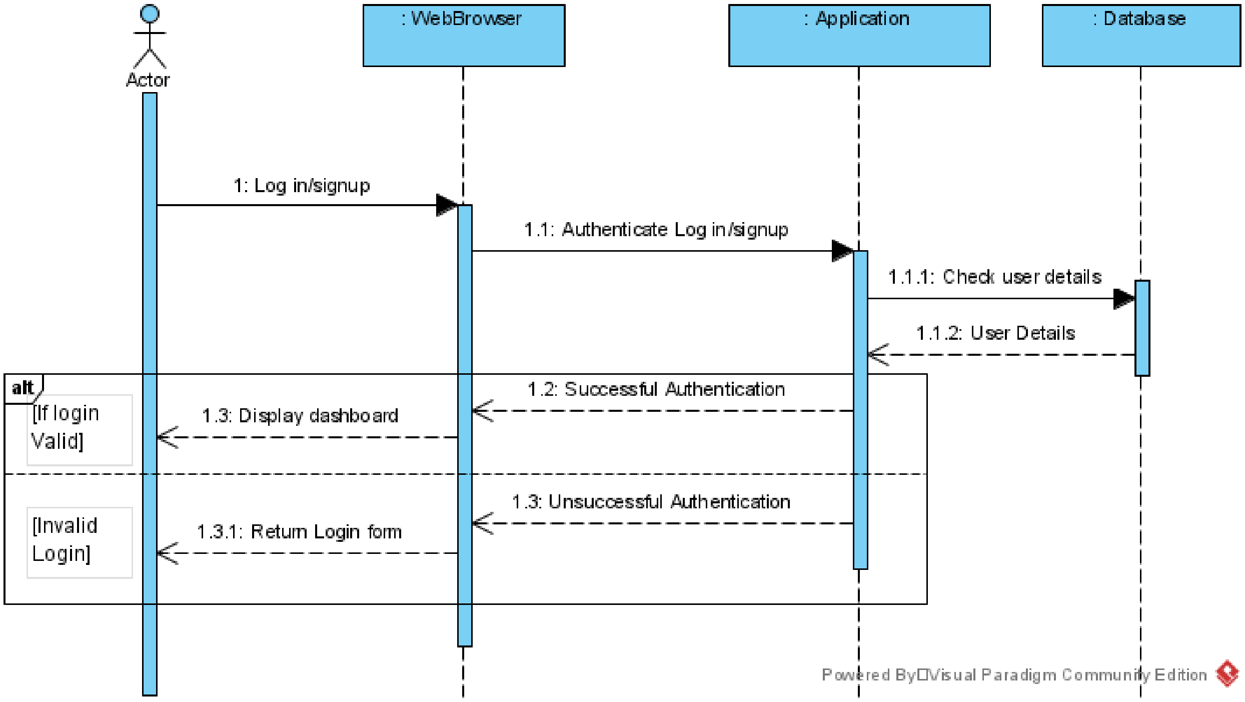
AND I can correctly input my current password when prompted

THEN I enter a new password and identical confirm password, and I can then sign into the website using these credentials, and the old password is void.

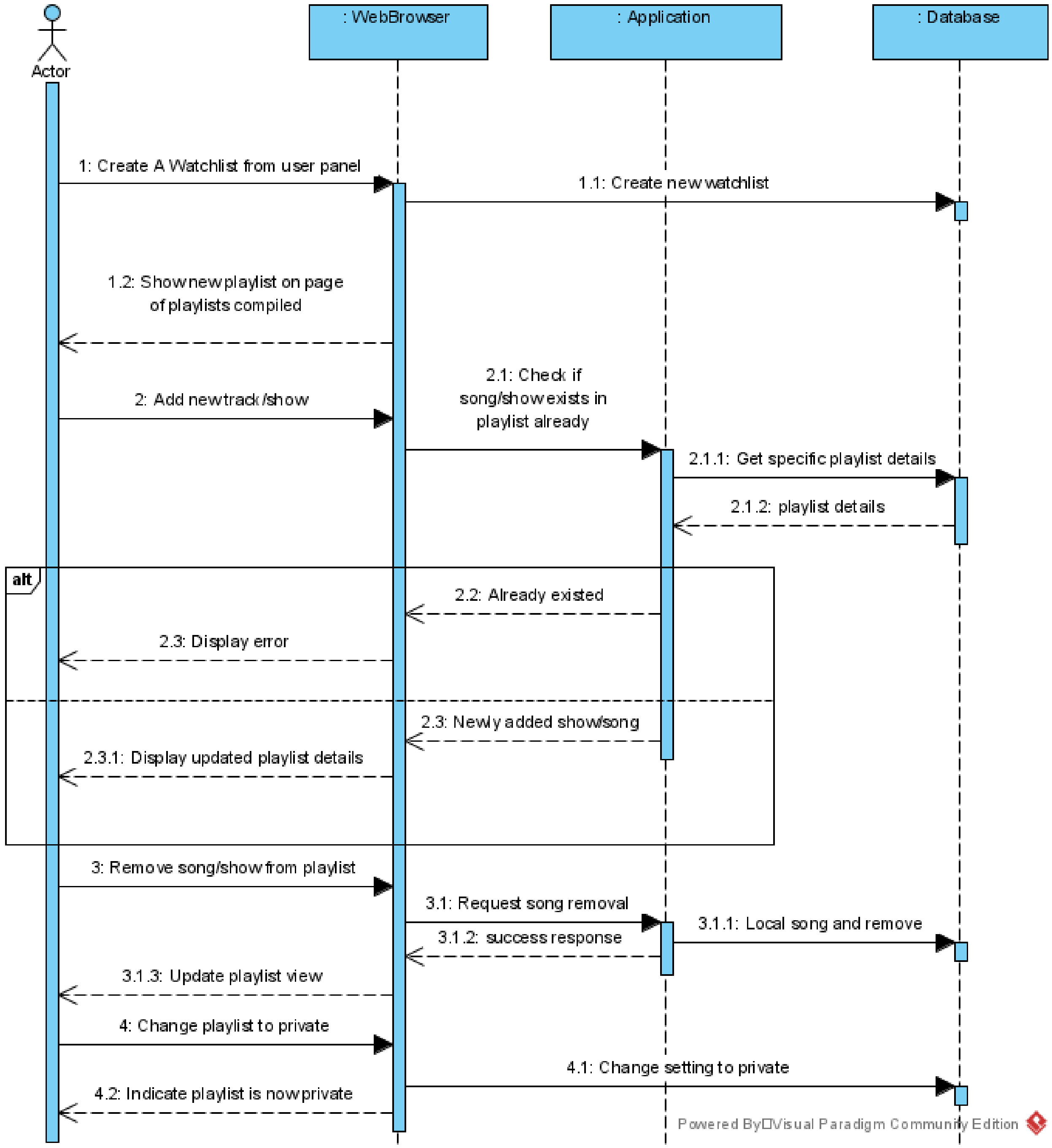
3.2 Sequence Diagram

The following Sequence Diagrams have been developed using UML Sequence Diagram definition. Each corresponds to a feature of the project’s software architecture.

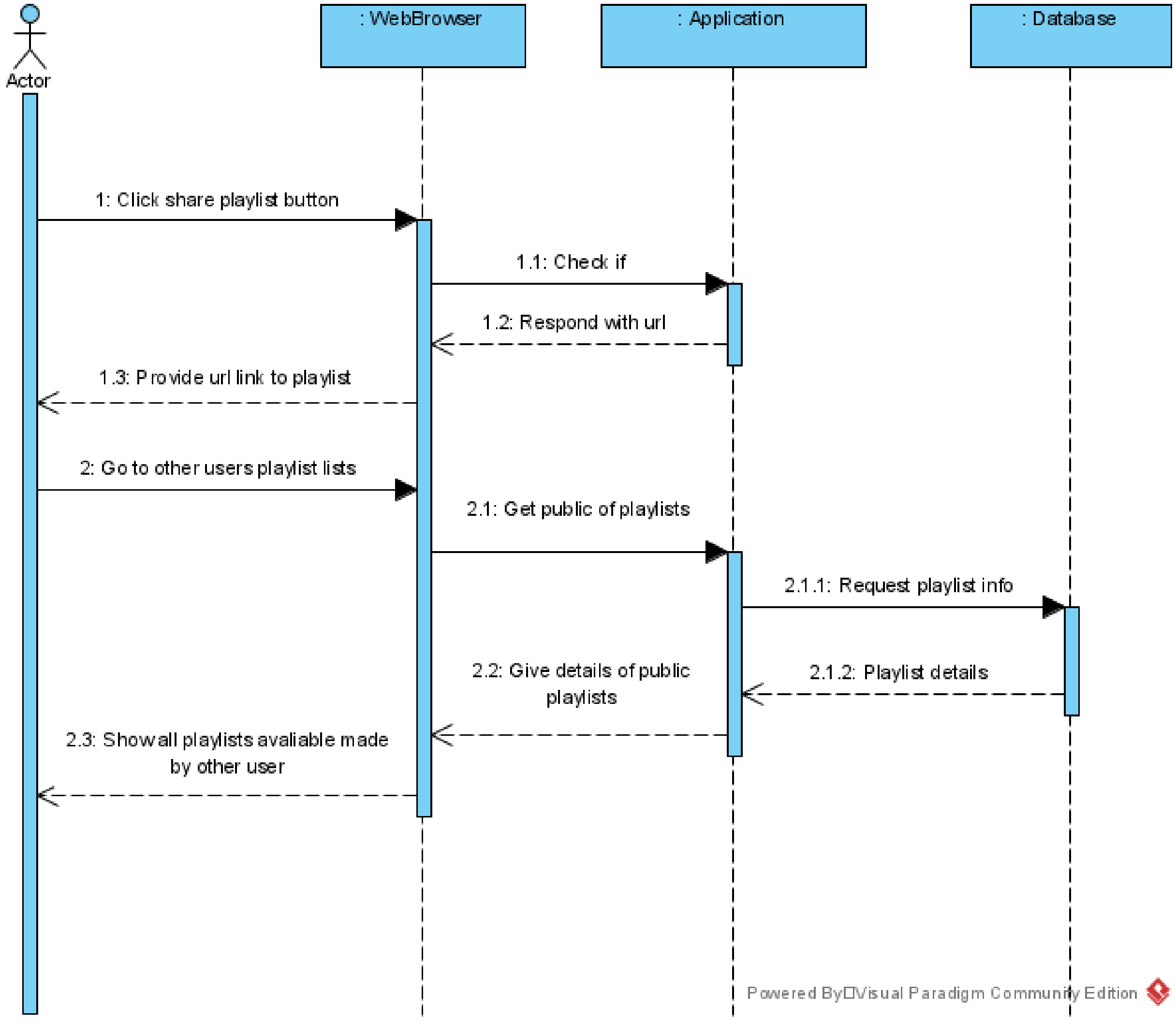
The following diagram represents the Login and Sign Up pages, as well as their interactions with the architecture.


Actor
: WebBrowser
: Application
: Database
1: Log in/signup
1.1: Authenticate Log in/signup
1.1.1: Check user details
1.1.2: User Details
alt
[If login Valid]
1.2: Successful Authentication
1.3: Display dashboard
[Invalid Login]
1.3: Unsuccessful Authentication
1.3.1: Return Login form
Powered By Visual Paradigm Community Edition

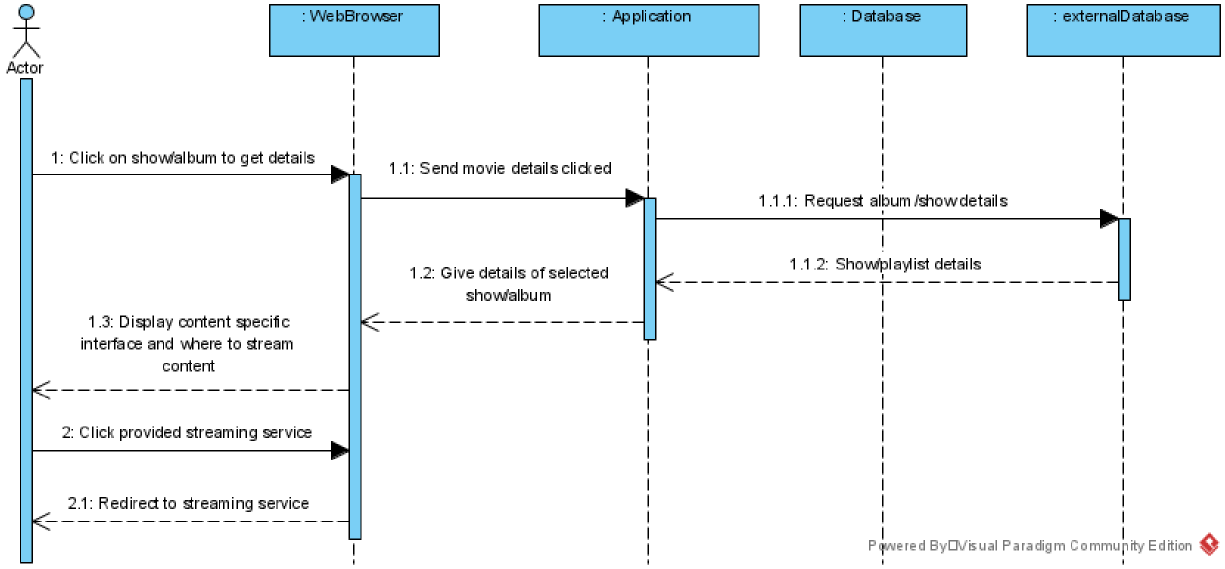
The following diagram is in relation with the Watchlist feature of the application.


Actor
: WebBrowser
: Application
: Database
1: Create A Watchlist from user panel
1.1: Create new watchlist
1.2: Show new playlist on page of playlists compiled
2: Add new track /show
2.1: Check if song/show exists in playlist already
2.1.1: Get specific playlist details
2.1.2: playlist details
alt
2.2: Already existed
2.3: Display error
2.3: Newly added show/song
2.3.1: Display updated playlist details
3: Remove song/show from playlist
3.1: Request song removal
3.1.1: Local song and remove
3.1.2: success response
3.1.3: Update playlist view
4: Change playlist to private
4.1: Change setting to private
4.2: Indicate playlist is now private
Powered By Visual Paradigm Community Edition

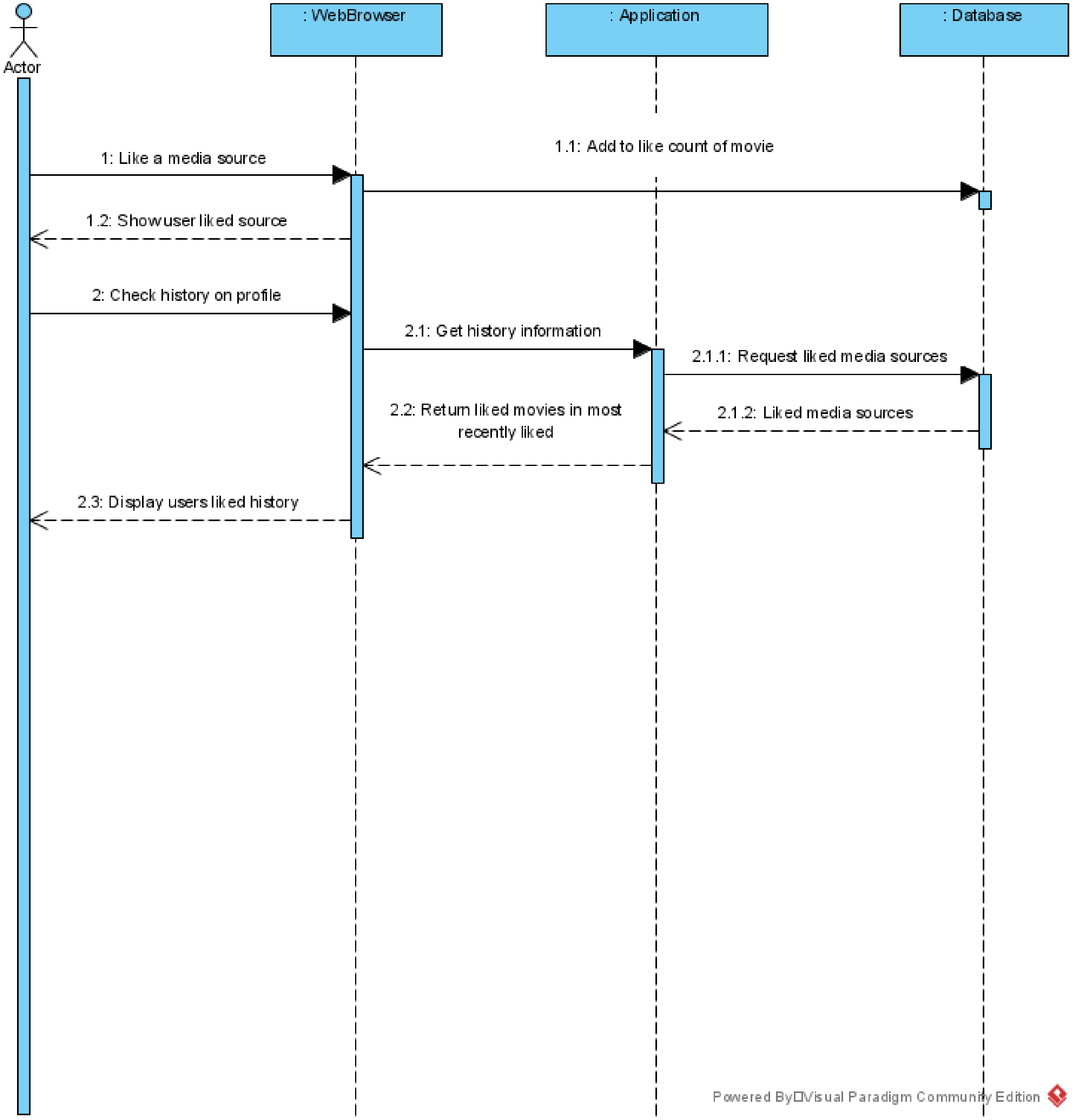
The below diagram is in relation to the application’s feature to share playlists.


: WebBrowser
: Application
: Database
Actor
1: Click share playlist button
1.1: Check if
1.2: Respond with url
1.3: Provide url link to playlist
2: Go to other users playlist lists
2.1: Get public of playlists
2.1.1: Request playlist info
2.1.2: Playlist details
2.2: Give details of public playlists
2.3: Show all playlists avaliable made by other user
Powered By□Visual Paradigm Community Edition

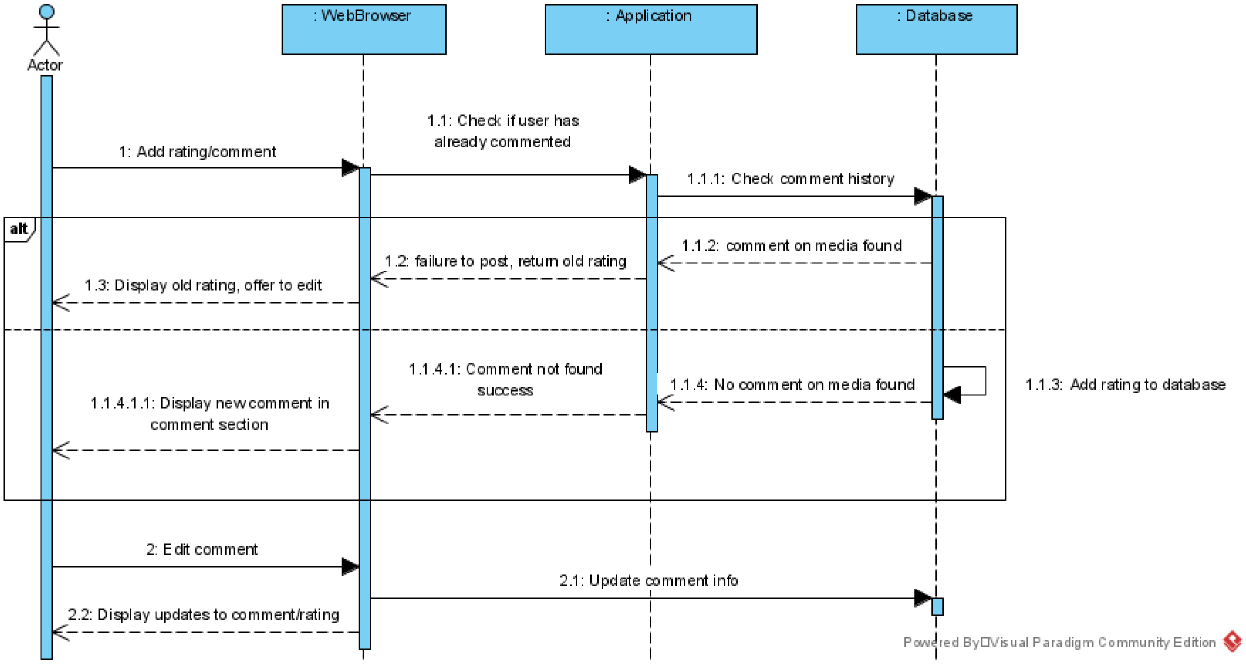
The below diagram is in relation to how the program will gather information on movies or albums.


Actor
: WebBrowser
: Application
: Database
: externalDatabase
1: Click on show/album to get details
1.1: Send movie details clicked
1.1.1: Request album /show details
1.1.2: Show/playlist details
1.2: Give details of selected show/album
1.3: Display content specific interface and where to stream content
2: Click provided streaming service
2.1: Redirect to streaming service
Powered By Visual Paradigm Community Edition

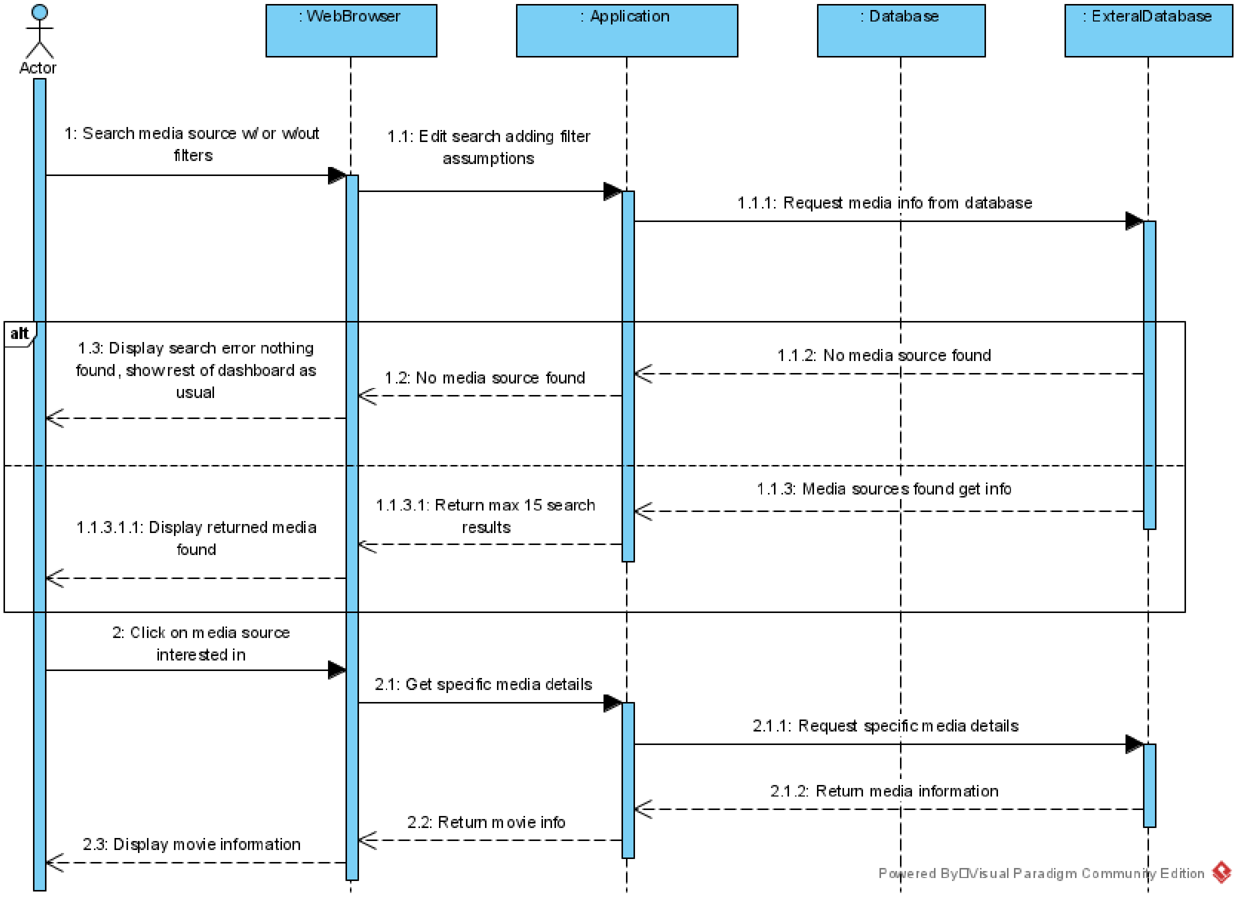
The following diagram illustrates the process of liking particular pieces of content.


Actor
: WebBrowser
: Application
: Database
1: Like a media source
1.1: Add to like count of movie
1.2: Showuser liked source
2: Check history on profile
2.1: Get history information
2.1.1: Request liked media sources
2.1.2: Liked media sources
2.2: Return liked movies in most recently liked
2.3: Display users liked history
Powered By Visual Paradigm Community Edition

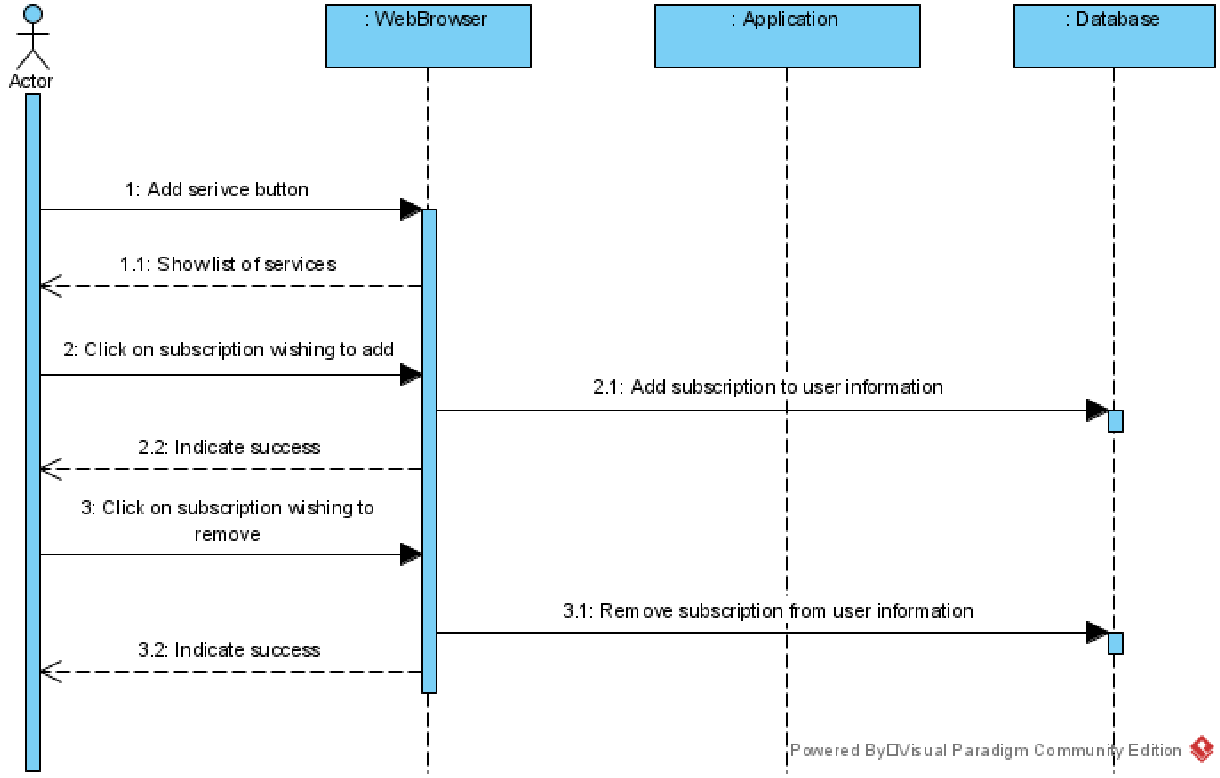
The below diagram demonstrates the process of adding ratings and comments.


: WebBrowser
: Application
: Database
Actor
1.1: Check if user has already commented
1: Add rating/comment
1.1.1: Check comment history
alt
1.1.2: comment on media found
1.2: failure to post, return old rating
1.3: Display old rating, offer to edit
1.1.4.1: Comment not found success
1.1.4: No comment on media found
1.1.3: Add rating to database
1.1.4.1.1: Display new comment in comment section
2: Edit comment
2.1: Update comment info
2.2: Display updates to comment/rating
Powered By Visual Paradigm Community Edition

The following diagram demonstrates the process of searching for a particular piece of media.

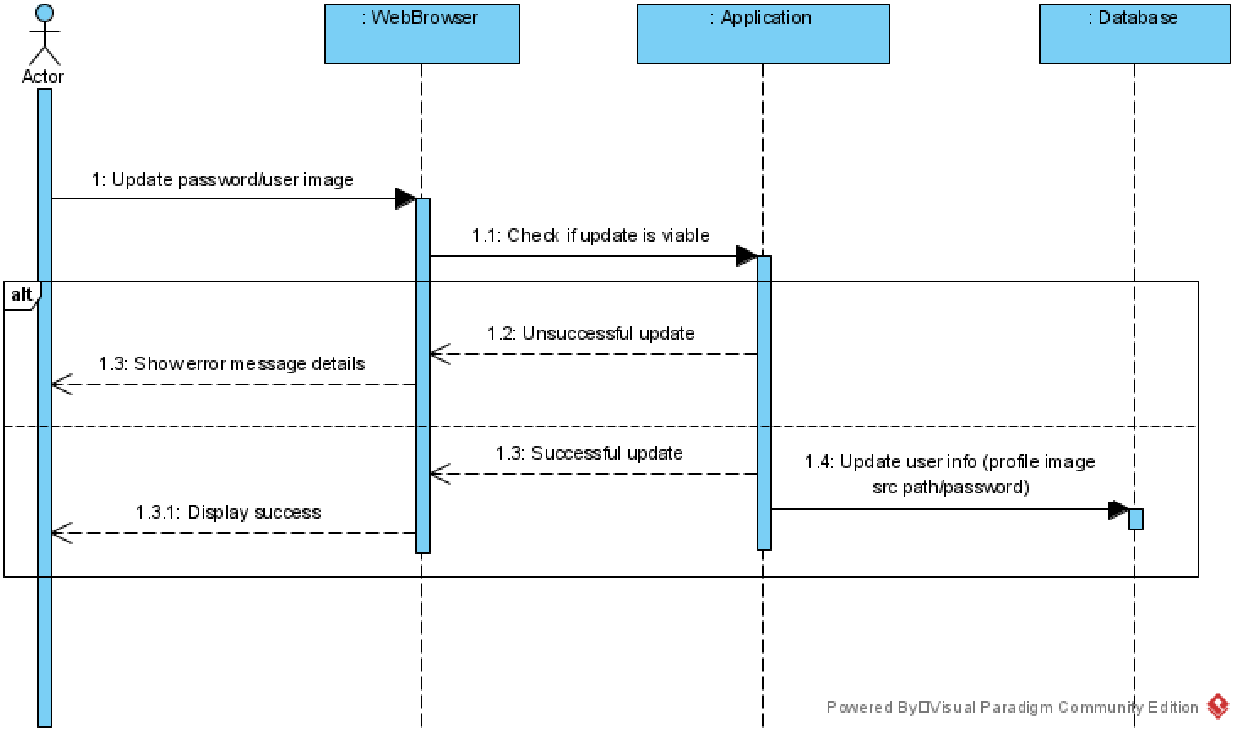

Actor
: WebBrowser
: Application
: Database
: ExteralDatabase
1: Search media source w/ or w/out filters
1.1: Edit search adding filter assumptions
1.1.1: Request media info from database
alt
1.3: Display search error nothing found, show rest of dashboard as usual
1.2: No media source found
1.1.2: No media source found
1.1.3: Media sources found get info
1.1.3.1: Return max 15 search results
1.1.3.1.1: Display returned media found
2: Click on media source interested in
2.1: Get specific media details
2.1.1: Request specific media details
2.1.2: Return media information
2.2: Return movie info
2.3: Display movie information
Powered By Visual Paradigm Community Edition

The following diagram demonstrates the software's ability to list the available streaming services for a particular show.



Actor
: WebBrowser
: Application
: Database
1: Add serivce button
1.1: Show list of services
2: Click on subscription wishing to add
2.1: Add subscription to user information
2.2: Indicate success
3: Click on subscription wishing to remove
3.1: Remove subscription from user information
3.2: Indicate success
Powered By Visual Paradigm Community Edition

The final diagram represents the process of updating a password and changing aspects of a user profile.


Actor
: WebBrowser
: Application
: Database
1: Update password/user image
1.1: Check if update is viable
alt
1.2: Unsuccessful update
1.3: Show error message details
1.3: Successful update
1.4: Update user info (profile image src path/password)
1.3.1: Display success
Powered By Visual Paradigm Community Edition

4.0 Conclusion

This report has provided an illustration of how the project’s software architecture has been selected in response to selected user requirements and the overall interface design. Moreover, it has demonstrated the initial software design process.