Mutibo - A Sample Capstone Project

"Art is the imposing of a pattern on experience, and our aesthetic enjoyment is recognition of the pattern."

- Alfred North Whitehead

Do you really like movies? Do you think you have a knack for solving puzzles? Test your movie trivia knowledge and puzzle solving abilities with Mutibo, a game in which you are asked to identify which movie in a group of movies is the odd one. After making your guess, the game will tell you the correct answer, explain the link between the movies, and let you rate the group of movie! Advanced versions of this game will allow you can compete with friends, create your own puzzle challenges and share high-scores!

Basic Project Requirements

Any potential Capstone project must support multiple users and should leverage services running remotely in the cloud. Each project's specification clearly outlines the app's intended high-level behavior, yet leaves substantial room for individual creativity. Students will therefore need to flesh out many important design and implementation details. Basic requirements for all Capstone MOOC project specifications include:

- Apps must support multiple users via individual user accounts. At least one user facing operation must be available only to authenticated users.
- App implementations must comprise at least one instance of at least two of the following four fundamental Android components: Activity, BroadcastReceiver, Service and ContentProvider.
- Apps must interact with at least one remotely-hosted Java Spring-based service over the network via HTTP.
- At runtime apps must allow users to navigate between at least three different user interface screens.
    - e.g., a hypothetical email reader app might have multiple screens, such as (1) a ListView showing all emails, (2) a detail View showing a single email, (3) a compose view for creating new emails, and (4) a Settings view for providing information about the user's email account.
- Apps must use at least one advanced capability or API from the following list covered in the MoCCA Specialization: multimedia capture, multimedia playback, touch gestures, sensors, or animation. Experienced students are welcome to use other advanced capabilities not covered in the specialization, such as BlueTooth or

Wifi-Direct networking, push notifications, or search. Moreover, projects that are specified by commercial organizations may require the use of additional organization-specific APIs or features not covered in the MoCCA Specialization. In these cases, relevant instructional material will be provided by the specifying organization.

- Apps must support at least one operation that is performed off the UI Thread in one or more background Threads or a Thread pool.
- There may also be additional project-specific requirements (e.g., required use of a particular project-specific API or service).

Basic Functional Description and App Requirements for Mutibo

1. A *Set* is a unit of data that contains four movie titles, optional associated images for each movie, information identifying the one movie that is not like the other three, and accompanying text, explaining the relationship between the three related movies.
2. A *User* should be able to log into the game using an authenticated user account.
3. A single game presents a series of Sets and guesses, until the *User* has made three *Incorrect Guesses*.
4. After viewing a Set, a *User* will be able to rate a Set based on the explanation of the link between the movies. If a Set receives a large number of poor ratings, it can be removed from the game.
5. For each successfully completed Set, the user will get *Points.*

All data (questions, answers, points, etc.) are stored to and retrieved from a web-based service accessible in the cloud.

Bonus Functional Description and App Requirements for Mutibo

If you are able to implement all of this app's basic functional requirements, you may consider adding special advanced features.

1. Users could be allowed to challenge a friend to a sudden death playoff. For example, friends could answer questions turn by turn, and the first person to make a mistake loses.
2. Users could be given progressively difficult questions (e.g., based on other users' previous success with each Set).
3. Users could be given special "power ups" such as the ability to pass a Set, or to get help from a friend them when they are stuck.

4. Users could be allowed to challenge Facebook friends to do various things such as to beat their high score, to help answer a question they are stuck on, etc.
5. The Mutibo Android app could be optimized for the Amazon Appstore and could leverage Amazon's GameCircle API to incorporate leaderboards. For information on how to do this refer to following links:
   https://developer.amazon.com/public/solutions/platforms/android-fireos
   https://developer.amazon.com/public/apis/engage/gamecircle

Implementation Considerations

The Mutibo project specification outlined above is intentionally underspecified to maximize opportunities for student creativity. Students must therefore consider and choose between a number of design and implementation issues and solution alternatives to produce their final product solution. For example, students should consider at least the following issues for this project:

- Will any Movie data be stored on your device? Or will it only be stored in the remote service? Will the app cache information on the local device (e.g., in a Content Provider)?
- How and by whom will Sets be defined?
- What APIs will you use to retrieve movie data for the game (e.g., Images/film data might be pulled from Amazon's Product Advertising API).
- What will the user interface screens look like? How will the user navigate between different screens?
- How, when, and how often will the user enter their user account information? For example, will the user enter this information each time they run the app? Will they specify the information as part of a preference screen?
- What user preferences can the user set? How will the app be informed of changes to these user preferences?
- How will the user navigate between playing the game, looking at leaderboards, asking to view hints, etc.?
- How will the app handle concurrency issues, such as how will periodic updates occur - via server push or app pull? How will search queries and results be efficiently processed? Will the data be pulled from the server in multiple requests or all at one time? Will the server data include full-sized images, or thumbnails plus URLs pointing to the full-sized images?
- Will the app provide extensions and improvements that go beyond the minimum requirements? For example, could the app leverage location information to find

movies playing close by the User and then serve notifications related to those movies? Also, using Maps may require students to have access to an actual device. Also, how will these enhancements affect the rest of the app?

- Does your app really require two or more fundamental Android components? If so, which ones? For example, this app might benefit from using a ContentProvider or from using a background Service that synchronizes local and remote data, only when the device is connected to a WiFi network.