**A Penny for Your Thoughts Ltd.** Movie Match for Android Software Design Document

Matthew Penny

Date: 12/08/2022

**Table of Contents**

1. [Introduction 2](#_bookmark0)
   1. [Purpose 2](#_bookmark1)
   2. [Scope 2](#_bookmark2)
   3. [Overview 2](#_bookmark3)
   4. [Reference Material 2](#_bookmark4)
2. [S](#_bookmark5)ystem [O](#_bookmark5)verview [2](#_bookmark5)
3. [S](#_bookmark6)ystem [A](#_bookmark6)rchitecture [2](#_bookmark6)
   1. [Architectural Design **2**](#_bookmark7)
   2. [Decomposition Description 2](#_bookmark8)
   3. [Design Rationale 3](#_bookmark9)
4. Data Design [3](#_bookmark10)
   1. [Data Description **3**](#_bookmark11)
   2. [Data Dictionary 3](#_bookmark12)

[3](#_bookmark13)

1. [Component Design 3](#_bookmark14)
2. [Human Interface Design 4](#_bookmark15)
   1. [Overview of User Interface 4](#_bookmark16)
   2. [Screen Images 4](#_bookmark17)
   3. [Screen Objects and Actions 4](#_bookmark18)
3. Introduction

## Purpose

This is the Software Design Document for the capstone project of Matthew Penny, which consists of an Android application that uses an API from Movie of the Night1, a Swipe-able UI, Firebase's Real-time Database as well as it's Storage function. We will be aiming for devices running Android 7.0 and higher to be able to accommodate as many users as possible. We will use Diagrams.net as the wire frame software, as well as to create any diagrams required, including but not limited to: DFD's (Data Flow Diagrams), UML diagrams, and Use Cases. GitHub will be the VCS and there will be public access for any administrators, and instructors who require it.

## Scope

The initial scope of the project was for couples and friends who can never agree on what to watch due to a plethora of choices in the video streaming domain. To start, three services will be offered, with others being added after the timeline for the original product has passed. There will be the ability to play the movie from the application, as well as social media sharing. The project's head administrator has full access through the VCS and will have final approval of the application's viability. The modeling approach that we have decided on for the project is an Iterative model. This way, the head admin can watch the progress and make any changes or ask questions as to how the project is progressing. The project will also pass through three separate phases: initial planning (including diagrams and database mapping), coding and use case implementation) and complete documentation (including this document). As we have only the one individual working on the project, he will oversee the completion of all tasks within the timeline handed down by the college.

We mentioned above that this was the initial scope of the project. After a midterm meeting with the head administrator, we decided on a small shift away from the original scope. As of that meeting, we moved away from the original targets and broadened the scope. Also, social media sharing and the ability to open the third-party streaming sites was mentioned and will be implemented during the project.

## Overview

We will be using this document to detail the functionalities, context, and design of the capstone project. As the sole employee of the company, Matthew Penny is the head designer, programmer, UI/UX developer and will be referred to as "we" throughout the document. As the project manager, he will also oversee any changes to the document as well as the diagrams that can be found throughout.

1. System Overview

### What It Does

Movie Match for Android allows users to browse through multiple streaming services without having to change applications. It also allows users to connect and communicate with each other if they so desire. It can be used by individuals looking for a one-stop browsing application, couples and friends that have difficulty choosing what to watch and by individuals under lock down or isolation measures to have someone to chat with while watching movies and series.

### Device Requirements

The application is for Android devices only and those devices must be capable of running Android version 7.0 or better. This is a requirement of the streaming services and as such, cannot be altered. The programming language is Java, even though Kotlin is the language of Android, as it is the language with which our head programmer is the most comfortable. The devices should also be using the most current version of whichever web browser that they prefer, to allow for compatibility. The application can interact successfully with the different social media platforms, the device's messaging application as well as the different streaming platforms.

1. System Architecture

### User Interface

We originally attempted to use a Card style view to display the movies and series, unfortunately we had some issues populating the cards with the API results. As such, after starting the project with a new VCS link, we pivoted to a Recycler View to hold the API results. The older files that will be shared with the finished project will show the Card version, which will be reattempted outside of the current scope and timeline. We have used the Navigation Drawer and Contextual menus for the project, allowing for ease of navigation. See figure 3.a for an example.

*Figure 3.a Recycler View example*

Graphical user interface, application

Description automatically generated

### Database

For storing matches and returning data to the different Recycler Views, we decided on Firebase Realtime Database and Firebase Storage to suit our needs. In the original planning, we decided upon three tables, Users, Movies, and Matches. See Figure 3.b below

*Figure 3.b Original Database*

**Graphical user interface, application, Teams

Description automatically generated**

As shown above, the three tables and their cardinalities were created and implemented in the early stages of the project. Once we applied normalization to the database, we found that the Matches table was containing redundant information and as such we dropped the table. Most of the streaming information is coming from the API call and as such we did not have to store information beyond the ID number and title of the movies and series. Due to the structures of the Firebase Database system (JSON tree), figure 3.c shows a clearer picture as to how the remaining tables were built.

*Figure 3.c Firebase JSON Structure*

Graphical user interface

Description automatically generated with medium confidence

The above example is a better representation of how we structured the database. The Users table above shows the Primary Key and Foreign Key used to reference our other table (Movies). We were able to implement all matches and display all “liked” movies without having to implement another table. Figure 3.d will show how the Movies table was structured after the midterm change in scope.

*Figure 3.d Movies Table*

**Graphical user interface, application

Description automatically generated**

**3.3**

1. Data Design

### Data Description

### As mentioned in the last section, our database, user management, and storage were supplied by Firebase Database. Within the application itself, we used several different structures to be able to store and display the information without too much wasted memory. Among these structures, we used OOP concepts for both the Users model and the Movies model, ArrayList to hold the different IDs and titles and also HashMap to be able to iterate and read through the JSON tree response from the API calls.