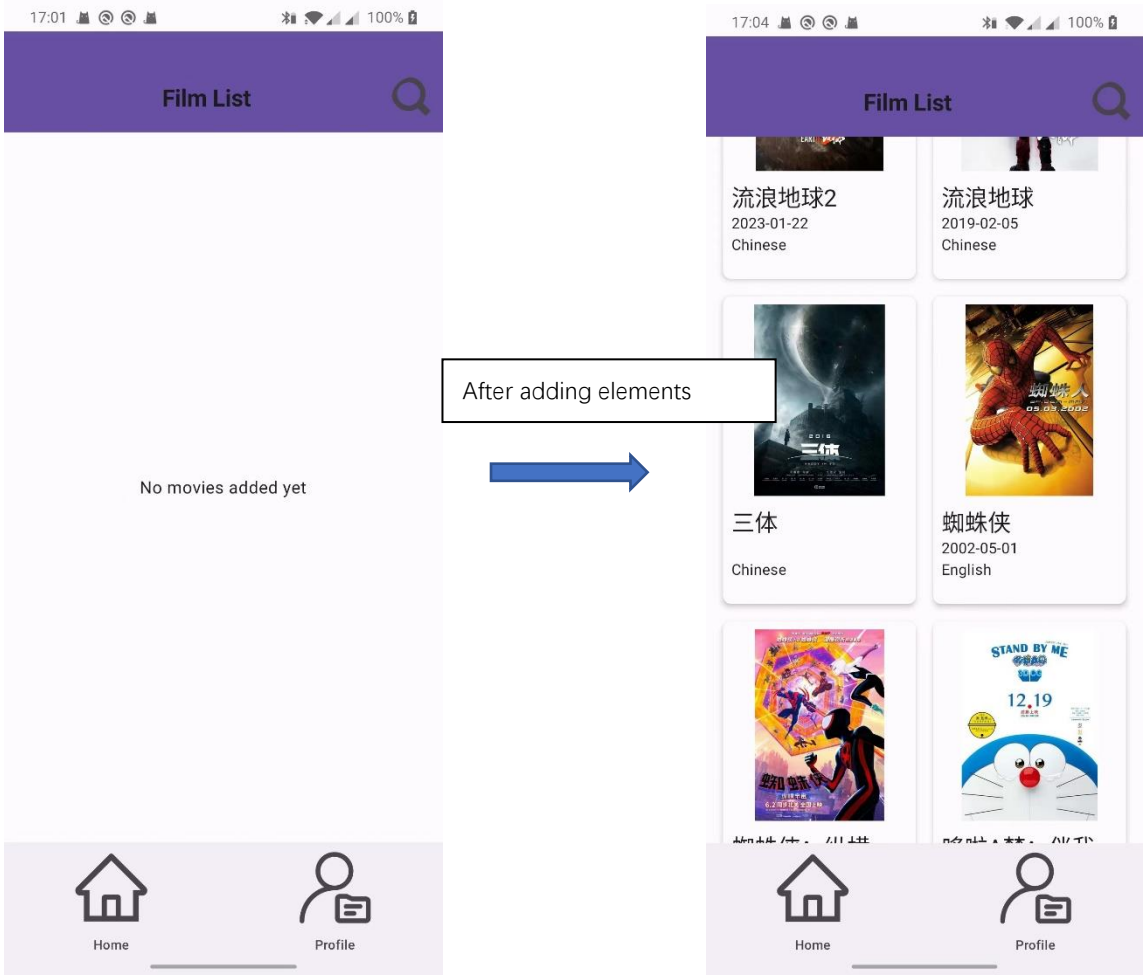
Implementation details

All requirements are satisfied as follow:

1. Each item in the list view contain at least one image, text description, and the title of film indicating the index of the item.



The corresponding code can be found in component/FilmCard.kt, ui/MovieScreen.kt, ui/HomeScreen.kt, and MainActivity.kt

2. The image should be loaded asynchronously

```
import coil.compose.rememberAsyncImagePainter
```

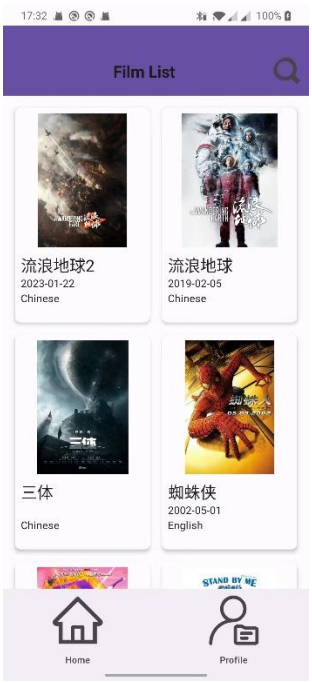
```
Image(
    painter = rememberAsyncImagePainter(posterUrl),
    contentDescription = film.title,
    modifier = Modifier
        .fillMaxWidth()
        .height(180.dp)
        .clip(RoundedCornerShape(8.dp)) // set the shape
)
```

Using coil.compose.rememberAsyncImagePainter to achieve.

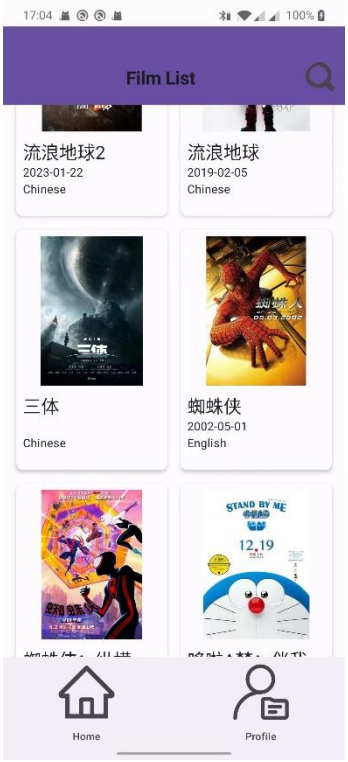
It is inside component/FilmCard.kt and component/SearchCard.kt

3. The item should be clickable to jump to another activity

If you click the item of each film, it will go to the detail page and show the detail information of the film



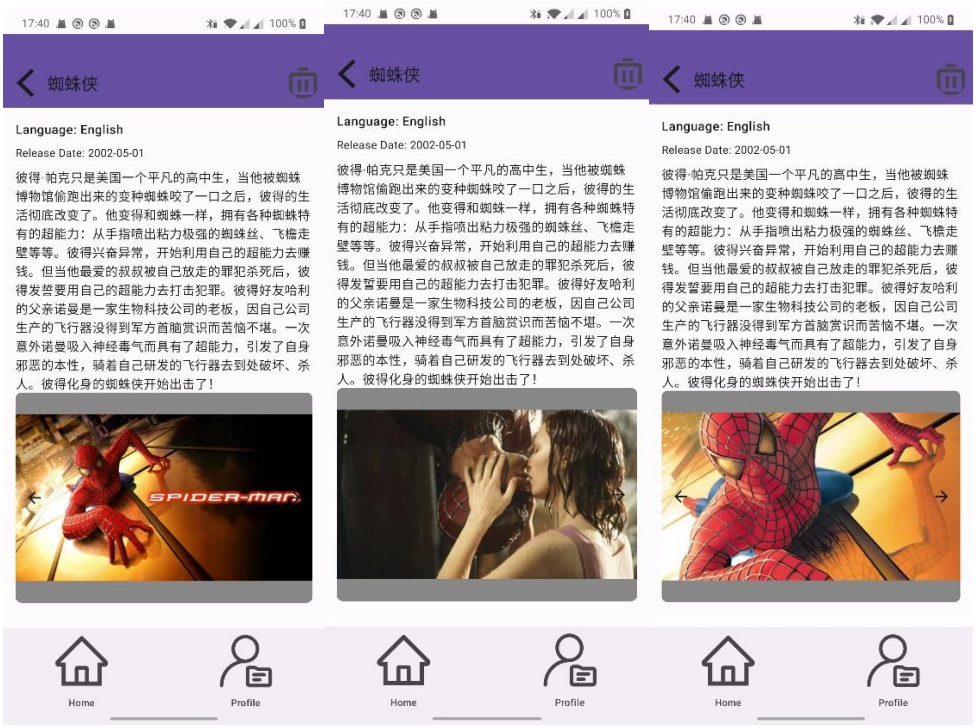
After clicking the film “流浪地球 2”, it jumps to show the details



After clicking the film “蜘蛛侠”, it jumps to show the details

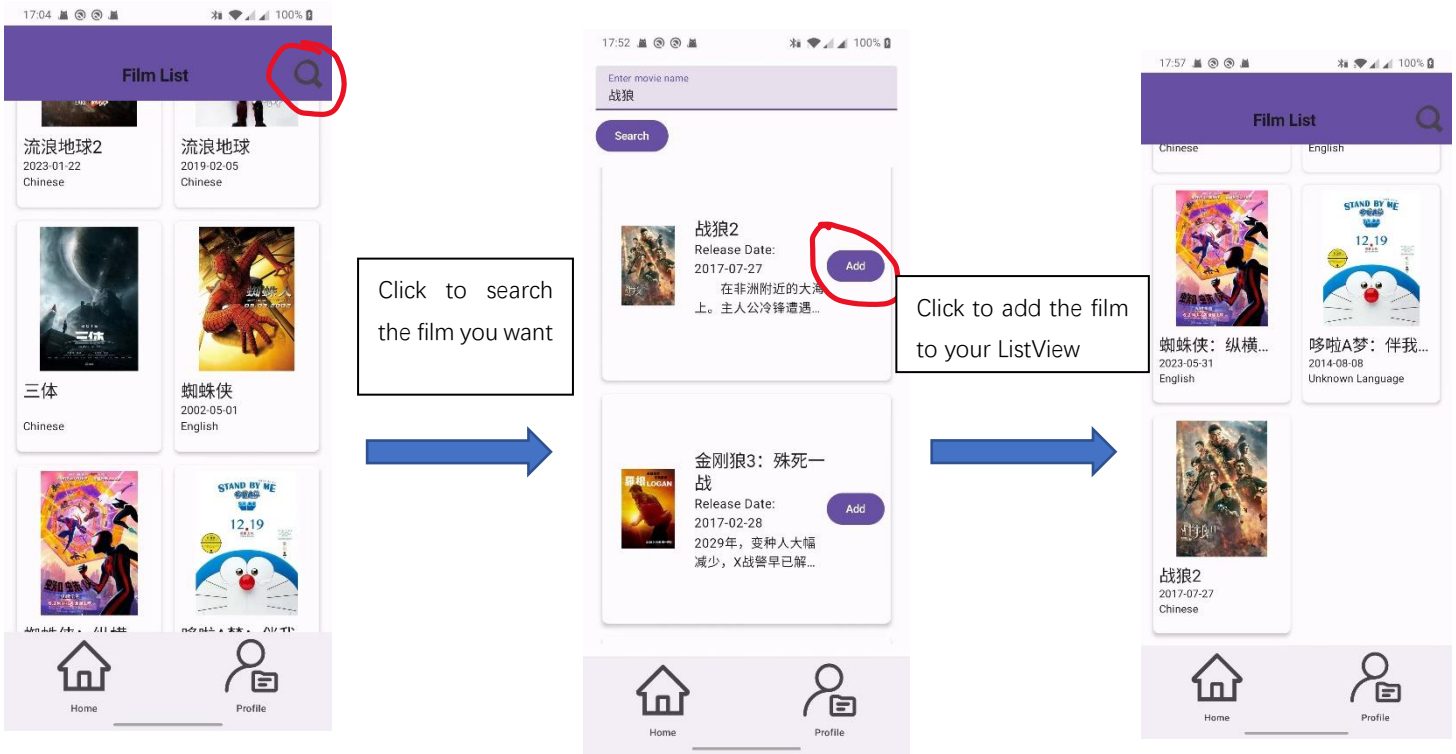
The corresponding code can be found in component/FilmCard.kt and ui/DetailScreen.kt

4. The new activity contain an image slider with multiple images.
The detail page contains multiple images of the film, click the arrow of the image slider and it will change to next image of the film.

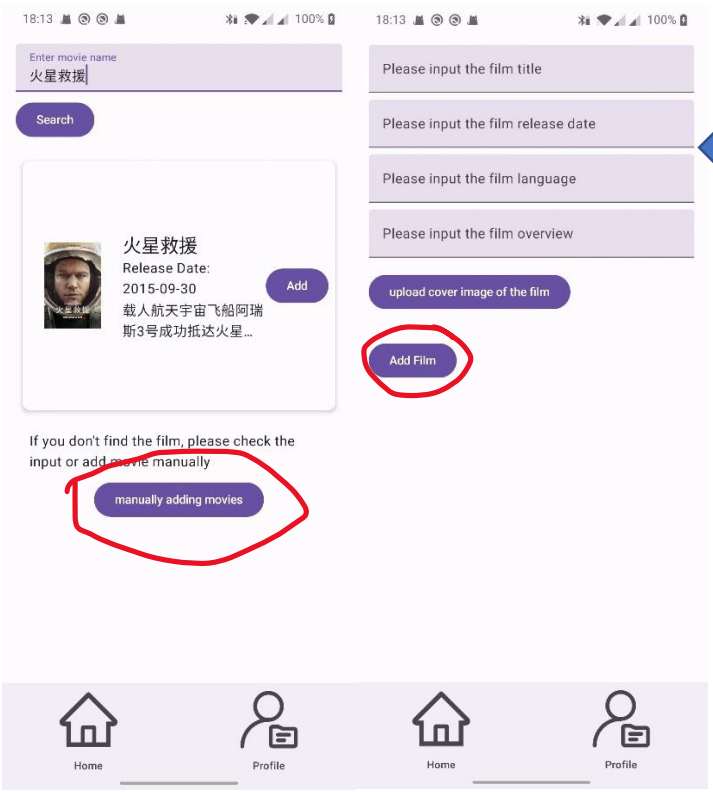


The corresponding code can be found in ui/DetailScreen.kt

5. The ListView is scrollable. And click the search button in the top right corner, search the film you want to add, click the add button of the result, and the new item will be added to the ListView.



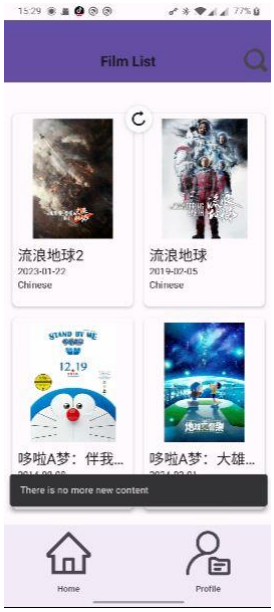
And if you don't find the film you want, you can click the manually add film button and add add new item by yourself.



Input the information of the film here

The corresponding code can be found in MainActivity.kt, ui/SearchScreen.kt, component/SearchCard.kt, data/FilmApi.kt and data/FilmApiResponse.kt

6. After pull-and-release for five times, the list view should indicate there is no more new content

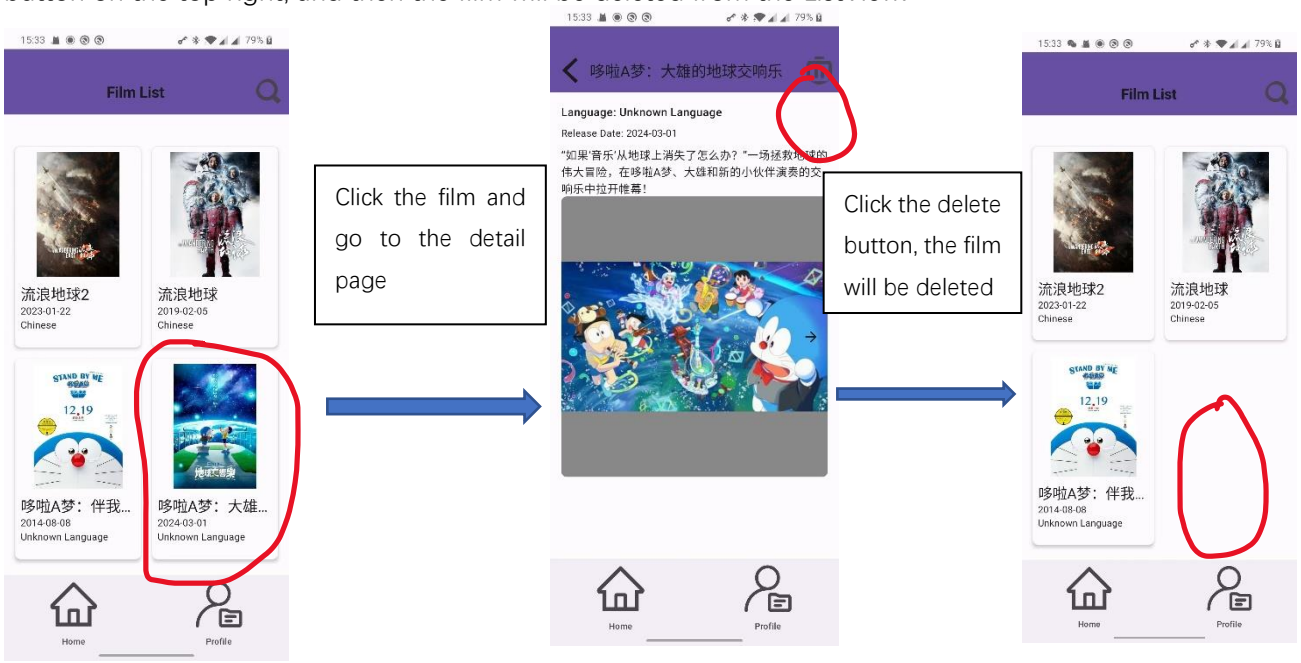


The corresponding code can be found in ui/MovieScreen.kt

7. The item can be deleted (2%)

To delete the item of ListView, you should click it and go to the detail page, click the delete

button on the top right, and then the film will be deleted from the ListView.



The corresponding code can be found in ui/DetailScreen.kt

```
actions = {
    IconButton(onClick = {
        // delete film from FilmData
        FilmData.films.removeIf { it.id == film.id }
        navController.popBackStack() // back to previous page
    }) {
        Icon(
            painterResource(id = R.drawable.delete),
            contentDescription = "Delete"
        )
    }
},
```

7. In pressing home button and re-enter the APP, the APP should stay in the content page of previous usage and do not crash.

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        FilmData.loadFilms( context: this) // load data
```

```
override fun onPause() {
    super.onPause()
    FilmData.saveFilms( context: this) // save data
}
```

```kotlin
// save the FilmData
fun saveFilms(context: Context) {
    val prefs = context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE)
    val editor = prefs.edit()
    val gson = Gson()
    val json = gson.toJson(films)
    editor.putString(FILM_KEY, json)
    editor.apply()
}

fun loadFilms(context: Context) {
    val prefs = context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE)
    val gson = Gson()
    val json = prefs.getString(FILM_KEY, null)
    val type = object : TypeToken<MutableList<Film>>() {}.type

    // Parse JSON directly into a list
    val loadedFilms: MutableList<Film> = gson.fromJson(json, type) ?: mutableListOf()

    // Clear the current list and add loaded movies
    films.clear()
    films.addAll(loadedFilms)
}
```

The app will use the above functions to save the data. If you have already added some films to the content page and close the app, it will automatically save what films you have added, and next time when you reopen the app, it will load the film data that you have already added to your list view.
The code can be found in MainActivity.kt and FilmData.kt

8. Load the contents in the listview from online resources.

```kotlin
val request = Request.Builder()
    .url(url)
    .addHeader( name: "accept",  value: "application/json")
    .addHeader( name: "Authorization", apiKey) // Replace with your API Key
    .build()
```

```
val jsonResponse = it.body?.string() ?: return
val gson = GsonBuilder().create()
val filmApiResponseType = object : TypeToken<FilmApiResponse>() {}.type
val filmApiResponse: FilmApiResponse = gson.fromJson(jsonResponse, filmApiResponseType)

// extract the required Film data
val films = filmApiResponse.results.map { apiFilm ->
    Film(
        id = apiFilm.id,
        title = apiFilm.title,
        releaseDate = apiFilm.release_date,
        overview = apiFilm.overview,
        posterPath = apiFilm.poster_path ?: "",
        localPosterPath="",
        originalLanguage = getLanguageName(apiFilm.original_language)
    )
}

onResult(films) // returns a list of extracted films
```

In FilmApi.kt, we have fetchMovie functions to help us load the online information of film. User can search the film they want to add, and in the search screen it will show the return information of the API. And if user add a film to the list view, the app will record the online resource and add it to the list view, including the information of the movie was added. And if a user can't the film they want from the API, they can also add the movie manually. Because these kinds of film don't have online resource, the information include poster image will store and load locally.

The mainly code can be found in ui/SearchScreen.kt, component/SearchCard.kt, component/FilmCard.kt, data/FilmApi.kt and data/FilmApiResponse.kt

What's more is that the API of online resources needs VPN to connect, and I use my personal API key.