ECS506U Software Engineering
Group Project 2015

Problem Definition

Version 1.0

9 December 2015

Prof. Martin Neil

TABLE OF CONTENTS

1 Introduction

1. The aim of the 2016 software engineering group project (referred to as simply *the project* in the remainder of this document) is for your group to create and supply a stand-alone Java based implementation of a music player called "Plookify".
2. It is important to note that the special requirements of the project are such that there is no point in students attempting to cheat by copying systems and websites online. The special requirements force groups to design their own solid object oriented code that they fully understand.
3. The requirements are organised into:
   - Functional requirements (Section 2)
   - Data requirements (Section 3)
   - Non-functional requirements (Section 4)
4. We have designed the requirements in such a way that you must apply solid object-oriented design principles in order to complete the project successfully.
5. The functional and data requirements for the system are final and as prescribed and thus not open to elucidation, negotiation or refinement, except at your own initiative.
6. You should read this document in conjunction with "Software Engineering Group Project 2016: Guidelines".

2 Functional requirements

The Plookify system must provide all or some of following modules, depending on group composition:

- Account management
- Track player
- Radio station generator
- Playlist manager
- Social manager
- Downloader

Each module will need to be developed separately and meet the following functional requirements:

1. Present a graphical customer interface to the customer to allow them to carry out all operations.
2. Allow tracks (records) and data to be added/edited and deleted from playlists.
3. Correctly implement the specific data requirements associated with the module.
4. Save and retrieve records to/from a database table(s).

5. Execute specific operations on the data associated with the module.
6. Provide a method of customer authentication.

Plookify, as a whole, will be integrated by combining all of these independently developed modules together and will meet the following functional requirements:

1. Present a single GUI for the system as a whole.
2. Deliver a single authentication mechanism.
3. Preserve the data integrity of the system by ensuring that changes made in one module are propagated to all others (e.g. deleting an account will delete the playlists associated with that account).

3 Module data requirements

The data requirements for each module are set out below. They delineate those data requirements specific to the module in question and those that belong elsewhere but which a given module must interact with.

Because the system is tightly integrated students are advised to understand requirements from ALL modules in order to best implement those relating to their module.

There are six modules specified in priority order. Groups with four members will do modules marked with an (A), those of size five will do those marked with (A) and (B) and those of size six will do (A), (B) and (C).

A database of two hundred tracks will be required, as a minimum, and classified by artist name, genre, track name and track length.

The delivered system should come with five customers already set up and with their associated data loaded.

3.1 Account management (A)

1. A customer is a member of the public who holds an account with Plookify.
2. Each customer must register their full name, address and contact details with Plookify when they open an account.
3. Each customer must register for either a free service or a paid subscription, which can either be paid for by PayPal or by credit card.
4. A customer should be able to close an account.
5. A customer can use the service on up to 5 devices associated with the account. A device can be a mobile phone, games console or PC/Mac/Tablet.
6. New devices can be added at any time but a device can only be replaced once per month.

7. Should a subscription payment be missed the account will remain open but marked as “payment outstanding” and the customer informed when they use the service. The account will then revert to a free account.
8. The subscription service will allow the customer to use all functionality. The free service will not allow access to Social or Radio features.

3.2 Track player (A)

1. Any customer can search the track database for a track by artist, genre or song name.
2. The tracks on the search results can be added individually or in their entirety to the “now playing” playlist.
3. Tracks listed on the now playing list can be removed.
4. The track player will allow the customer to view the now playing track list.
5. During the playing of a track the customer will be able to pause/resume it in real time, and restart playing.
6. A track can be fast forwarded or rewound at any time during playback and will play from the point it was move to.
7. A customer can choose to play the track at any start point, so if a track is 3 minutes long the customer could start it 1 minute and 5 seconds into its elapsed duration.
8. The duration of the track will be displayed alongside the elapsed play time.
9. Plookify must play an actual mp3 file or provide a visual imitation of one playing.

3.3 Radio station generator (A)

1. A subscription customer will be able to search for an artist from the track database and Plookify will randomly choose similar artists, belonging to the same genre category, to add to a radio channel/playlist.
2. For those artists chosen at random, tracks belonging to them will also be chosen at random by Plookify, for inclusion on the channel/playlist.
3. There will be at most 10 tracks in a radio channel.
4. Radio channels once generated can be saved as playlists.
5. The customer will be able to browse a list of songs in the radio channel/playlist.

3.4 Playlist manager (A)

1. A customer can create any number of playlists.
2. The customer can search the track database for an artist, display their tracks and add them to a new or existing playlist.
3. Each playlist will be composed of tracks which should be organised to play in sequential order.

4. Playlists can be renamed and tracks removed and added at any point in the sequence.
5. The customer will be able to browse a list of songs on the playlist.
6. There will be two different types of playlist: Friend and Private.
7. Friend playlists are playable by friends and private ones remain useable by the customer who created them only.
8. A playlist can be added to the "now playing" track list.

3.5 Social manager (B)

1. If the customer has a paid subscription The customer can add other customers as friends provided the friend is a paying subscriber.
2. The customer can mark themselves as discoverable so that friends can find them or the customer can remain as private.
3. To add someone as a friend a private message is sent to another customer, who can accept or decline.
4. Upon acceptance of a friend invite they are added as a friend to the customer's 'friend list'.
5. A customer can remove a friend from their "friends list" at any time.
6. Subscribed users must be able to see the tracks in their friends' playlists but should not be able to edit them.

3.6 Downloader (C)

1. Some tracks can be (notionally) downloaded to the customer 's machine and copyright requirements may demand that play rights will be for a limited time only. These are called "licensed tracks". Other tracks will not have this requirement and are called "license free tracks".
2. "Licensed tracks" can be downloaded for a maximum fixed period of 30 days, after which they will be automatically removed from the local machine. "license free tracks" can be downloaded for an unlimited amount of time.
3. Downloading is only available to those customers with an up to date subscription. Should the customer move back to the free service all tracks downloaded must be removed automatically.
4. Subscription users must be able to tag the tracks from the search results in order to download them.

4 Non-functional requirements

In addition to correct implementation of the functional requirements, your project submission will be assessed against the following non-functional requirements:

- *Ease of installation*: This is the most critical non-functional requirement. You will have to learn how to 'package up' your code in such a way that it can be

executed with a click/invocation of the application from within the file system. Ideally this could be a jar, a batch or an exe file.

- *Portable*: Your application must run on any PC with an appropriate JVM and other necessary software installed on it.
- *Ease of use*: This is judged from the perspective of an end customer . The elegance/attractiveness of the interface is also considered here.
- *Reliability/robustness*: The extent to which the code runs without failing. Some reasonable 'stress testing' must be performed.
- *Efficiency (speed):* GUI response time should be fast.
- *Efficiency (memory):* This will be judged on the amount of memory used when running, and the size of the application files.
- *Easy to Integrate:* Be developed in such a way that it will easily integrate any module with the other modules in the system.
- *Verisimilitude:* Be developed with sufficiently rich test data to able to demonstrate all functionality required.