Software Engineering Large Practical

Documentation

Ramona Comanescu (s1427590)

January 2017

Contents

1 Introduction

This document represents the documentation for Grabble. The resulting product follows the features mentioned in the design document, with some major improvements.

2 Bonus Features

Apart from the basic requirement of Grabble, this application has the following bonus features:

User accounts This feature allows the user to personalize the game with their unique email address. All the user data is saved on a cloud Google Firebase server, meaning that users can use their account on any device and retrieve their previous actions in the game. This goes further from the design document, which proposed that only one account will be allowed for a device. In order to implement this, cloud storage has been used in all instances, as opposed to local storage.

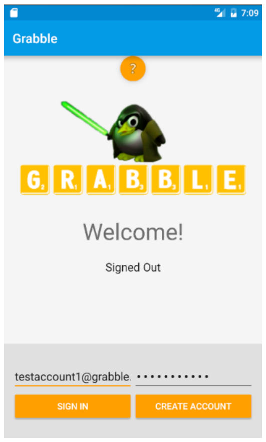
Challenges This feature adds further goals to the game, apart from collecting letters. The user is presented with some challenges to complete in exchange for bonus hints.

Hints Once the user enters two letters in the Word Arena Activity, they can use a bonus hint and ask to be given a completion. Autocomplete is an extra feature which was not mentioned in the design document.

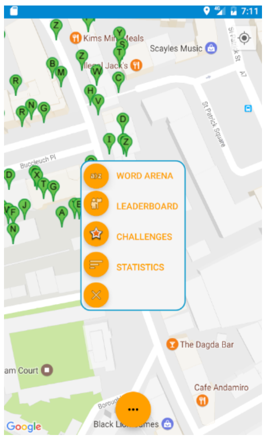
Leaderboard This feature introduces the idea of competition into the game, displaying other users' scores against the current player.

Statistics This feature allows the user to visualize their top scoring words.

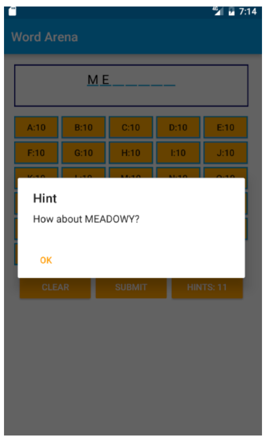
3 Activities overview



(a) Welcome



(b) Campus Map

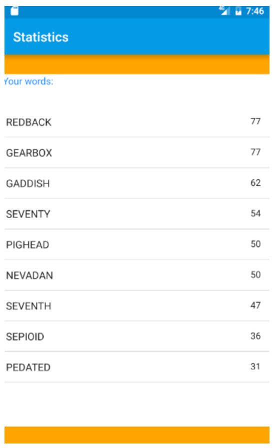


Hint

How about MEADOWY?

OK

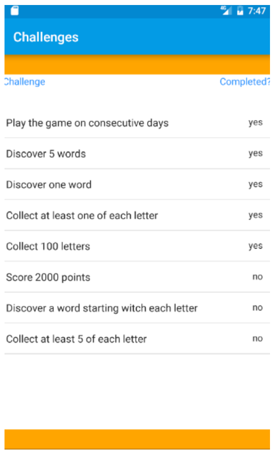
(c) Word Arena



Your words:

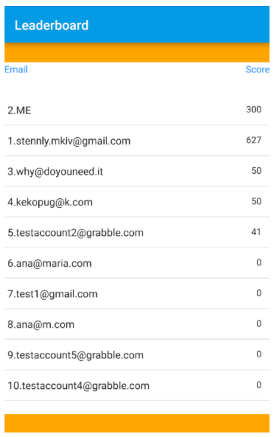
REDBACK	77
GEARBOX	77
GADDISH	62
SEVENTY	54
PIGHEAD	50
NEVADAN	50
SEVENTH	47
SEPIOID	36
PEDATED	31

(d) Statistics



Challenge	Completed?
Play the game on consecutive days	yes
Discover 5 words	yes
Discover one word	yes
Collect at least one of each letter	yes
Collect 100 letters	yes
Score 2000 points	no
Discover a word starting witch each letter	no
Collect at least 5 of each letter	no

(e) Challenges



Email	Score
2.ME	300
1.stennly.mkiv@gmail.com	627
3.why@doyouneed.it	50
4.kekopug@k.com	50
5.testaccount2@grabble.com	41
6.ana@maria.com	0
7.test1@gmail.com	0
8.ana@m.com	0
9.testaccount5@grabble.com	0
10.testaccount4@grabble.com	0

(f) Leaderboard

4 Algorithms and data structures

Download and parsing of the daily Grabble letter map from the server This is realized by using an asynchronous task to download the file from the website. The resulting file is then parsed by a custom written *kXML* parser which uses an *XMLPullParser* to get markers' location and add them to the map.

Efficient look up of words in the Grabble dictionary

The Grabble dictionary is stored locally. All words from the file are read line by line and stored in a *prefix tree(Trie)*. This data structure was chosen because it offers constant look up time (proportional to the length of the word) and has a space complexity of $\mathcal{O}(N * M)$ where N is the number of words and M is the highest length of the string(7).
As opposed to a hash map, a trie also allows for prefix search, which is used to provide autocompletion as part as the Hints game feature. (Given a prefix, it can return all words in the dictionary starting with that prefix, in a depth-first search manner.)

Efficient detection of the letters which can be grabbed

A method provided by Location has been used, allowing a distance of 25m to account for GPS inaccuracy.

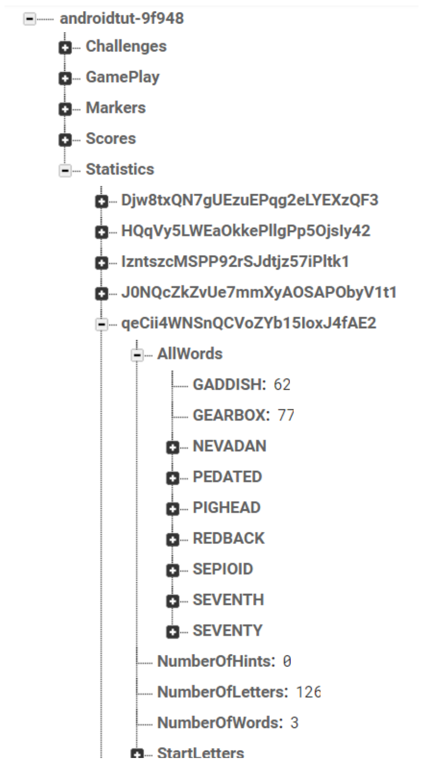
```
Location.distanceBetween(userLocation.getLatitude(),
userLocation.getLongitude(), letterPosition.latitude,
letterPosition.longitude, results)
```

5 Database structure

A NoSQL database is provided by the Firebase API. The resulting json database file is included in the **server/** folder, as of January 16. In order to provide good look up time, the level of nesting has been kept low. Classes that can be directly stored and retrieved from the database have been created (Score, Challenge), taking advantage of the object-oriented database.
There is a main users database, which provides a hash key for each user.
The game database contains:

- Challenges - contains user keys and on the next level of nesting stores a list of Challenge objects which contain description and status (complete/not complete)
- Score - for each user key, store a list of Score objects (user email and score value)
- Markers - for each user key store the list of markers which have not been yet collected.
- Gameplay - for each user key, store a list with 26 letters and the count of each letter collected by the user. Also store the last time game was played.
- Statistics - for each user key, store word history, total number of hints, number of letters, number of words.

Figure 1: Database structure



```
androidtut-9f948
  Challenges
  GamePlay
  Markers
  Scores
  Statistics
    Djw8txQN7gUEzuEPqg2eLYEXzQF3
    HQqVy5LWEaOkkePllgPp5OjsIy42
    IzntszcMSPP92rSJdtjz57iPltk1
    J0NQcZkZvUe7mmXyAOSAPObyV1t1
    qeCii4WNSnQCVoZYb15IoxJ4fAE2
      AllWords
        GADDISH: 62
        GEARBOX: 77
        NEVADAN
        PEDATED
        PIGHEAD
        REDBACK
        SEPIOID
        SEVENTH
        SEVENTY
      NumberOfHints: 0
      NumberOfLetters: 126
      NumberOfWords: 3
      StartLetters
```

6 Triggering events

The implementation makes use of Listeners for triggering various actions. In order to address energy concerns, all listeners are unregistered when onPause() is called.

Location listener A location listener is used for detecting user's location in Campus Map. When location no longer needs to be polled, this listener is released.

Connectivity Receive Listener Activities that need to read data from a server or online database implement this interface, warning the users when Internet is turned off.

Firebase Value Event Listeners ValueEventListeners are used for getting latest values from the database. Some of them only request the value once, while others constantly poll for changes.

Challenge Manager A singleton class manages all challenges. All the activities involved in letter collection or word creation alert this Manager when a new event is triggered, in order to check if the user should be given an award. All listeners for this class are again released when not in use.

7 Testing

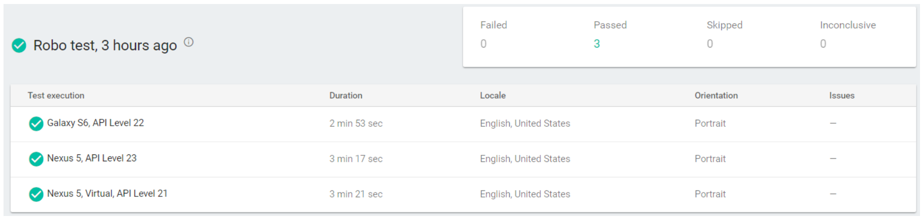
7.1 Demo credentials

A demo account should already be input when starting the game (email: *testaccount1@grabble.com* and password: *testaccount*). This account has enough collected letters so that the main game features can be tested. New accounts can also be created.

7.2 Automated testing

Unit tests have been written to test utility classes(such as the prefix tree and markers file download). Basic UI Instrumented tests has been written, testing sign in in with a default account and accessing screens. The Firebase testing tool Robo test has been used to test the game on different devices, testing the UI elements.

Figure 2: Firebase Test Lab



✓ Robo test, 3 hours ago

Failed	Passed	Skipped	Inconclusive
0	3	0	0

Test execution	Duration	Locale	Orientation	Issues
✓ Galaxy S6, API Level 22	2 min 53 sec	English, United States	Portrait	–
✓ Nexus 5, API Level 23	3 min 17 sec	English, United States	Portrait	–
✓ Nexus 5, Virtual, API Level 21	3 min 21 sec	English, United States	Portrait	–

7.3 Physical and Virtual testing

A version of the game has been released to 2 non-informatics testers on a period of one week, to assess how user friendly the game is and their feedback has been taken into account. Minor bugs such as layouts not rendering correctly on a smaller screen have been fixed.
On the developer's part, the game has been tested both using an emulator(mostly for being able to collect letters from different locations) and multiple physical devices.
Test cases that have been manually tested:

1. Creating a new account
2. Logging in
3. Signing out and logging in with a different account
4. Viewing instructions

5. Opening the Campus Activity with no Internet connection and then turning on the Internet connection. Result: markers are still loaded when Internet is detected
6. Collecting letters, exiting the game and returning to collect remaining letters
7. Opening the Campus Activity with GPS turned off. A prompt is displayed to turn on GPS.
8. Collecting letters within 25m range.
9. Collecting letters which are out of range.
10. Challenge detection has been tested by long term playing. All challenges are successfully triggered for the right event. For example, logging in on consecutive days, collecting 100 letters, achieving a score of 2000, discovering 5 letters, collecting one of each letter
11. Creating a new word that is present in the dictionary
12. Trying to submit word that is not in the dictionary
13. Asking for hints when prefix exists in dictionary
14. Asking for hints when prefix does not exist in dictionary
15. Viewing completed Challenges
16. Viewing Leaderboard
17. Viewing word Statistics
18. Offline database access: Firebase has proven robust in this case, updating values as soon as a connection becomes available

8 Version control

Git has been used for version control, proving useful for backing up the project and reverting to earlier versions. The private Github repository can be found at this address. User *sgilmore* has been added to the repository.

9 Conclusion

- All parts of the design document have been implemented and a few extra features have been added(multiple user accounts and autocomplete hints).
- Various game scenarios have been tested for an extensive period of time.

- Code has been documented and software design practices has been followed.
- Feedback from people testing the game has been and they reported enjoying the experience.