SOFTWARE ENGINEERING LARGE PRACTICAL - PROJECT DOCUMENTATION

Boyan Yotov – s1509922

OVERVIEW

1. Game Flow

- The **MainActivity** class and the **DownloadDataTimestamp** class maintain a check for data missing and data update. The logo of Songle is set as only layout for the main activity, working as a loading screen.
- When all data update is finished the Main Menu is activated. Its only task is to separate the options menu from the game and allow the start of a new game, while one already exists.
- Proceeding to play the game, the **ChooseDifficulty** and **ChooseLevel** activities allow for a choice of 5 difficulty sets containing all of the playable levels.
- The **MapsActivity** is the activity where all of the gameplay is achieved. It is separated into three functional blocks – level menu, scroll-bar lyrics display and Google Maps fragment.

2. Data Structures

2.1. **sharedPreferences**:

- **GameData –** keeps information about all the songs and their timestamps.
- **Settings** – stores Boolean values about the chosen settings such as if vibration is on or if the dark theme is turned on.
- **Progress** – stores if a game has already started and what levels have been passed, and the difficulty to play
- **Level –** stores information about the song currently played

2.2 **Song**:

- Holds information about the number, title, artist and the youtube.com link.

- All songs are downloaded in a **List** of songs.

2.3 **Lyrics**:

- The lyrics are saved in three variables. One is a **HashMap<String1, String2>**, where String2 is the word which it represents and String1 is key created from concatenating the row and column position in the text (i.e. "row:col ").

- Other two variables are considered helpers. **Lyrics_rows** hold the number of rows in the text lyrics and a **Map<row,value>** keeps the word count on each of the rows.

2.4 **Placemarks**:

-Similar to the songs, the placemarks are kept in a List of Placemarks, where each Placemarks keeps the whole information from the server.

2.5 **IconStyles**:

- The Icon Styles data structure was created to keep the different sizes and image **pngs** for the different classification of Placemarks, however, they are not used, even if their download and save is implemented. .

3. Algorithms

3.1 **Download and Parse:**

- For all available online data a separate download task with a parser was created.

- The downloads are similar **–** using **Async Processes** and **onPostExecute** tasks.

- The parser for the lyrics used a simple string splitting, unlike the **Xml parsers** used by the other data

3.2 **Level sets generations:**

- The levels are separated in five difficulty groups and the Song's number **(mod 5)** was used to split the songs in different category.

3.3 **Guess song:**

- To guess a song the input String in **lowercase** is required to be equivalent to the song's title in **lowercase.**

3.4 **Hints generation**:

- Due to the fact that the song guessing is pretty though, hints were implemented, hardcoded in cases for the five difficulty categories.

- The player receives **Hint Points,** whenever he finishes a level. The cost for the hints is deduced from the **Hint Points**.

- A singleton class was used to implement the generator.

- The hardest difficulty "ImpossiBro" does not allow for any hints to be purchased.

3.5 **Collect markers:**

- The function **CalculationByDistance(LatLng,LatLng)** calculates the distance between the picked marker and the player's location.

- If the distance is less than or equal to **0.02** the marker is recognized as obtained and then it is removed.

4. Additional features

4.1 **Design –** the design was refurbished and new background and button colors were designed. Also, instead of having buttons to increase and decrease difficulty, now there are arrows and after the hardest difficulty the rotation is reset and vice-versa.

4.2 **Click to collect marker –** If markers are close enough the player can pick them up.

4.3 **Slide-Up panel –** The slide up panel to show the map was not implemented. Instead three blocks of layouts interchange in between to accommodate the three units of functionality in the **MapsActivity**.

5. Incomplete tasks

5.1 **Save current level -** A save for the current level was supposed to be implemented. Currently, every time a player goes back to choose a level, the data is renewed and old data deleted. However, when a player mistakenly click the "Go back to levels" button called "Levels" an alert message is displayed.

5.2 **GameDataManager –** In the end it was decided that the singleton class to hold and manage all data was making the code too heavy, so it was removed.

5.3 **Randomise songs in level sets –** It was intended to have a randomized stacking of all songs in the difficulty sets depending on a time value upon first opening of the app. A simpler approach was taken.

6. Acknoledgements

- https://developer.android.com/studio/intro/index.html - A lot of useful guides to create most of the design, load and upload.
- https://stackoverflow.com/questions/14394366/find-distance-between-two-points-on-map-using-google-map-api-v2 - Used for the CalculationByDistance() functionality.