Documentation

Software Engineering Large Practical

Stylianos (Stelios) Milisavljevic

s1509375

Contents

Introduction

The documentation describes the implementation of Songle. Following the design document, all the features described have been realised in the final app.

Bonus Features Implemented

Point System

The implementation uses a point system that rewards the player for exploring George Square Campus by increasing the score for each lyric and song found. A player that chooses not to travel much will get a much lower score than a player that walks a lot and collects many lyrics. In addition, the scoring system is different for each difficulty and has been revised to reflect the feedback received for the design document.

Distance travelled

The distance travelled is being tracked and displayed to the user. If the user guesses the song correctly, bonus points are earned related to the distance travelled and the difficulty chosen.

Statistics

The implementation of statistics, allows the player to see the total distance travelled, the highest score, the total number of songs found and the guessing accuracy. This provides the player with a clear way to see the progression made while playing the game and leads to competitive play between the userbase of the app.

Replayability

The game is worth playing more than once.

- Separate difficulties give different point earnings. A user playing a harder difficulty gets much more points from the scoring actions than a user playing an easier one.
- Order of the songs presented to the user is random. Forces the user to collect lyrics again if starting a new game, instead of memorizing the order of the songs and writing the titles right away. Further applied by having a deduction of points for each wrong guess, so that the user doesn't try every song found in previous plays.
- Once a song is found, it is discarded from the list of songs that can be used when deciding randomly the next song. When a new game is started, all songs are added back to the list.
- When choosing to skip a song, the song is not revealed. If the user decides to play the game again, the answer is not spoiled by skipping it in a previous play.

Reveal a lyric

Option of revealing a lyric from wherever the player is, but for the cost of some points.

Skip a song

When user is stuck, there is an option to skip the current song without revealing its title. To prevent the user from using the skip function too frequently, the points of finding the song are instead deducted. The distance travelled in this song gets discarded, losing any bonus points the user would have gotten.

Continue

The game has the functionality to continue from where it was left off saving all the progress made when leaving.

Help Screen

The help screen includes all the information required for the user to understand how to play the game.

Connection settings

The user can choose whether downloads of the remote files can be done using data or only Wi-Fi connections.

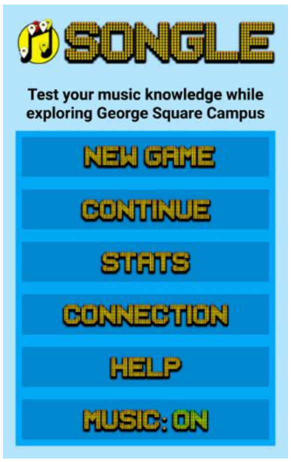
Background Music

The game has background music that plays through all activities and can be disabled. (Music used is Royalty Free)

Close Enough Submissions (New)

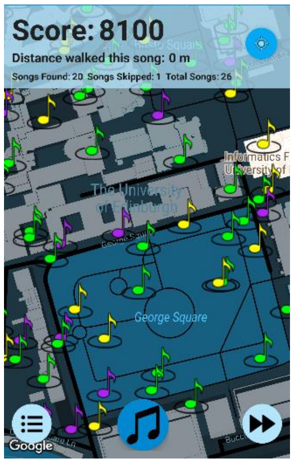
When a user submits his guess of the song title, the game will allow submissions that are close enough but not exact to be passed as correct. This bonus feature was added so that, minor spelling mistakes or missing commas are accepted.

Views Overview

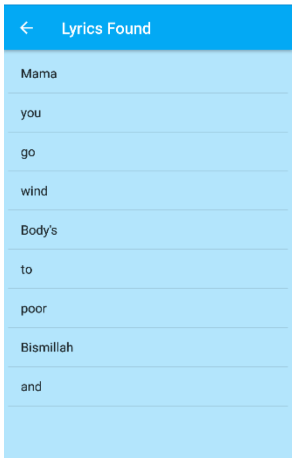


Test your music knowledge while exploring George Square Campus

MAIN SCREEN

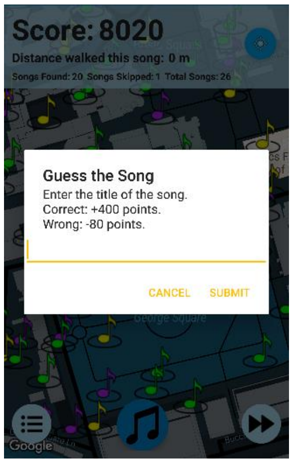

Score: 8100
Distance walked this song: 0 m
Songs Found: 20 Songs Skipped: 1 Total Songs: 26

MAP SCREEN



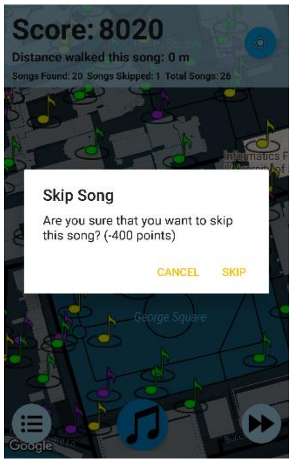
Mama
you
go
wind
Body's
to
poor
Bismillah
and

LYRICS FOUND LIST


Score: 8020
Distance walked this song: 0 m
Songs Found: 20 Songs Skipped: 1 Total Songs: 26

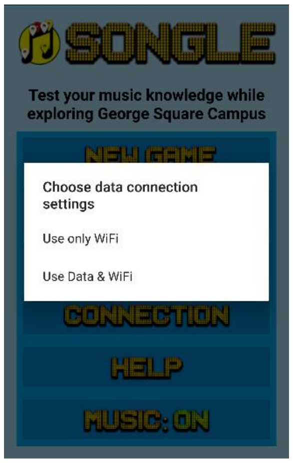
Guess the Song
Enter the title of the song.
Correct: +400 points.
Wrong: -80 points.

GUESS THE SONG DIALOG


Score: 8020
Distance walked this song: 0 m
Songs Found: 20 Songs Skipped: 1 Total Songs: 26

Skip Song
Are you sure that you want to skip this song? (-400 points)

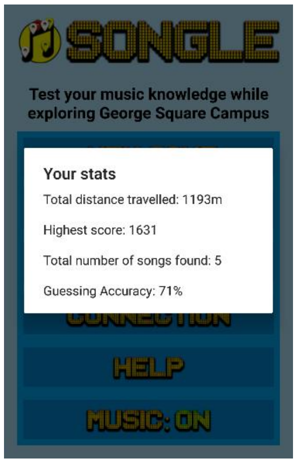
SKIP SONG DIALOG



Test your music knowledge while exploring George Square Campus

Choose data connection settings
Use only WiFi
Use Data & WiFi

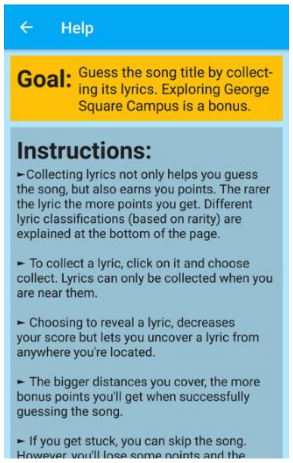
SET DATA CONNECTION



Test your music knowledge while exploring George Square Campus

Your stats
Total distance travelled: 1193m
Highest score: 1631
Total number of songs found: 5
Guessing Accuracy: 71%

STATISTICS



Goal: Guess the song title by collecting its lyrics. Exploring George Square Campus is a bonus.

Instructions:
►Collecting lyrics not only helps you guess the song, but also earns you points. The rarer the lyric the more points you get. Different lyric classifications (based on rarity) are explained at the bottom of the page.

► To collect a lyric, click on it and choose collect. Lyrics can only be collected when you are near them.

► Choosing to reveal a lyric, decreases your score but lets you uncover a lyric from anywhere you're located.

► The bigger distances you cover, the more bonus points you'll get when successfully guessing the song.

► If you get stuck, you can skip the song. However, you'll lose some points and the

HELP SCREEN

Algorithms and data structures

Song class

Each Song instance encapsulates the details of a song (number, artist, title and YouTube link). In addtition, this class includes some static methods that concern songs like generating a new song number that hasn't been used in the session yet (method described in Song Submission subsection).

Placemark class

Each lyric that can be collected is displayed as a marker on the map. The placemark class defines those different markers by their lyric name (<line number>:<word number>), the difficulty description, latitude and longitude location.

Online resources download and parsing

The downloading and parsing of the XML-based Songle song list from the server is handled by an asynchronous task. This task is triggered by a network receiver that checks if there is an active connection that is also allowed in the settings (WiFi only or both Data and WiFi).

The parsing is done by an XmlPullParser that captures the structure of the specific structure of the song list. The songs are then saved as a list of Song instances.

Similarly, the XML-based placemarks list is parsed and added automatically to the map once the user enters the Map Activity. The lyrics being a text file, aren't handled by the XmlPullParser. They are parsed using a stream Scanner and split into a list of lines, with each line further split into an array of words by using regular expression pattern matching.

Collection of words

For a user to collect a word, first the location on the device must be enabled (a Toast message is shown) and the permissions to access the location must be granted (if not granted, the user will be asked again to grant them).

Then the user need to press on the lyric shown in the map, as the collection is handled using the onMarkerClickListener interface provided by the Google Maps API. A dialog will be shown with a choice to either reveal or collect the lyric. Revealing the lyric deducts some points from the user's score, but adds the lyric to the set of lyrics found without checking its distance. However, when choosing to collect the lyric, the distance between the last location of the device and the lyric is calculated using Location.distanceBetween() and is checked against a predetermined threshold. If the distance is close enough it adds the lyric to the set of lyrics found.

Music Service

The background music is implemented as a Service, which creates a MediaPlayer instance with the music set to loop. It then provides options to pause the music (when the user switches it off) by saving the current position of the track. Resuming the music, seeks the track back to the position saved.

This service is started by at the creation of the MainActivity if the option was saved as switched on. Otherwise, if the option was turned off in the previous session, the service is started when the user switches it back on, so that it doesn't waste resources.

Song Submission

When there is an attempt to guess the song, the given is string is compared with both the whole title of the song and a simplified version that removes bracketed phrases e.g. It's The End Of The World As We Know It (And I Feel Fine). For the comparison measurement, the Levenshtein distance is used so that submissions that are close enough to the correct answer (determined by a threshold), are accepted as correct. This implementation allows minor spelling mistakes to be ignored.

After the submission being correct, the game checks whether the game is completed, by comparing the size of the song list with the size of the set of the songs used. If not, the next song is selected randomly from the songs not played yet. This is done using a loop that chooses a random song number in the song list range, until it finds one that it is not present in the songs used set.

Finally, the asynchronous task that downloads the new song's lyrics and placemarks is called and the map is updated accordingly.

Acknowledgements

Some code used in the project was based on different online sources.

Description	**Source**
Background Music Service	CodeProject
Levenshtein distance	Wikipedia
Random number generation in a specific range (for new song number)	StackOverflow
Watching Music Video on Youtube	StackOverflow
Input text dialog	StackOverflow
Royalty Free Music	YouTube

Persistent Storage

The progress and the settings of a Songle player, are saved between sessions using SharedPreferences. These values are retrieved and then stored into a variable, thus reducing I/O actions. When these values are changed, the corresponding SharedPreference is updated accordingly.
The SharedPreferences divided in “global” and “session”.
Global SharedPreferences store the stats and settings values that should be retained and not reset after starting a new game.

Global Preference Key	**Description of the value saved**
NETWORK_PREF_KEY	The connection preference of the user
TOTAL_DIST_KEY	Total distance travelled while playing
HIGHSCORE_KEY	Highest score achieved across all games
TOTAL_SONGS_FOUND_KEY	Number of times that the user guessed the song correctly
TOTAL_GUESS_ATTEMPTS_KEY	Number of times the user attempted to guess the song
IS_MUSIC_ON_KEY	Used to check the user preference for the background music

Session SharedPreferences store the values that concern the current game “session” and when starting a new game are overwritten.

Session Preference Key	**Description of the value saved**
SONG_KEY	Number of the current song to guess
DIFFICULTY_KEY	Difficulty of the current play
LYRICS_FOUND_KEY	Lyrics found for the current song
POINTS_KEY	Score
CURRENT_SONGS_FOUND_KEY	Number of times the user guessed the song correctly
CURRENT_SONGS_SKIPPED_KEY	Number of times the user skipped the song
SONGS_USED_KEY	List of the Songs used (found and skipped) in the current session

Testing

Automated Testing

Unit tests

Unit testing was used for individual pieces of code to determine whether they are fit for use.

Unit Test Name	Checks
findWordsFoundTest	The functionality of Song.findWordsFound(lyricsFound,allWords) • correct lyric list is returned • null lyricsFound behaviour • null allWords behaviour
GenerateNewSongNumTest	The functionality of Song.generateNewSongNum(total,songsUsed) • correct generated song number • behaviour when the total is less than the songsUsed size
GetSongTest	The functionality of Song.getSong(songNumber,songList) • returns the correct song • null is returned when a song is not in the song list • handling null argument given
LyricToWordTest	The functionality of Song.lyricToWord(lyric,allWords). Note: lyrics found are stored in the "lineNumber:wordNumber", and when they have to be presented to the user, this function retrieves the correct word from the song text. • correct lyric is returned • handling unexpected lyric • null allWords behaviour
StringComparisonTest	The String comparison used in the submission of a song for correct, incorrect and similar enough, pairs of strings. • Levenshtein distance function • roughlyEquals function

Instrumented Tests

Instrumented tests, check the application on a physical device or emulator. This allows checking behaviour that relies on the Android API and cannot be captured by unit tests. They use the espresso testing framework to simulate user interactions within Songle.

Unit Test Name	Requirements	Checks
CompleteGameTest	- active internet connection - location enabled - permissions for location enabled	User is redirected to the MainActivity with no continue option and the global stats are updated correctly, when completing the game.
CorrectSubmissionTest	- active internet connection - location enabled - permissions for location enabled	When submitting a correct song: • Text view of found songs increases • Song is changed • Points are increased
SkipSongTest	- active internet connection - location enabled - permissions for location enabled	When skipping a song: • Text view of skipped songs updated • Song is changed • Points are deducted
WrongSubmissionTest	- active internet connection - location enabled - permissions for location enabled	When submitting a wrong song: • Text view of found songs remains the same • Song is same • Points are reduced
HelpActivityTest		User is directed to the HelpActivity and when the navigation up button is pressed, it goes back to main activity.
LyricsFoundTest		Given set of lyrics are shown in their correct positions.
NewGameFromCleanTest	- active internet connection - location enabled - permissions for location enabled	Creates a new game from a fresh start. Checks that the continue button is not present when there is no progress and that it shows up after creating a new game.
OfflineTest	- internet connection turned off	If a new game can be started while offline. (it shouldn't)
WifiOnlyTest	- active data connection - WiFi disabled - permissions for location enabled	The app doesn't start a new game when there is no WiFi and the data connection is enabled, but in the connection settings allows only connections over WiFi

If permissions for location not enabled and required, then the tester can manually give permission when prompted while running the test.

Manual Testing

Several different tests were performed using a physical device as well as an emulator. Apart from the Automatic Tests described above, these tests were made:

- Zoom on location button use
- Collecting a lyric
- Revealing a lyric
- Viewing found lyrics
- Denying permission access for location
- Background music correctly playing through all the activities
- Background music being stopped and resumed
- New songs are added from an updated remote xml
- Different placemark types for each of the difficulties
- Proper viewing in screen sizes 4 inches and above

User Testing

Songle was given to several friends that volunteered to play the game and provide feedback and find any unexpected bugs. On a whole, the user interface proved to be friendly and simple enough for anyone to play right away.

A feature that was added from the feedback, is showing the total number of songs in the maps activity, so the player knows how many songs are there left for him/her to find and if the number of songs is increased.

Also, a user recommended that genres of music are added as a choice when starting a new game, as most of the songs used in the game weren't known to him. This feature couldn't be implemented with the current xml song list provided, but if a genre attribute was added for each song, then it would be feasible.

All in all, the users enjoyed playing Songle.

Version Control

The project was managed using Git version control throughout the whole process. It was used to compare the working version with older versions of the files. In addition, being used as a backup, with each version possible to be reverted at any time. The private GitHub repository used has user **sgilmore** as a collaborator.

Conclusion

Everything envisioned in the design document was implemented and tested for bug free use. Songle is now ready to be used by the public.