

Updated Design Document for UrbanFlix

Authors: Emmanuel Gallegos, Jamie Nguyen, Dhruvinkumar Patel

Date: 2.21.20

Status: Version 1.0 Ready for Launch

1) Concept Summary

The purpose behind UrbanFlix is to provide the film viewing community with the means of quickly and simply sharing their opinions openly on the internet. Quick and simple are two attributes driving the methodology behind our concept. Once the application has been downloaded to the device, the user can immediately begin participating within the application's user base by seeing an updated stream of user reviews.

Once the user logs in, the user can search a film title and view all corresponding movie reviews, as well as make their own if they so wish. In order to optimize the user's experience, streamlining the simplicity of functionality is an overarching goal in the software's implementation. We've tried to limit users' ability to skew data by forcing users to create an account in order to actively affect the database of reviews.

2) Audience/Customer

UrbanFlix is targeted towards movie buffs and general film fans. We expect users to be comfortable downloading and deploying Android applications, and comfortable with the process of creating accounts manually via email/username/password. However, as accounts are not necessary if the user only wants to read reviews, the app should function fine even if the user does not have the capability to make an account.

3) Background

To understand the project, one should be familiar with the basic concepts of applications and the purpose of databases, as well as the idea of user accounts. The prerequisites for understanding the app are low; although, understanding the business models and interfaces of the following services will help explain our goals with regards to emulating parts of their designs within our app:

- Urban Dictionary - helpful for understanding the core concept of a user driven database
- IMDB - helpful for understanding what our app would like to offer; we want to provide an alternative to traditional movie database apps/sites such as IMDB that is purely user driven instead of being carefully curated
- Reddit - helpful for understanding how our algorithms will work with regards to upvoting, downvoting, and list priority

4) Application Cost and Projected Success

- App cost: \$0
- Monetization possibilities

- Advertisement page on startup, close it to make it stop appearing until next reopen
 - Ad banners between reviews within each movie (still not implemented)
 - Pro Account (\$0.99): (not implemented)
 - Ad Removal
 - View history of previously created reviews
- Projected Success Measures
 - A robust repository of user created film reviews
 - A large amount of active users and frequent participants

5) Interface Mockups

Home Page



Search (Movie Reviews)



Make Review

9:47

← Titanic →

Review Title

(30 character limit)

Review

(200 character limit)

Hashtags (Up to 5)

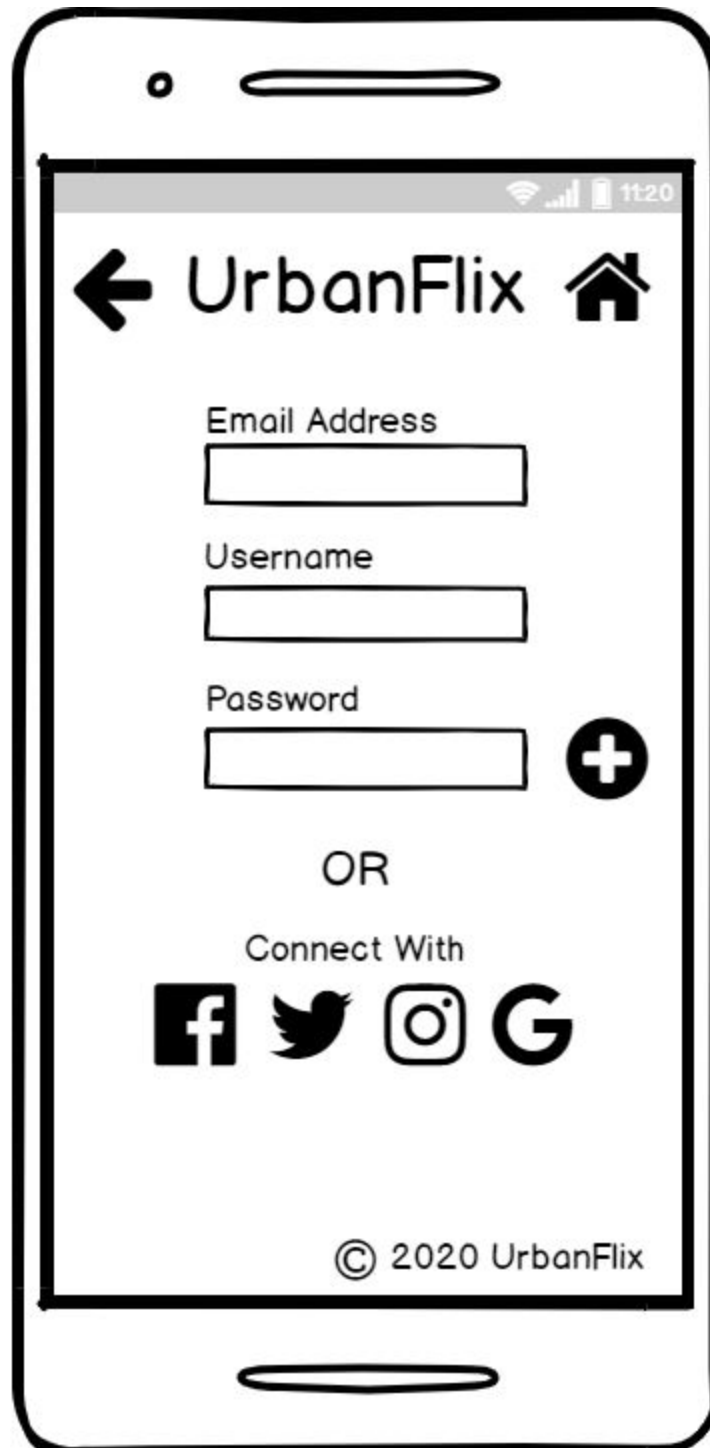
+

Author

Anonymous

+

Account Creation



A mobile application interface for account creation. The screen is framed by a thick black border. At the top, there is a status bar with a grey background showing a Wi-Fi signal, cellular signal, and the time 11:20. Below the status bar is a navigation bar with a back arrow on the left, the text "UrbanFlix" in the center, and a home icon on the right. The main content area contains three input fields: "Email Address", "Username", and "Password". To the right of the "Password" field is a circular button with a plus sign. Below these fields is the text "OR" and "Connect With". Under "Connect With" are four social media icons: Facebook, Twitter, Instagram, and Google+. At the bottom of the screen is a copyright notice: "© 2020 UrbanFlix".

← UrbanFlix 🏠

Email Address

Username

Password

+

OR

Connect With

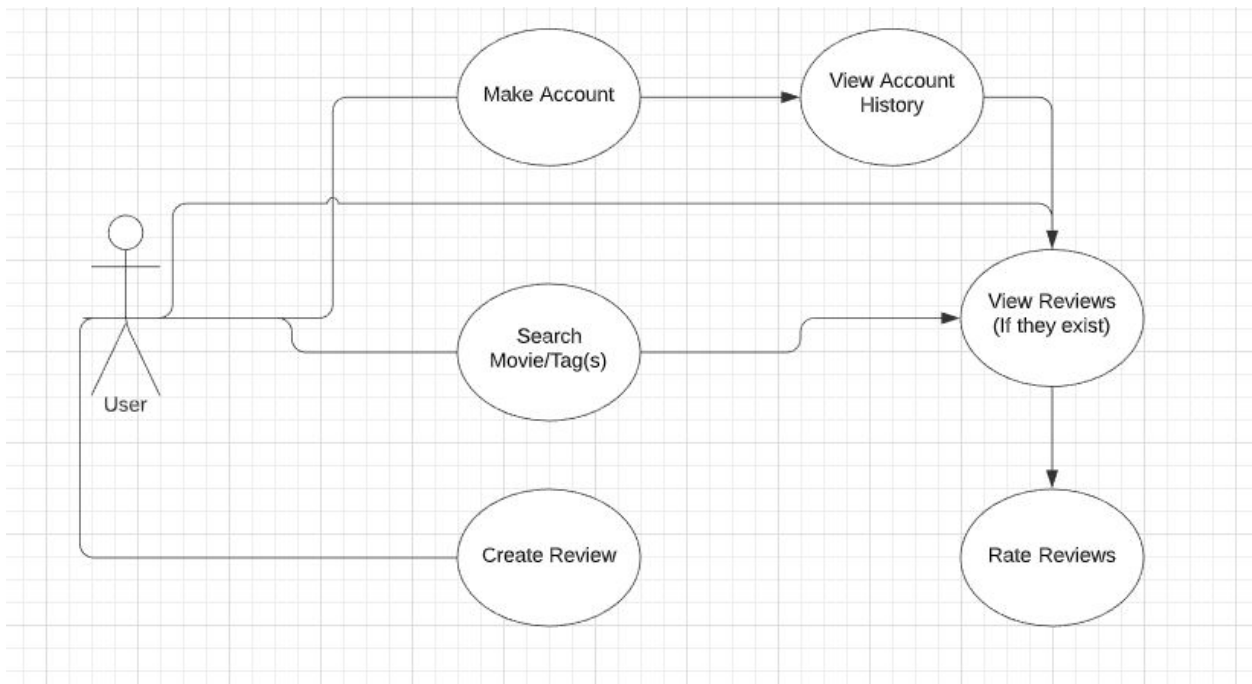
f 🐦 📷 G

© 2020 UrbanFlix

Account Summary (Paid Feature)



6) Use Case Diagram

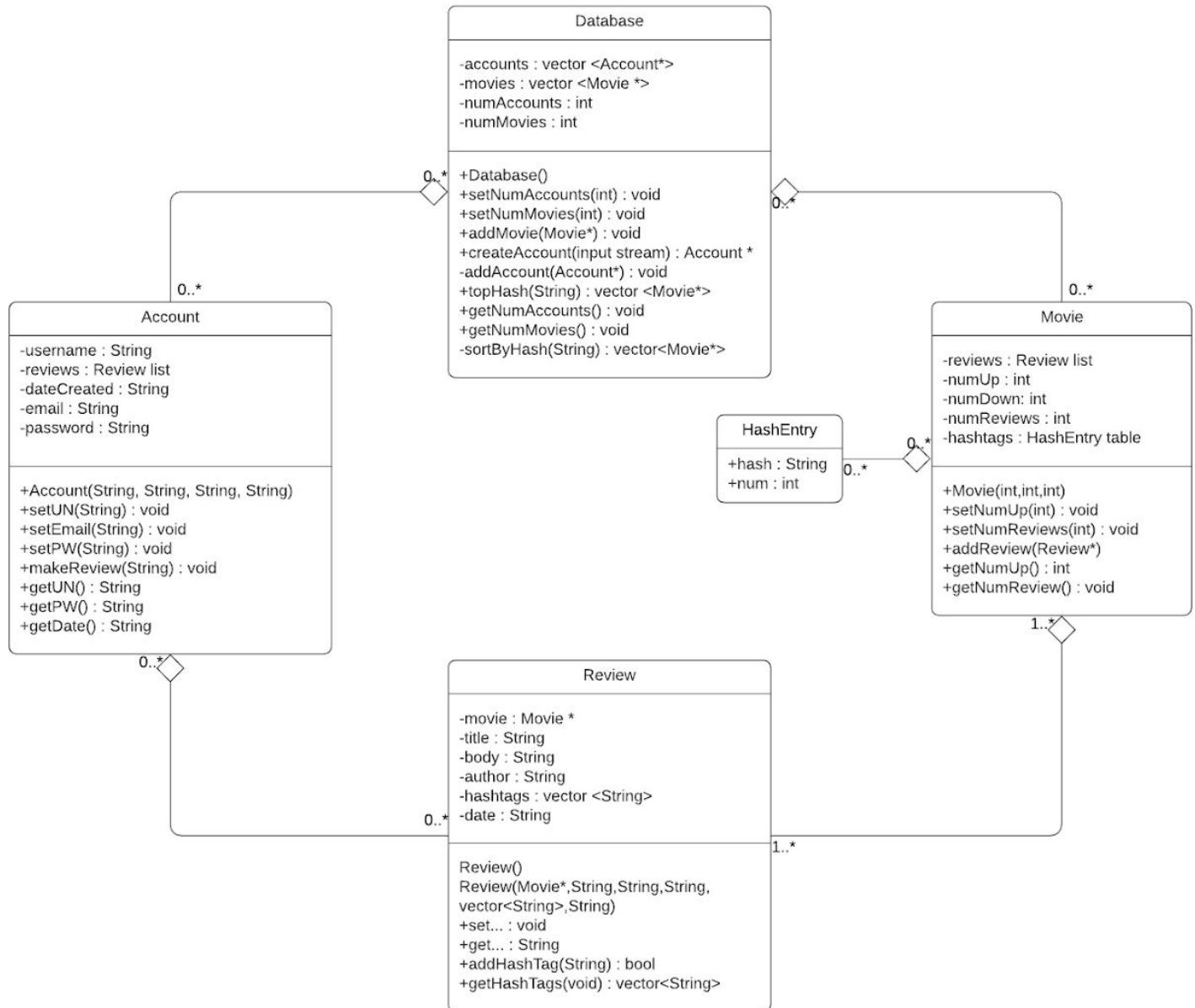


7) Detailed Design

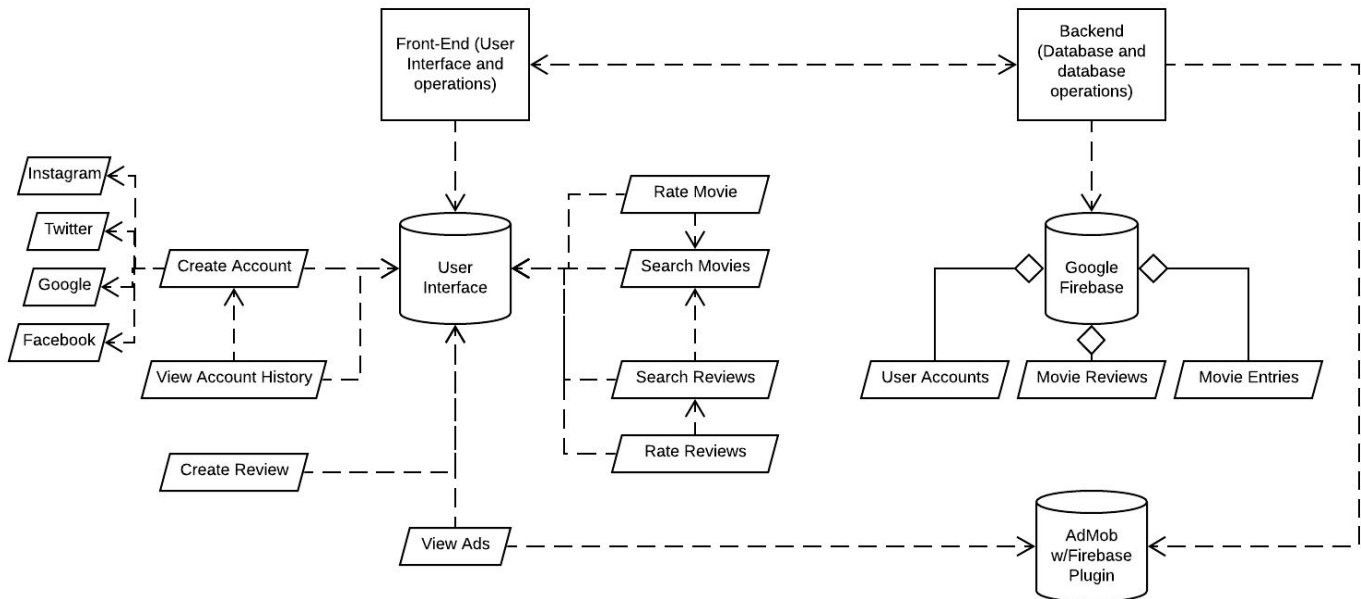
Class Diagram

UrbanFlix Class UML Diagram

Emmanuel Gallegos | February 17, 2020



Systems Diagram



8) Related Work

We drew inspiration for UrbanFlix largely from a similar site/app known as the UrbanDictionary. The idea of UrbanDictionary is unique in that it allows the masses to decide on alternative definitions to words, often slang terms, and possibly creating new slang words entirely. Rather than drawing any definitions from a pre existing database, entries are entirely user generated and their list priorities are determined by user votes. The service UrbanFlix offers is similar to other film and television databases such as IMDB, Reelgood, JustWatch. However, while these sites are curated and draw data from external sources such as Netflix, Hulu, or Disney+, our database is entirely user driven.

When designing our app, we wanted to distinguish our database from other services which have strong influence from special interests (eg film critics, or the film industry). The reviews provided on UrbanFlix will be unadulterated and the most relevant reviews will be determined via popular vote in conjunction with novelty, similar to the way Reddit's sorting algorithms work for subreddits and comments. Reviews that are both new AND popular will float to the top, while old reviews will eventually sink to the bottom, even if they were popular when they were written.

9) Framework/Services/Cloud/Backends

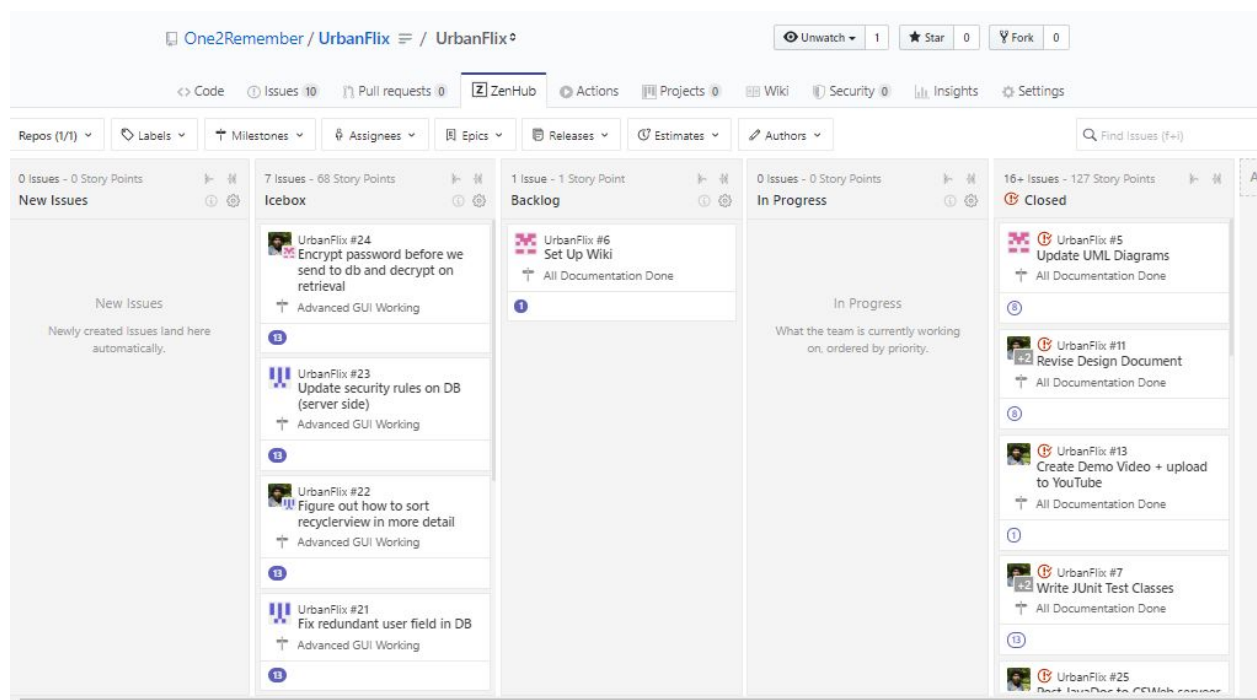
The App will be created using Android Studio using Google Firebase for storing our database of accounts and reviews.

10) Testing

Testing was done using JUnit 4 for MovieReview, User, and SharedPreferencesHelper classes, as those contained and managed the vast majority of our business logic. The other classes operate almost entirely based on GUI manipulation, as our application is fundamentally a content stream service. For PreferencesHelper test class, SPMockBuilder framework was used for creating a mock SharedPreferences file for testing purposes.

We also went through multiple iterations of user testing throughout the process, leading to added functionality we didn't originally anticipate we'd want, like textviews that provided dynamic character limits, as well as consolidating the limited functionality from the movie class into the movie review class.

11) Schedule



12) Dependencies

- Firestore
 - implementation 'com.google.firebase:firebase-firestore:21.4.2'
 - implementation 'com.google.firebase:firebase-analytics:17.2.2'
 - implementation 'com.firebaseui:firebase-ui-firestore:6.2.1'
- Other Firebase/Play Services
 - implementation 'com.google.firebase:firebase-auth:19.3.0'
 - implementation 'com.google.android.gms:play-services-auth:18.0.0'

- FirebaseUI (for authentication)
- implementation 'com.firebaseui:firebase-ui-auth:6.2.1'
- Support Libraries
- implementation 'androidx.appcompat:appcompat:1.1.0'
- implementation 'androidx.vectordrawable:vectordrawable-animated:1.1.0'
- implementation 'androidx.cardview:cardview:1.0.0'
- implementation 'androidx.browser:browser:1.0.0'
- implementation 'com.google.android.material:material:1.1.0'
- implementation 'androidx.multidex:multidex:2.0.1'
- Testing
- testImplementation 'junit:junit:4.12'
- testImplementation 'org.mockito:mockito-core:1.10.19'
- testImplementation 'com.github.IvanShafran:shared-preferences-mock:1.0'
- testImplementation 'org.powermock:powermock:1.6.5'
- testImplementation 'org.powermock:powermock-module-junit4:1.6.5'
- testImplementation 'org.powermock:powermock-api-mockito:1.6.5'
- androidTestImplementation 'androidx.test.ext:junit:1.1.1'
- androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
- androidTestImplementation 'com.android.support.test:rules:1.0.2'
- Other
- implementation fileTree(dir: 'libs', include: ['*.jar'])
- implementation 'androidx.appcompat:appcompat:1.1.0'
- implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
- implementation 'androidx.recyclerview:recyclerview:1.1.0'