3rd Year Project:

Innovative Cinema Listings

Technical Manual

Authors:

Shaun Carey *16450454*

Nigel Guven *14493422*

Supervisor:

Charles Daly

Table of Contents

1. Introduction

1.1 Overview

The project is developed around a mobile application developed solely for Android operating systems. The application displays cinema listings in a user-friendly guided format. These cinemas are in Ireland and Northern Ireland only. Users are told where the nearest cinema is and what time the next film will start at and if the user will be late. Requirements necessary for a smooth performance are the enabling of Wi-Fi and user location from within their Android mobile settings menu. The system will prevent the user from progressing further if neither of these requirements are met, both of which are mandatory. The system involves scraping of HTML web data and using the Java library JSoup to parse it into a readable string which can then be manipulated by the system or displayed to the user and solid information.

There is a link to the Google Maps API wherein the user's location is displayed and with their nearest local cinemas displayed as accessible map markers. The web scraping scripts are initialized at runtime so the user can be presented with the latest cinema-listings information. Picasso is another library for Android which is built in to the application to download film images from a URL and cache them into the system database. The web scripts are initialized when the user clicks a button but are ran asynchronously to prevent a clogging of memory or 'lagging' on the UI thread. The application does not coordinate with other systems although it does require an active connection to the websites that it is scraping from.

1.2 Glossary

JSoup: A Java library purpose-built for interacting with HTML. Operates by making a connection to a webpage and grabbing the HTML within which can be extracted.

Asynchronous: Asynchronous execution is performed when other tasks are finished processing. Tasks that are asynchronous operate at different times to regular computation

Picasso: Picasso is an Android library that is used to download and display images with no loss of data and with full resolution of the picture.

HTML: Hypertext Markup Language or HTML is a prime languages for creating web pages and web applications. It defines the text and order but not the structure or interaction of a document

API: An Application Programming Interface or API allows the creation of applications that access the features or data of an operating system and application.

SDK: A Software Development Kit or SDK is a tool that allows for the creation of applications for certain software package, platform or computer system.

Gradle: Gradle is a build-automation system which configures Android applications at runtime by downloading necessary libraries and setting the API level.

ADB: Android Debug Bridge is a tool used to read Android mobile data from a computer which is not necessarily able to utilise Android apps. Essentially a comms platform between a computer and an android device.

IDE: Integrated Development Environments are software applications with comprehensive facilities for allowing its users to develop other pieces of software.

Android Code Version: The value represents the version of the applications code.

Android Activity Lifecycle: Android operates by an activity lifecycle. Transitions between different interfaces and themselves are controlled by methods like onCreate, onStop, and OnResume.

Android Manifest: The Android Manifest is an XML file which contains all the activities in an Android app as well as setting the permissions which the Android app has access to.

Listview: Listview is an XML type which contains data and displays it in a list

Scrollview: Scrollview is an XML type which allows for dynamic scrolling of an activity where not all data might fit into a screen.

Kotlin: Kotlin is a programming language that is primarily used for developing Android apps. It can be interoperated alongside Java in any application.

ChaquoPy: ChaquoPy is a library which enables a user to manipulate Python on Android Studio where the IDE doesn't support Python. It allows for cross-compatibility with Java and Kotlin in the IDE.

ArrayList: Arraylists are data types which are lists of data. Unlike arrays which are simple, Arraylists can be expanded dynamically.

LogCat: LogCat is the internal debugging terminal which retrieves all data from an Android device and displays its information inside of Android Studio.

onCreate: onCreate is the initialization function of an activity. It calls an XML layout file and runs a set of functions which display on the UI Thread.

2. Installation Guide

2.1 System Specifications

Developed on HUAWEI PRA-LX1 and LG-X Venture Android phones
Platform Support: Android Mobile devices only
Android OS Minimum API/SDK Version: API14/Android 4.0 (Ice Cream Sandwich)
Android OS Maximum API/SDK Version: API27/Android 8.1 (Oreo)
Runnable on 100% of Android devices.
Android Code Version: 1.0

External Requirements:
Google Maps API Key
Google Play Services v16.0.0
Google Maps Services v16.1.0
Square Picasso v2.5.2
JSoup v1.11.3 :*download via* https://jsoup.org/
Android Gradle v3.3.1

2.2 Installation Instructions

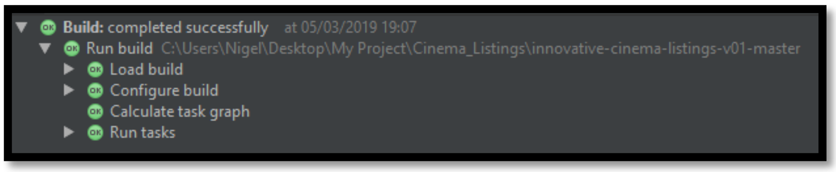
1. Prerequisites:
   a. Android Studio is installed on the user's device.
   b. ADB is installed on the users device.

2. Download the Project Repository from its repository located on Gitlab at the link: https://gitlab.computing.dcu.ie/careys26/2019-ca326-innovative-cinema-listings.

3. Find the downloaded project on the users file system (*by default in the Downloads folder on Windows, Mac or Linux operating systems*).

4. Open the application in the Android Studio IDE. This can be found by looking for the Android Studio icon in the applications folder. This icon called app is the root of the entire application.

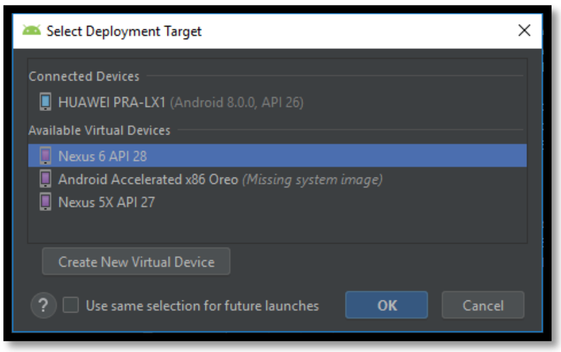
5. Wait for the program to build and execute its tasks.



```
▼ Build: completed successfully   at 05/03/2019 19:07
  ▼ Run build  C:\Users\Nigel\Desktop\My Project\Cinema_Listings\innovative-cinema-listings-v01-master
    ▶ Load build
    ▶ Configure build
      Calculate task graph
    ▶ Run tasks
```

6. A Run option should become available at the top.

7. Click the Run button. In the case where the user has not downloaded ADB, an alternative option to download a virtual device is available.



Select Deployment Target

Connected Devices

HUAWEI PRA-LX1 (Android 8.0.0, API 26)

Available Virtual Devices

Nexus 6 API 28

Android Accelerated x86 Oreo *(Missing system image)*

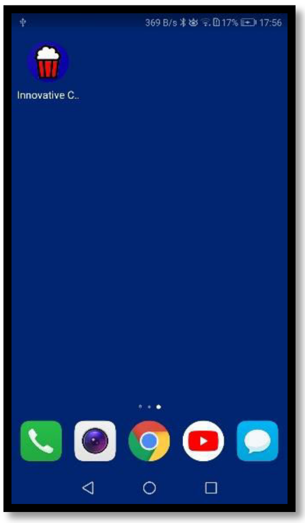
Nexus 5X API 27

Create New Virtual Device
Use same selection for future launches
OK
Cancel

8. The application will appear on the selected device either through the ADB or the virtual device.

9. Google will ask the user to sign in to an email (*This is necessary so that Google can get the users location for the app to function correctly*).

10. Once logging in the app will display on the home screen as a shortcut.



11. Click on the application icon to start it.

12. See the User Manual for further information about how to use the application.

3. System Architecture

Fig 3.1

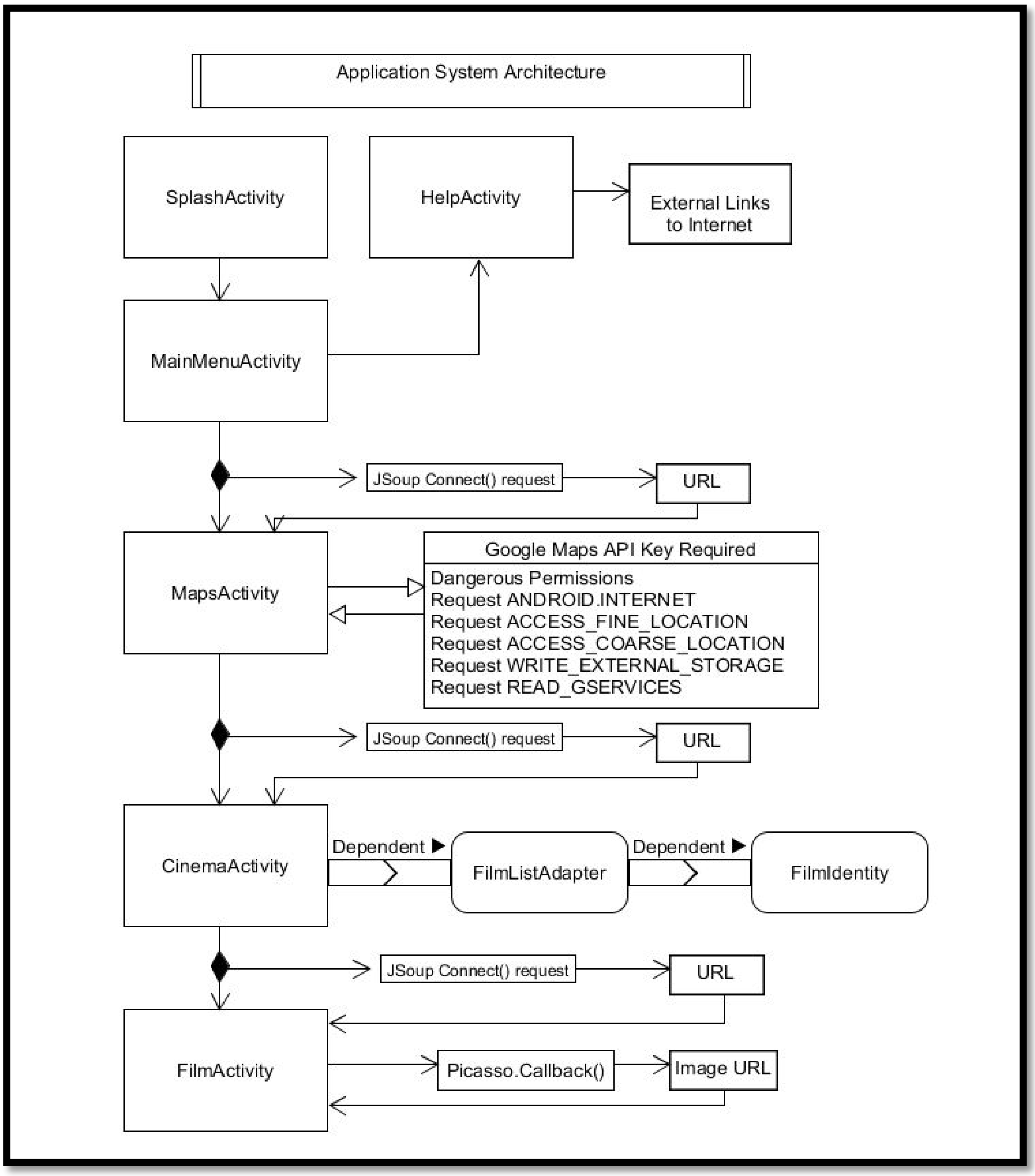
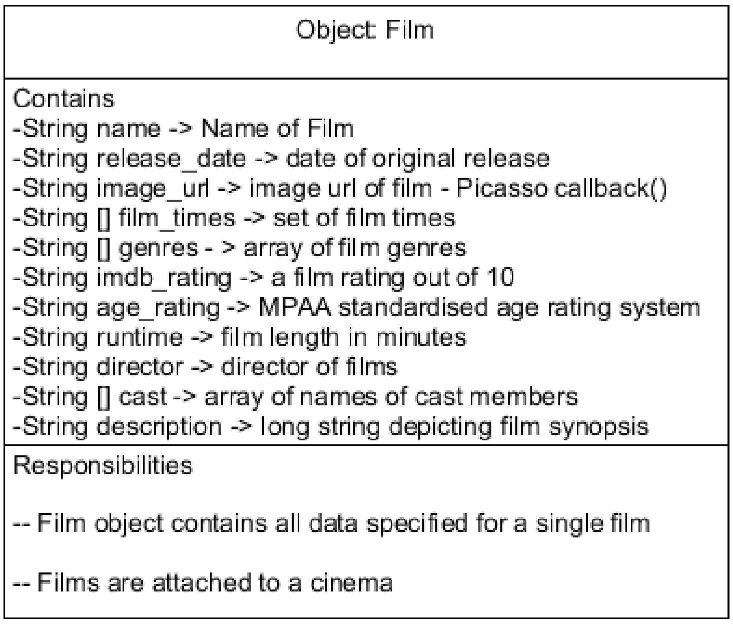


Fig 3.1 contains the overall architecture of the system. Android uses the activity lifecycle where each activity is a new user interface with its own links, functions and dependencies. A Google Maps API key is required for the Map to load up. Also the application requires certain permissions which are contained in the Android Manifest. JSoup and Picasso have launcher functions to retrieve data from external URL's. The FilmListAdapter and FilmIdentity classes are dependent in order from the activity where they display information in a ListView.

4. High-Level Design

Fig 4.1



Object: Film

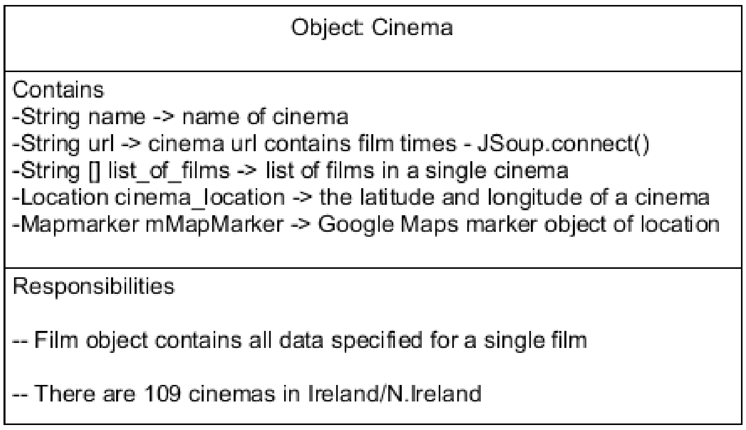
Contains
-String name -> Name of Film
-String release_date -> date of original release
-String image_url -> image url of film - Picasso callback()
-String [] film_times -> set of film times
-String [] genres - > array of film genres
-String imdb_rating -> a film rating out of 10
-String age_rating -> MPAA standardised age rating system
-String runtime -> film length in minutes
-String director -> director of films
-String [] cast -> array of names of cast members
-String description -> long string depicting film synopsis

Responsibilities

-- Film object contains all data specified for a single film

-- Films are attached to a cinema

Fig 4.2



Object: Cinema

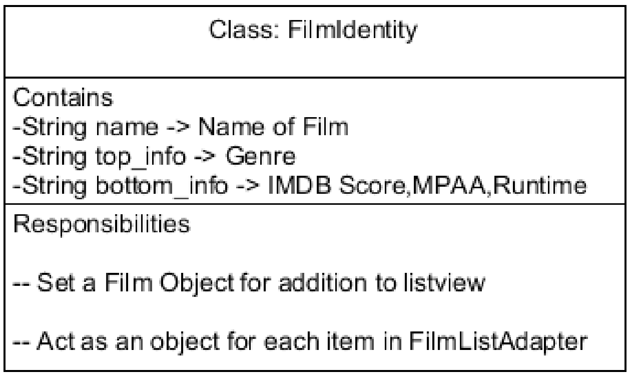
Contains
-String name -> name of cinema
-String url -> cinema url contains film times - JSoup.connect()
-String [] list_of_films -> list of films in a single cinema
-Location cinema_location -> the latitude and longitude of a cinema
-Mapmarker mMapMarker -> Google Maps marker object of location

Responsibilities

-- Film object contains all data specified for a single film

-- There are 109 cinemas in Ireland/N.Ireland

Fig 4.3



Class: FilmIdentity

Contains
-String name -> Name of Film
-String top_info -> Genre
-String bottom_info -> IMDB Score,MPAA,Runtime

Responsibilities

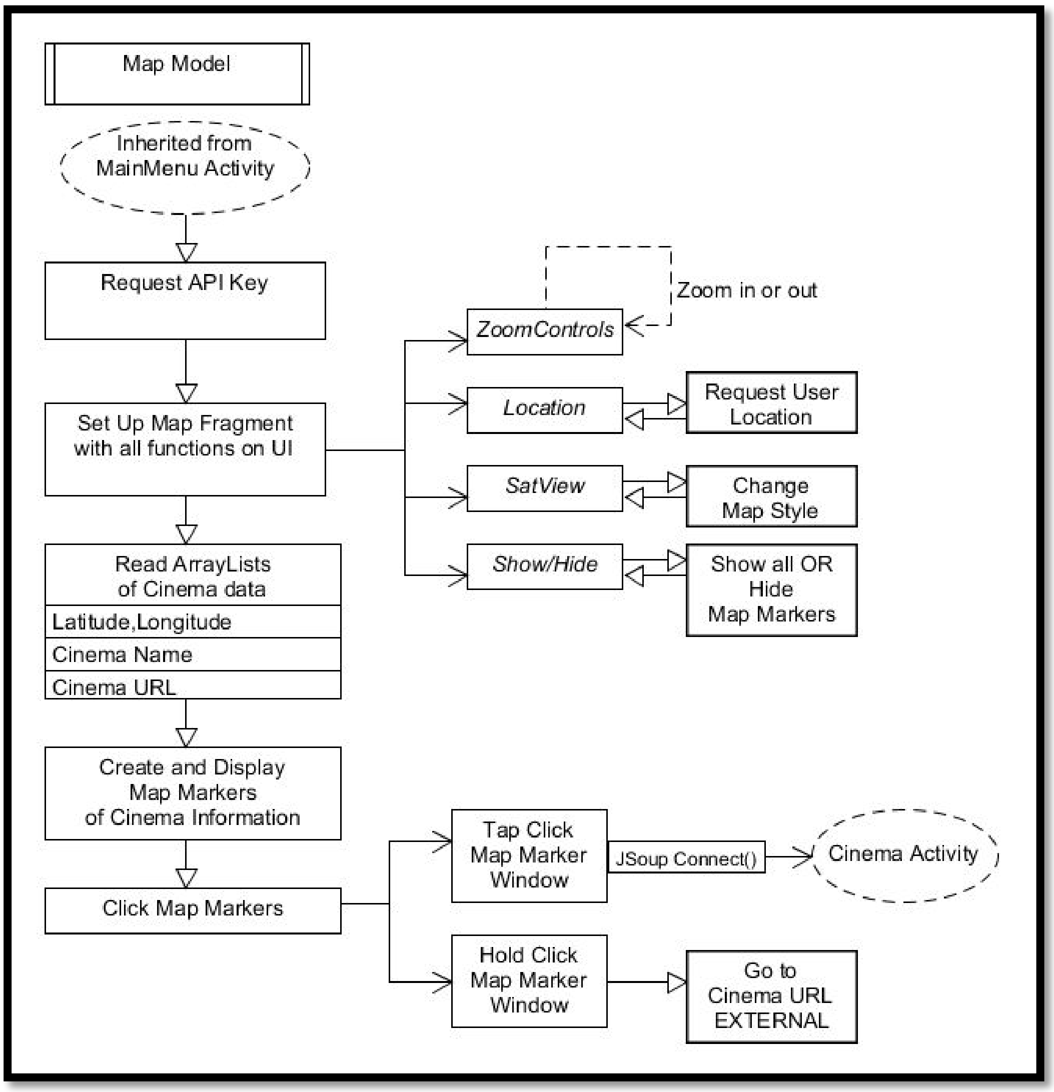
-- Set a Film Object for addition to listview

-- Act as an object for each item in FilmListAdapter

Fig 4.4

Class: FilmListAdapter
Contains -Context mContext -> CinemaActivity -Resource mResource -> ListView
Responsibilities -- Displays a FilmIdentity object -- Extends from default listview model -- Is dynamic to capture an ambiguous set of data

Fig 4.5



Map Model
Inherited from MainMenu Activity
Request API Key
Set Up Map Fragment with all functions on UI
Read ArrayLists of Cinema data
Latitude,Longitude
Cinema Name
Cinema URL
Create and Display Map Markers of Cinema Information
Click Map Markers
ZoomControls
Zoom in or out
Location
Request User Location
SatView
Change Map Style
Show/Hide
Show all OR Hide Map Markers
Tap Click Map Marker Window
JSoup Connect()
Cinema Activity
Hold Click Map Marker Window
Go to Cinema URL EXTERNAL

5. Problems and Resolutions

The following sections are dedicated to some of the major bugs that arose in the application and how we managed to solve them.

5.1 BeautifulSoup on Android Studio

Our original plan was to use BeautifulSoup, a Python library for scraping web data. Once we had the script written we had to examine how we would incorporate it into Android Studio. Some helper libraries like ChaquoPy would enable us to use Python on Android Studio which is primarily a Java/Kotlin based IDE. When we tried to incorporate BeautifulSoup, the application refused to accept it and ChaquoPy itself was causing problems.

Resolution: We decided to rewrite the script to scrape web data in Java. JSoup is the alternative to BeautifulSoup and we installed this as a library in the Android application. It worked much better albeit with a bigger hit to performance.

5.2 Film Data displaying twice in Cinema List View

The Film Data had been placed in a listview in the cinema activity. Each cinema has a certain amount of films on show. Films would display a small amount of information in the listview. However, we ran into a problem whereby when a cinema activity was opened, it displayed its films twice on the list.

Resolution: We realised that if a user clicked into a cinema then into another cinema, the arraylist of film data would append more information onto it without clearing information from the previous cinema. We created a check method to examine if an arraylist already contained data and if so to remove this data before adding another cinema's data.

5.3 Web Scraper crashing when run on UIThread

We placed the web scraper in the beginning to the onCreate function of the activity it was in. The onCreate activity would run the script and its regular data. Everytime the activity was activated, it would crash with a fatal error in LogCat.

Resolution: We realised that a massive web scraper would not work in the onCreate method. We created a thread to run asynchronously in between each activity. When a user clicked button, it would fire the script into action. This solved the problem of the fatal crash and hanging on the UI thread.

5.4 ListView not working within a Scrollview

In the Film Activity, there contains all the film data which is inside of an XML Scrollview. The scrollview is also expandable. We wanted to display the film times inside of the scrollview in an intuitive way. Our original design was a listview with a film time and a highlighted object to display the nearest time to the users current time.

Upon entering the film activity interface, the listview was not visible and would not expand inside the scrollview. The XML was causing a problem. One solution was to make the listview a separate object but this did not make it intuitive for a user to view.

Resolution: We decided to create a function which displayed rows and columns of film data. The film times were made to appear orderly. This was successful in that it appeared in a correct format while also appearing dynamically on the interface.

-End of Document-