Next Reminder

Description

An application to help people decide on what to watch/read/listen to next. The user can use the name of the media (music, movies, TV shows, books, video games or podcasts) they like to get similar recommendations. They can then check out the recommendation details, and if they are interested, bookmark the media as a memo for the next time they are out of ideas.

If the user has bookmarked many items and doesn't know where to start, they can use the Random function to have the app choose one media for them from their bookmarks.

Features

- Search media recommendations in TasteDive's database.
- Filter search results beforehand, search with one or several names.
- Check the media's Wikipedia summary in a detail page.

- Bookmark media recommendations.
- Have the app choose a random media from the bookmarks.
- Browse the app in day or night mode following the phone’s settings.

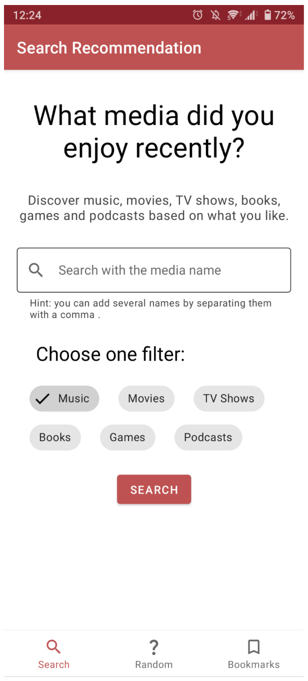
Overview

1. Home (MainActivity/HomeFragment)

The first screen that appears when the user opens the app. It is where the user can define a search by typing a media name (or several media names separated by a comma) and choosing one filter in the chip list.

Once the search is defined, a tap on the “Search” button will send an intent to open the Search Result screen, where the actual search request will be made.

If the input text field is left empty, an error message will be displayed under the field. The BottomNavigation can also be used to navigate to other screens (Random and Bookmarks).


12:24 72%

Search Recommendation

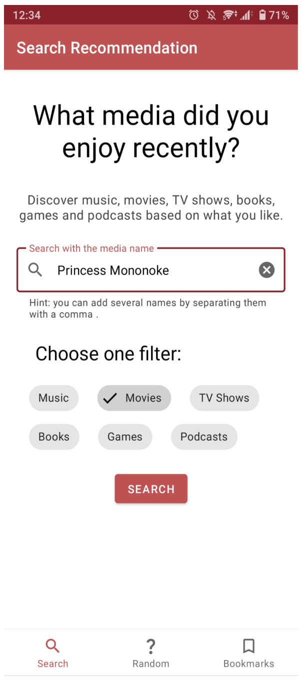
What media did you enjoy recently?

Discover music, movies, TV shows, books, games and podcasts based on what you like.

Search with the media name

Hint: you can add several names by separating them with a comma .

Choose one filter:


12:34 71%

Search Recommendation

What media did you enjoy recently?

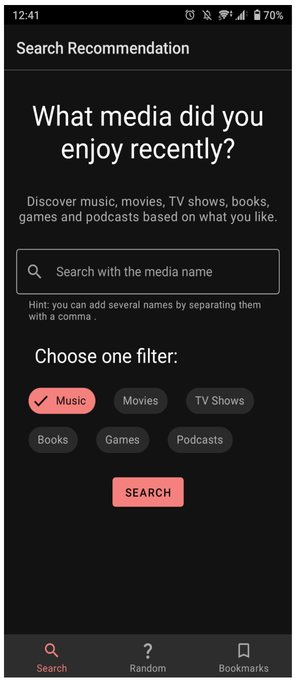
Discover music, movies, TV shows, books, games and podcasts based on what you like.

Search with the media name

Princess Mononoke

Hint: you can add several names by separating them with a comma .

Choose one filter:


12:41 70%

Search Recommendation

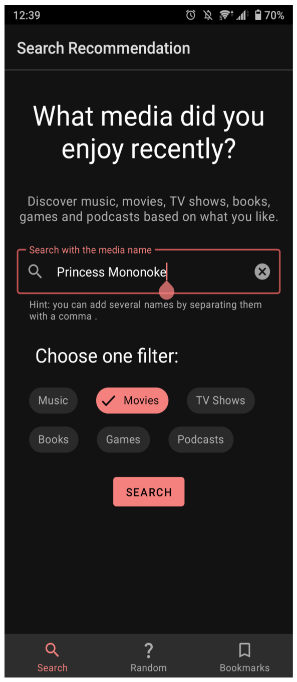
What media did you enjoy recently?

Discover music, movies, TV shows, books, games and podcasts based on what you like.

Search with the media name

Hint: you can add several names by separating them with a comma .

Choose one filter:


12:39 70%

Search Recommendation

What media did you enjoy recently?

Discover music, movies, TV shows, books, games and podcasts based on what you like.

Search with the media name

Princess Mononoke

Hint: you can add several names by separating them with a comma .

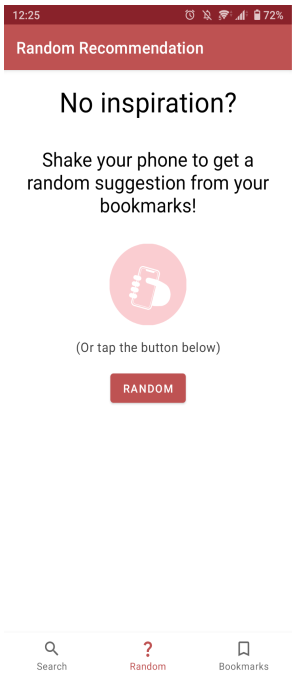
Choose one filter:

2. Random (MainActivity/RandomFragment)

This screen is the second option on the Bottom Navigation of MainActivity. The user can get the app to choose randomly an item from the bookmarks. The search is triggered if the user shakes their phone device or taps on the Random button.

Accelerometer is used to detect if the user shook their phone. An animation is displayed to hint at the shaking functionality, and another one during the loading state.

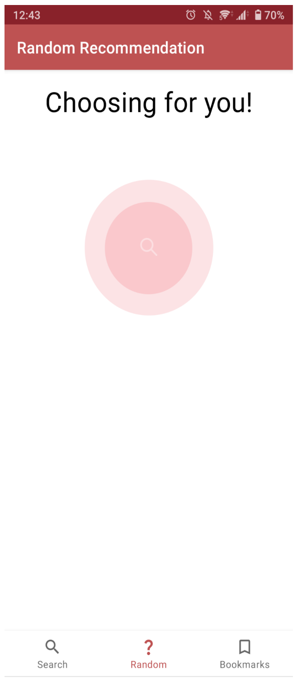
Once the loading is finished, the app will open the Detail screen to display the randomly chosen item. If there is no bookmark, the search won't be launched and a Toast will inform the user about the fact they need to add bookmarks first.



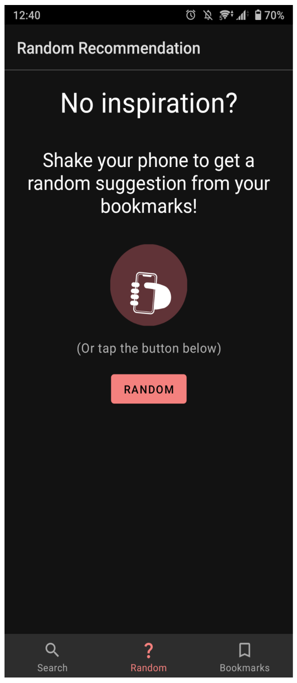
No inspiration?

Shake your phone to get a random suggestion from your bookmarks!

(Or tap the button below)



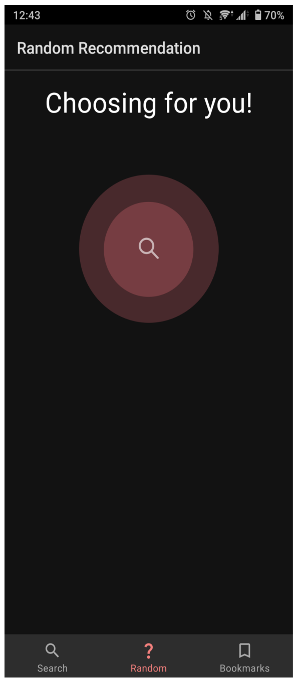
Choosing for you!



No inspiration?

Shake your phone to get a random suggestion from your bookmarks!

(Or tap the button below)



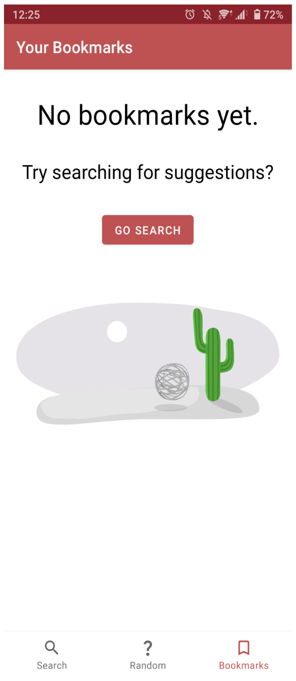
Choosing for you!

3. Bookmarks (MainActivity/BookmarkFragment)

This screen is the third option on the Bottom Navigation of MainActivity.It displays the media items that the user bookmarked during previous searches. Only the item type, name and a few lines of the summary are displayed, in addition to a bookmark icon.

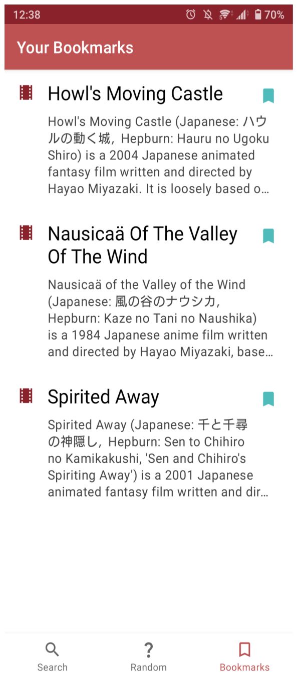
The user can tap on the item to see its details in the Detail screen, or tap on the bookmark icon to remove the item from the bookmarks. Removing a bookmark will remove the item from the list and trigger a confirmation SnackBar that offers the option to add again the bookmark, in case the user misclicked. Opening the Detail screen will launch a Shared Element Transition, animating the title, the item type icon and the description.

If the user has not added any bookmark yet, an empty screen explaining the situation will be displayed, containing an animation illustrating the empty state and a button that sends the user to the search screen once tapped.



No bookmarks yet.

Try searching for suggestions?



Howl's Moving Castle

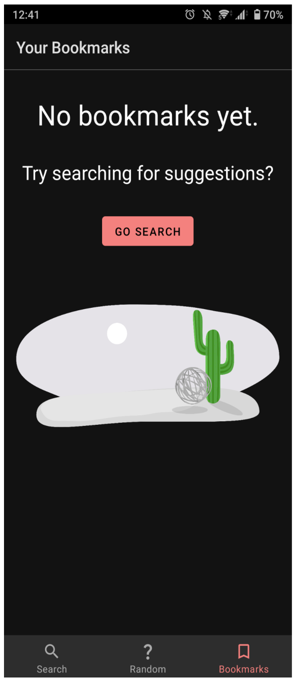
Howl's Moving Castle (Japanese: ハウルの動く城, Hepburn: Hauru no Ugoku Shiro) is a 2004 Japanese animated fantasy film written and directed by Hayao Miyazaki. It is loosely based o…

Nausicaä Of The Valley Of The Wind

Nausicaä of the Valley of the Wind (Japanese: 風の谷のナウシカ, Hepburn: Kaze no Tani no Naushika) is a 1984 Japanese anime film written and directed by Hayao Miyazaki, base…

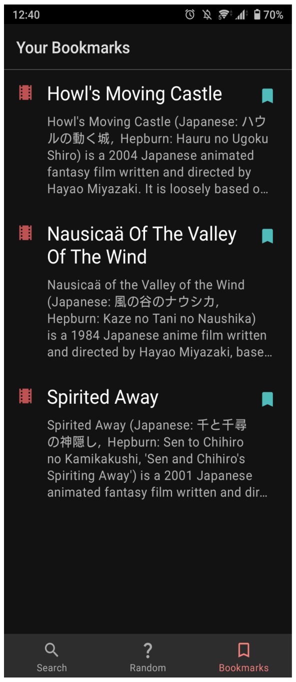
Spirited Away

Spirited Away (Japanese: 千と千尋の神隠し, Hepburn: Sen to Chihiro no Kamikakushi, 'Sen and Chihiro's Spiriting Away') is a 2001 Japanese animated fantasy film written and dir…



No bookmarks yet.

Try searching for suggestions?



Howl's Moving Castle

Howl's Moving Castle (Japanese: ハウルの動く城, Hepburn: Hauru no Ugoku Shiro) is a 2004 Japanese animated fantasy film written and directed by Hayao Miyazaki. It is loosely based o…

Nausicaä Of The Valley Of The Wind

Nausicaä of the Valley of the Wind (Japanese: 風の谷のナウシカ, Hepburn: Kaze no Tani no Naushika) is a 1984 Japanese anime film written and directed by Hayao Miyazaki, base…

Spirited Away

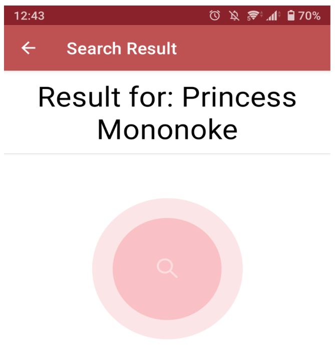
Spirited Away (Japanese: 千と千尋の神隠し, Hepburn: Sen to Chihiro no Kamikakushi, 'Sen and Chihiro's Spiriting Away') is a 2001 Japanese animated fantasy film written and dir…

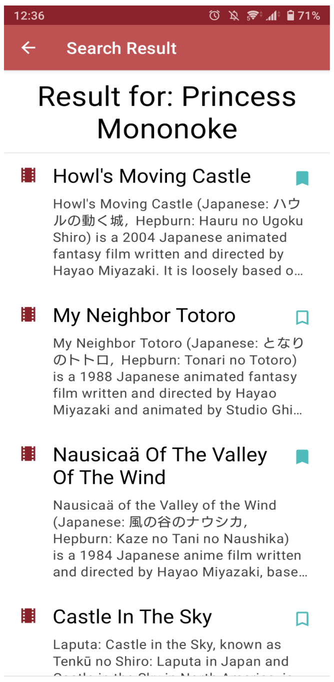
4. Search Result (SearchResultActivity)

The search result screen is opened from HomeFragment with a Bundle containing the search keyword and filter. The API query is launched after checking for the Bundle content to handle null state if necessary. The screen title containing the search keyword is displayed with a loading animation while the network query is being done.

If the search has results, the item list will be displayed up to 20 items. The user can tap on an item to check its details in the Detail screen, or tap on the bookmark icon to save the item in the local database. Opening the Detail screen will launch a Shared Element Transition, animating the title, the item type icon and the description.

If the search did not return any item, an empty screen containing an error message, a "Search Again" button to return to the search screen and an animation illustrating the situation. If there was a network error, the "Search Again" button and the animation will be displayed, but the error message will be displayed through a Toast.


12:43 70%
Search Result
Result for: Princess Mononoke

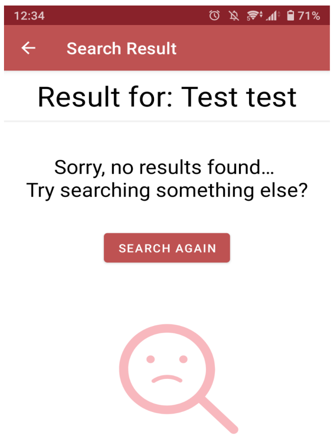

12:36 71%
Search Result
Result for: Princess Mononoke

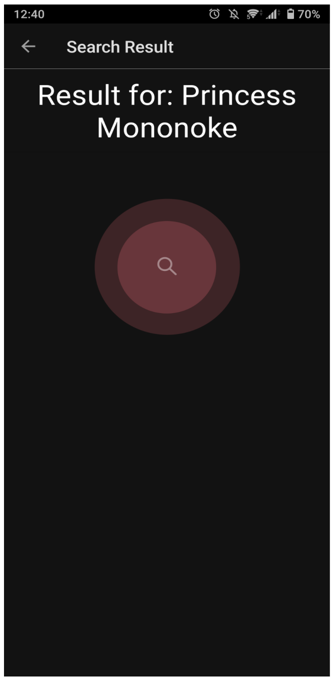
Howl's Moving Castle
Howl's Moving Castle (Japanese: ハウルの動く城, Hepburn: Hauru no Ugoku Shiro) is a 2004 Japanese animated fantasy film written and directed by Hayao Miyazaki. It is loosely based o...

My Neighbor Totoro
My Neighbor Totoro (Japanese: となりのトトロ, Hepburn: Tonari no Totoro) is a 1988 Japanese animated fantasy film written and directed by Hayao Miyazaki and animated by Studio Ghi...

Nausicaä Of The Valley Of The Wind
Nausicaä of the Valley of the Wind (Japanese: 風の谷のナウシカ, Hepburn: Kaze no Tani no Naushika) is a 1984 Japanese anime film written and directed by Hayao Miyazaki, base...

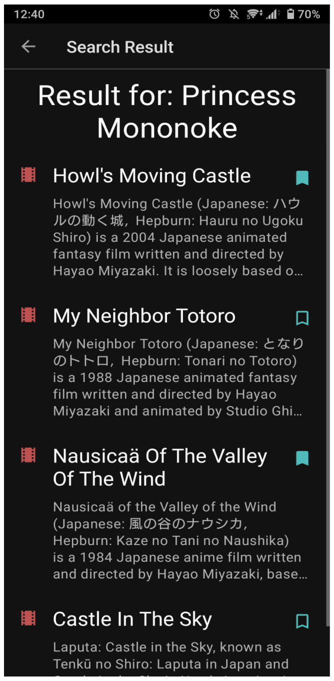
Castle In The Sky
Laputa: Castle in the Sky, known as Tenkū no Shiro: Laputa in Japan and


12:34 71%
Search Result
Result for: Test test
Sorry, no results found...
Try searching something else?


12:40 70%

Search Result

Result for: Princess Mononoke


12:40 70%

Search Result

Result for: Princess Mononoke

Howl's Moving Castle

Howl's Moving Castle (Japanese: ハウルの動く城, Hepburn: Hauru no Ugoku Shiro) is a 2004 Japanese animated fantasy film written and directed by Hayao Miyazaki. It is loosely based o…

My Neighbor Totoro

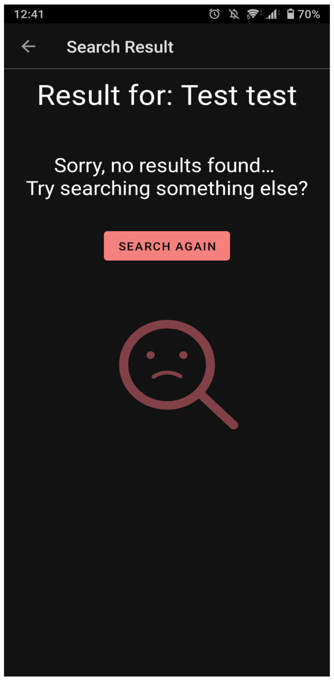
My Neighbor Totoro (Japanese: となりのトトロ, Hepburn: Tonari no Totoro) is a 1988 Japanese animated fantasy film written and directed by Hayao Miyazaki and animated by Studio Ghi…

Nausicaä Of The Valley Of The Wind

Nausicaä of the Valley of the Wind (Japanese: 風の谷のナウシカ, Hepburn: Kaze no Tani no Naushika) is a 1984 Japanese anime film written and directed by Hayao Miyazaki, base…

Castle In The Sky

Laputa: Castle in the Sky, known as Tenkū no Shiro: Laputa in Japan and


12:41 70%

Search Result

Result for: Test test

Sorry, no results found…
Try searching something else?

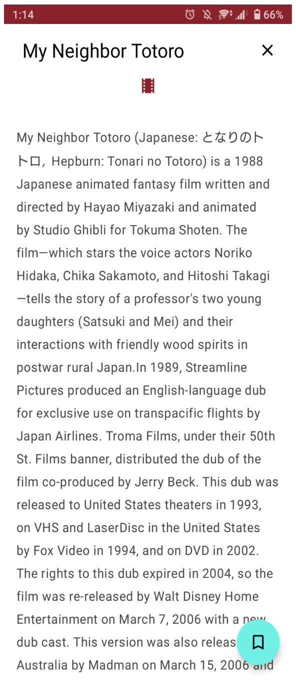
5. Detail (DetailActivity)

The Detail screen is opened from the Search Result, the Bookmarks or the Random screens. It displays all the data available (returned from the API) about a media: the title, the item type, its Wikipedia summary, the Wikipedia page URL and a Youtube video URL. A bookmark icon is also available as a Floating Action Button, allowing the user to add or remove the item from the bookmarks.

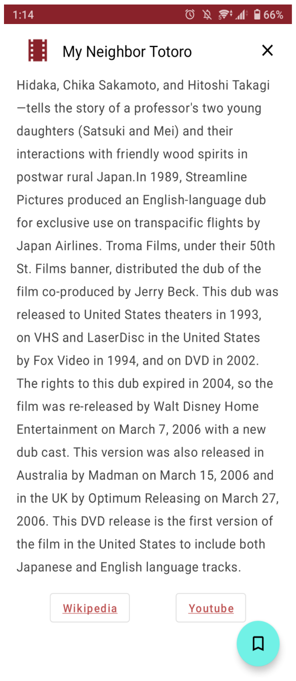
The screen can be closed by tapping on the button at the top right corner or by using the system's back button at the bottom. If the screen was opened through a Shared Element Transition, closing it will launch the animation in reverse, animating the title, the item type icon and the description back to their place in the item list.

An animation using MotionLayout is also used to animate the item type icon and the title in a CollapsingToolbar fashion when the user scrolls down. Keeping the item type icon and title on the screen allows the user to have all information available while checking the rest of the summary. The item type icon will also rotate on itself before ending at the top left of the screen.

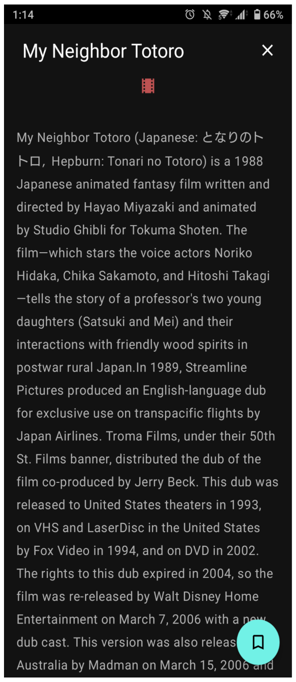
The Wikipedia and Youtube buttons at the bottom of the screen open their respective URLs in the device's default browser.



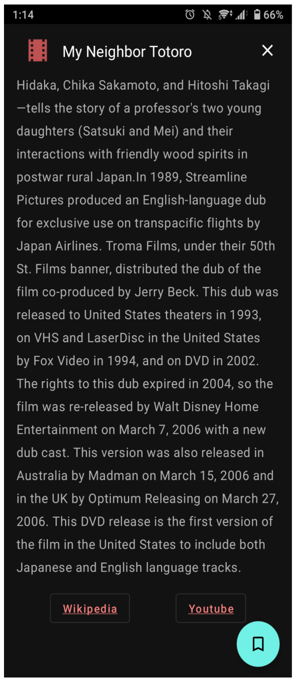
My Neighbor Totoro (Japanese: となりのトトロ, Hepburn: Tonari no Totoro) is a 1988 Japanese animated fantasy film written and directed by Hayao Miyazaki and animated by Studio Ghibli for Tokuma Shoten. The film—which stars the voice actors Noriko Hidaka, Chika Sakamoto, and Hitoshi Takagi —tells the story of a professor's two young daughters (Satsuki and Mei) and their interactions with friendly wood spirits in postwar rural Japan.In 1989, Streamline Pictures produced an English-language dub for exclusive use on transpacific flights by Japan Airlines. Troma Films, under their 50th St. Films banner, distributed the dub of the film co-produced by Jerry Beck. This dub was released to United States theaters in 1993, on VHS and LaserDisc in the United States by Fox Video in 1994, and on DVD in 2002. The rights to this dub expired in 2004, so the film was re-released by Walt Disney Home Entertainment on March 7, 2006 with a new dub cast. This version was also releas Australia by Madman on March 15, 2006 and



Hidaka, Chika Sakamoto, and Hitoshi Takagi —tells the story of a professor's two young daughters (Satsuki and Mei) and their interactions with friendly wood spirits in postwar rural Japan.In 1989, Streamline Pictures produced an English-language dub for exclusive use on transpacific flights by Japan Airlines. Troma Films, under their 50th St. Films banner, distributed the dub of the film co-produced by Jerry Beck. This dub was released to United States theaters in 1993, on VHS and LaserDisc in the United States by Fox Video in 1994, and on DVD in 2002. The rights to this dub expired in 2004, so the film was re-released by Walt Disney Home Entertainment on March 7, 2006 with a new dub cast. This version was also released in Australia by Madman on March 15, 2006 and in the UK by Optimum Releasing on March 27, 2006. This DVD release is the first version of the film in the United States to include both Japanese and English language tracks.



My Neighbor Totoro (Japanese: となりのトトロ, Hepburn: Tonari no Totoro) is a 1988 Japanese animated fantasy film written and directed by Hayao Miyazaki and animated by Studio Ghibli for Tokuma Shoten. The film—which stars the voice actors Noriko Hidaka, Chika Sakamoto, and Hitoshi Takagi —tells the story of a professor's two young daughters (Satsuki and Mei) and their interactions with friendly wood spirits in postwar rural Japan.In 1989, Streamline Pictures produced an English-language dub for exclusive use on transpacific flights by Japan Airlines. Troma Films, under their 50th St. Films banner, distributed the dub of the film co-produced by Jerry Beck. This dub was released to United States theaters in 1993, on VHS and LaserDisc in the United States by Fox Video in 1994, and on DVD in 2002. The rights to this dub expired in 2004, so the film was re-released by Walt Disney Home Entertainment on March 7, 2006 with a new dub cast. This version was also releas Australia by Madman on March 15, 2006 and



Hidaka, Chika Sakamoto, and Hitoshi Takagi —tells the story of a professor's two young daughters (Satsuki and Mei) and their interactions with friendly wood spirits in postwar rural Japan.In 1989, Streamline Pictures produced an English-language dub for exclusive use on transpacific flights by Japan Airlines. Troma Films, under their 50th St. Films banner, distributed the dub of the film co-produced by Jerry Beck. This dub was released to United States theaters in 1993, on VHS and LaserDisc in the United States by Fox Video in 1994, and on DVD in 2002. The rights to this dub expired in 2004, so the film was re-released by Walt Disney Home Entertainment on March 7, 2006 with a new dub cast. This version was also released in Australia by Madman on March 15, 2006 and in the UK by Optimum Releasing on March 27, 2006. This DVD release is the first version of the film in the United States to include both Japanese and English language tracks.

Roadmap

1. Decide on basic UI, core functionalities and color palette.
2. Set up the project, create the first screen, add dependencies, make sure the project builds.
3. Roughly decide on architecture, create folders and empty fragments accordingly, set up Navigation Components graph and BottomNav.
4. Add Retrofit, get and hide in a suitable place the key for TasteDive API, create a basic network request and handle its response.
5. Create the search result screen, add DataBinding and RecyclerView adapters.
6. Add Room, create database.
7. Add bookmark functionality to the search result screen.
8. Create the bookmarks screen to display all current bookmarks.
9. Create the detail screen.
10. Create the random screen, use Accelerometer to know if the user shook the device.
11. Handle queries with several names and the search filter.
12. Handle errors and screen rotation for each screen.
13. Add MotionLayout to the detail screen.
14. Add animations and a shared element transition.
15. Clean XML (check theme and style consistency, store resources in suitable places).
16. Check accessibility.
17. Add tests.

Project Specifications

Android UI/UX

Build a navigable interface consisting of multiple screens of functionality and data.

Application includes at least three screens with distinct features using either the Android Navigation Controller or Explicit Intents.	The app has 5 screens. Both Navigation Components and Explicit Intents are used.
The Navigation Controller is used for Fragment-based navigation and intents are utilized for Activity-based navigation.	Navigation Controller is used to handle the Bottom Navigation between the 3 main fragments in the MainActivity. Intents are used to reach other activities (SearchResultActivity and DetailActivity).
An application bundle is built to store data passed between Fragments and Activities.	An application bundle is used to pass datas to SearchResultActivity and DetailActivity.

Construct interfaces that adhere to Android standards and display appropriately on screens of different size and resolution.

Application UI effectively utilizes ConstraintLayout to arrange UI elements effectively and efficiently across application features, avoiding nesting layouts and maintaining a flat UI structure where possible.	ConstraintLayout is used to arrange UI elements effectively such as list_item. Nesting is avoided everywhere, except for activity_detail due to the NestedScrollView with MotionLayout.
Data collections are displayed effectively, taking advantage of visual hierarchy and arrangement to display data in an easily consumable format.	The search result list and the bookmark list display the items data in an easily consumable format.
Resources are stored appropriately using the internal res directory to store data in appropriate locations including string values, drawables, colors, dimensions, and more.*	All resources are stored in appropriate locations.
Every element within ConstraintLayout should include the id field and at least 1 vertical constraint.	All elements in ConstraintLayout have an ID assigned and at least 1 vertical constraint.

Data collections should be loaded into the application using ViewHolder pattern and appropriate View, such as RecyclerView.	The search result list and the bookmark list use RecyclerView to display their data.

Animate UI components to better utilize screen real estate and create engaging content.

Application contains at least 1 feature utilizing MotionLayout to adapt UI elements to a given function. This could include animating control elements onto and off screen, displaying and hiding a form, or animation of complex UI transitions.	The detail screen uses MotionLayout to animate the title and type icon when the user scrolls down to keep the information on the screen while allowing the user to keep reading the summary.
MotionLayout behaviors are defined in a MotionScene using one or more Transition nodes and ConstraintSet blocks.	The MotionLayout behavior is defined in a MotionScene (detail_title_motion), using 1 Transition node and 2 ConstraintSet blocks.
Constraints are defined within the scenes and house all layout params for the animation.	Constraints are defined within the scenes and contain all necessary layout params.

Local and Network data

Connect to and consume data from a remote data source such as a RESTful API.

*The Application connects to at least 1 external data source using **Retrofit** or other appropriate library/component and retrieves data for use within the application.*	The application connects to 1 external data source (TasteDive) using Retrofit.
*Data retrieved from the remote source is held in local models with appropriate data types that are readily handled and manipulated within the application source. Helper libraries such as **Moshi** may be used to assist with this requirement.*	Moshi is used to parse the network request.
The application performs work and handles network requests on the appropriate threads to avoid stalling the UI.	Network requests are made on background thread using Coroutines.

Load network resources, such as Bitmap Images, dynamically and on-demand.

*The Application loads remote resources asynchronously using an appropriate library such as **Glide** or other library/component when needed.*	The API response does not return images so such libraries were not needed.
Images display placeholder images while being loaded and handle failed network requests gracefully.	There are no images loaded from external sources, however the app does display placeholder/loading images while waiting for network requests, and when on error/empty screens.
All requests are performed asynchronously and handled on the appropriate threads.	All network requests or loading are done on background thread using Coroutines.

Store data locally on the device for use between application sessions and/or offline use.

*The application utilizes storage mechanisms that best fit the data stored to store data locally on the device. Example: SharedPreferences for user settings or an internal database for data persistence for application data. Libraries such as **Room** may be utilized to achieve this functionality.*	Room is used to store the user's bookmarks.
Data stored is accessible across user sessions.	The bookmarked items can still be browsed when the user opens the app again at a later time.
Data storage operations are performed on the appropriate threads as to not stall the UI thread.	Operations to the database are done on a background thread using Coroutines.
Data is structured with appropriate data types and scope as required by application functionality.	Data retrieved from the network and from the database are handled by different data classes (SimilarItem and SimilarEntity). SimilarEntity contains an ID parameter that acts as the PrimaryKey, which does not exist in the network response (SimilarItem). A third data class (SimilarDTO) is used as the final format to communicate the data to the ViewModels/Activity and XML files. It does not contain the Entity ID's parameter, however it has an isBookmarked parameter that does not exist in the other formats.

Android system and hardware integration

Architect application functionality using MVVM.

Application separates responsibilities amongst classes and structures using the MVVM Pattern: ● *Fragments/Activities control the Views* ● *Models houses the data structures,* ● *ViewModel controls business logic.*	The application uses the MVVM pattern, with a Model handling the communication with the database and the network requests (SimilarRepository), ViewModels (for each Activities) to control the business logic and Activities/Fragments to control the Views.
Application adheres to architecture best practices, such as the observer pattern, to prevent leaking components, such as Activity Contexts, and efficiently utilize system resources.	The application uses the observer pattern (with viewLifecycleOwner in Fragments), and does not use the Context in ViewModels and layers above.

Implement logic to handle and respond to hardware and system events that impact the Android Lifecycle.

Beyond MVVM, the application handles system events, such as orientation changes, application switching, notifications, and similar events gracefully including, but not limited to: ● *Storing and restoring state and information* ● *Properly handling lifecycle events in regards to behavior and functionality* ○ *Implement bundles to restore and save data* ● *Handling interaction to and from the application via Intents* ● *Handling Android Permissions*	The application handles orientation changes and application switching, for example by saving the inputted search keywords as a LiveData.

Utilize system hardware to provide the user with advanced functionality and features.

Application utilizes at least 1 hardware component to provide meaningful functionality to the application as a whole. Suggestion options include: ● *Camera* ● *Location* ● *Accelerometer* ● *Microphone* ● *Gesture Capture*	The random screen uses the Accelerometer to detect if the user shook their phone to launch a search.

• *Notifications*	
Permissions to access hardware features are requested at the time of use for the feature.	The Accelerometer does not require special permissions, except by mentioning its use in the Manifest.
Behaviors are accessed only after permissions are granted.	See above.