Design Document

1 Design Consideration

1.1 Assumptions

Before the project is completed, questions and activities must be submitted in a CSV document for the application to read and use. The CSV document must contain at least 10 questions and 50 different activities. While creating these documents there are some things that are assumed about the content: A question must have at least 2 no more than 6 answers. An activity must contain at least one category but has to distribute 5 threshold points to its categories – no more no less. There can be no negative value thresholds.

1.2 Constraints

1.2.1 Since multiple devices can use the application, during development, diligent consideration to the user interface must be considered.

1.2.2 Due to the lack of experience from the developer, me. And due to the decision to keep the application self-contained off-line, updating the the aggregation of suggestions can result in too much overhead. This can be detrimental in scalability.

1.3 System Environment

The application that this project will implement is solely runnable on Android devices and should be compatible with Android APT 14 and higher. The application will store the data of the application on the users device, and should be completely runnable off-line

2 Architectural Design

2.1 Overview *

2.1.1 The application runs through the Android Activity class in a normal fashion. The Activity class controls the state of the application from its creation to its termination.

2.1.2.1 The application will load the previous state of the application (if there is one) from when it was last used through a Bundle during the onCreate stage. The Cache should be uploaded as well. The 'main menu' ViewGroup well be set to the canvas.

2.1.3 Once the Activity class is on the OnResume method, the user simply clicks the 'Begin' or 'Start Over' view button to begin answering questions. Or the user may hit the 'Resume' button to continue the previously started state.

2.1.4 The current state of the application will be encapsulated by a Request class. When the user starts a new Request for a list of Suggestion(s),[1]the Request class will reset the current state. The Request class will control the state of the application.

---

* Within the overview, classes that will be used in the system will have their first letter in each word capitalized.

1 The Suggestion class refers to the activities that the application will suggest to the user, the word 'activity' is not used to avoid confusion with the Android Activity class

2.1.4.1 If the 'Begin' or 'Start Over' button are selected, the Request class needs to repopulate its questions list and suggestions set via two CSV files (questions, suggestions) .

2.1.5 The Request Class

2.1.5.1 The Request class controls the state of the user's request for a suggestion.

2.1.5.2 There can only be one Request at a time.

2.1.5.3 The Request class contains as attributes its Questions list, a map that takes all distinct categories as keys and maps a list of Suggestions that contains the key attribute, and a list of Suggestions that have reached its threshold.

2.1.6 The Suggestion Class

2.1.6.1 The Suggestion class is a representation of a potential suggested activity to give to the user.

2.1.6.2 The Suggestion class has as attributes, a hash map of attributes as keys that maps to a discrete threshold, and its text, a boolean value to represent if all thresholds have been met, and a unique ID that will be used by the cache.

2.1.7 The Request class will give the user questions one at a time. Questions and Answers are displayed in the user interface via a QuestionViewGroup. The Request class will remove the question from the list so it is not repeated.

2.1.7.1 The QuestionViewGroup class will consist of the Question's text and an AnswerViewGroup.

2.1.7.2 The AnswerViewGroup will consist of each Answer's text and a touchable point to select the answer.

2.1.8 The Question class

2.1.8.1 The Question class is a representation of the questions that will be asked to the user.

2.1.8.2 The Question class will contain its text, and its answers.

2.1.9 The Answer class

2.1.9.1 The answer class is a representation of a single answer to a question.

2.1.9.2 The answer class contains as attributes its categories and its text.

2.1.6 When an Answer is selected by the user, the Answer will be sent to the Request class.

2.1.7 The Request class will extract the categories and threshold values from the Answer and update each suggestion from the Suggestion map that pertain to the categories[2] of the given answer.

2 A distinct Suggestion will only contain a few attributes, so not all Suggestions in the map need to be updated.

2.1.7.1 When a Suggestion is updated and its threshold has not been met, it needs to be checked to see if by updating, it has met its threshold.

2.1.7.1.1 If the suggestion does reach its threshold. It must check against the cache to see if it is already in it. If it is not in the cache, it can be removed from the list within the map and be put in the Request's reachedThreshold list.

2.1.7.2 Once the Request has finished iterating through the suggestions, it checks to see if its reachedThreshold list has at least 3 suggestions.

2.1.7.2.1 If it has reached 3 suggestions, then the user is given the GiveSuggestionView which displays three suggestions randomly selected from the reachedThreshold list that the user can select. The user should select one so it can be cached.

2.1.8 The Cache class

2.1.8.1 The Cache class represents the data that the application stores onto the users device. The data pertains to the previously selected Suggestions. This is to prevent giving the user the same suggestions that was recently given.

2.1.9 If the minimum amount of Suggestions is not in the reached Threshold list, then the Request class will randomly select a new question to be asked.

2.2 Rationale

The architecture of the application will not rely on any heavy cloud computation. This decision has been made in order to have easy, quick access to the application. Due to the nature of boredom, the application should not expect a reliable Internet connection to use the application. Simply put, if someone is bored , it could be due to the fact that they don't have any Internet access in the first place.

The application will instead use an event driven-based architectural structure to process the data of the system. The Request class is used as the main processing element of the application. Therefor, it acts as the intersection between all elements. This decision to use a single main processing element seemed appropriate because the system is not too complicated in interactions.

2.3 Conceptual (or Logical) View

see attachment 3.2

3 Low Level Design

3.1 Class Diagram

see attachment 3.1

3.2 Sequence Diagram

see attachment 3.2

4 User Interface Design

see attachment 4.1