Digital Music Lab Database Enrichment

Software Design Document

Version 0

January 16, 2014

Team 14
Atullya Singh
Marek Ceglowski

Prepared for
Computer Science 4ZP6 – Capstone Project
Instructor: Dr. Rong Zheng
Fall/Winter 2014-2015

Revision History

Date	Version	Author	Comments
11/01/2015	0.00	Atullya Singh	Document template and Introduction
12/01/2015	0.00	Marek Ceglowski	Module Overview
13/01/2015	0.00	Atullya Singh	Design Decisions and Data Flow
15/01/2015	0.00	Marek Ceglowski	Proofread
16/01/2014	0.00	Atullya Singh	Future Changes

List of Figures

List of Tables

Table of Contents

1. Introduction

1.1 Purpose

The goal of the project is to add columns to the music streaming information database managed by McMaster's Digital Music Lab, namely age and sex of users that stream songs using Microsoft's MixRadio application.

1.2 Description

This project aims to enrich the database provided by Nokia to the Digital Music Lab with age and sex inferences for users. The database includes download history for users, but not their personal information. Our program will infer age based on the reminiscence bump phenomenon, according to which, users mostly recall songs they heard when they were in their mid to late teens. The Last.FM database publicly provides some basic personal information about users and their streaming histories. By making a correlation between user age/sex and their listening preferences, we can assign age and sex information to users in our own database without compromising on their anonymity (having to share their personal information). This information will help with suggesting more music for users.

1.3 Scope

The scope of this project is only estimating the age and sex of the users. This project doesn't involve recommending songs based on the age and sex estimated. Developers at MixRadio would be expected to use this information to make it easier for them to recommend songs. The accuracy of the estimation is unknown at this point. The estimated ages of two users with similar musical tastes will likely be similar, in which case relative ages of users will be accurate.

2. Overview

2.1 Design Principles

2.1.1 Hierarchy

Hierarchy is a design principle for software with functions divided into levels where:

- Level 0 is the set of all units that use no other units
- Level x is the set of all units that use at least one unit at level < x and no unit at level > x

Hierarchical structure forms basis of design
- Facilitates independent development
- Isolates ramifications of change
- Allows rapid prototyping

Our motivation for using this design principle is that it reduces module interactions by restricting the topology of relationships. Our main function is at the highest level and needs the find-users, find-usersongs and find-userinfo functions, but the reverse is not true. Similarly, find-usersongs and find-userinfo need find-users, but find-users is an independent function.

2.1.2 Information Hiding

Information hiding (commonly used interchangeably with Encapsulation) is the hiding of data implementation by restricting access to accessors and mutators.

- Accessor: A method used to ask an object about itself
- Mutator: Public methods used to modify the state of an object, while hiding how it is modied.

Information Hiding is used in the implementation our functions that find users and our main algorithm, since it is imperative that the algorithms to find users and assign age and sex will keep being refined even though their functionality remains the same.

2.2 Document Structure

The document structure is based on **IEEE Std 1016-1998**. It was modified according to the research-based nature of our project. Our document gives a general overview of the description, scope, and context of our project, followed by the direction we envision it taking and any changes we predict will happen. The project is first evaluated at a macro level in terms of its Implementation Details and System Architecture. It is then decomposed into modular components and the implementation of each module is evaluated. Finally, the data flow will be discussed.

2.3 Requirements and Traceability

SRS Requirement	DD Module
Estimation of Age for Current Database Information	3.2.1
Estimation of Sex for Current Database Information	3.2.1
Updating Inferred Age	3.2.2
Updating Inferred Sex	3.2.2
Estimating Age before Adding a New User	3.2.3
Estimating Sex before Added a New User	3.2.3

Table 1: Requirements and their corresponding design documents modules.

2.4 Version History

Version	Date
Version 0	January 16, 2014

Table 2: Version History

3. Likely Future Changes

3.1 Anticipated Changes

Include algorithm for estimating user sex Develop an algorithm to estimate user age from their listening preferences

Assign artist scores Create a function that assigns a score to each artist based on how much they conform with the ideal reminiscence bump. This will help with determining user age based on the artists they listen to.

Increasing number of Last.FM usernames Enhance the UsernameCollector function to add users' friends on Last.FM so that artist scores can be based on a larger sample size. This will further help enhance the reminiscence bump score to be assigned to each artist

Eliminating the CSV file and entering data directly into the Artist DB After user streaming history is collected, the calculated ages an artist was heard at, will be entered directly into the Artist DB, thus eliminating an extra step

3.2 Extra Design Features

Data Visualization Include a map of users' ages and sexes in different regions based on an input of artist or genre. Users of the visualization will be able to look at statistics such as average ages and sex-ratios in different areas for any artist that is in the database

4. Development Details

4.1 Language of Implementation

The source code is written in Python. This design decision was taken given the open source nature of python and the vast abundance of libraries available

4.2 Supporting Frameworks/APIs

- MySql (Digital Lab Database) [1]
- BeautifulSoup (HTML Parsing) [2]
- csv (Python library for manipulating CSV files) [3]
- pylast (Python library for Last.FM API) [4]

4.3 Timeline of Critical Components

Component to be Implemented	Expected Completion Date
Development of algorithm for estimating sex	January 18th, 2014
Increasing Last.FM usernames	January 26th, 2014
Assign Artist Scores	January 28th, 2014

Table 3: Critical components timeline table for Revision 0

5. Data Flow Diagram

The diagram below shows how the described modules are connected, and how information flows between these modules. Please refer to individual sections for specific information on the data generated by each module.

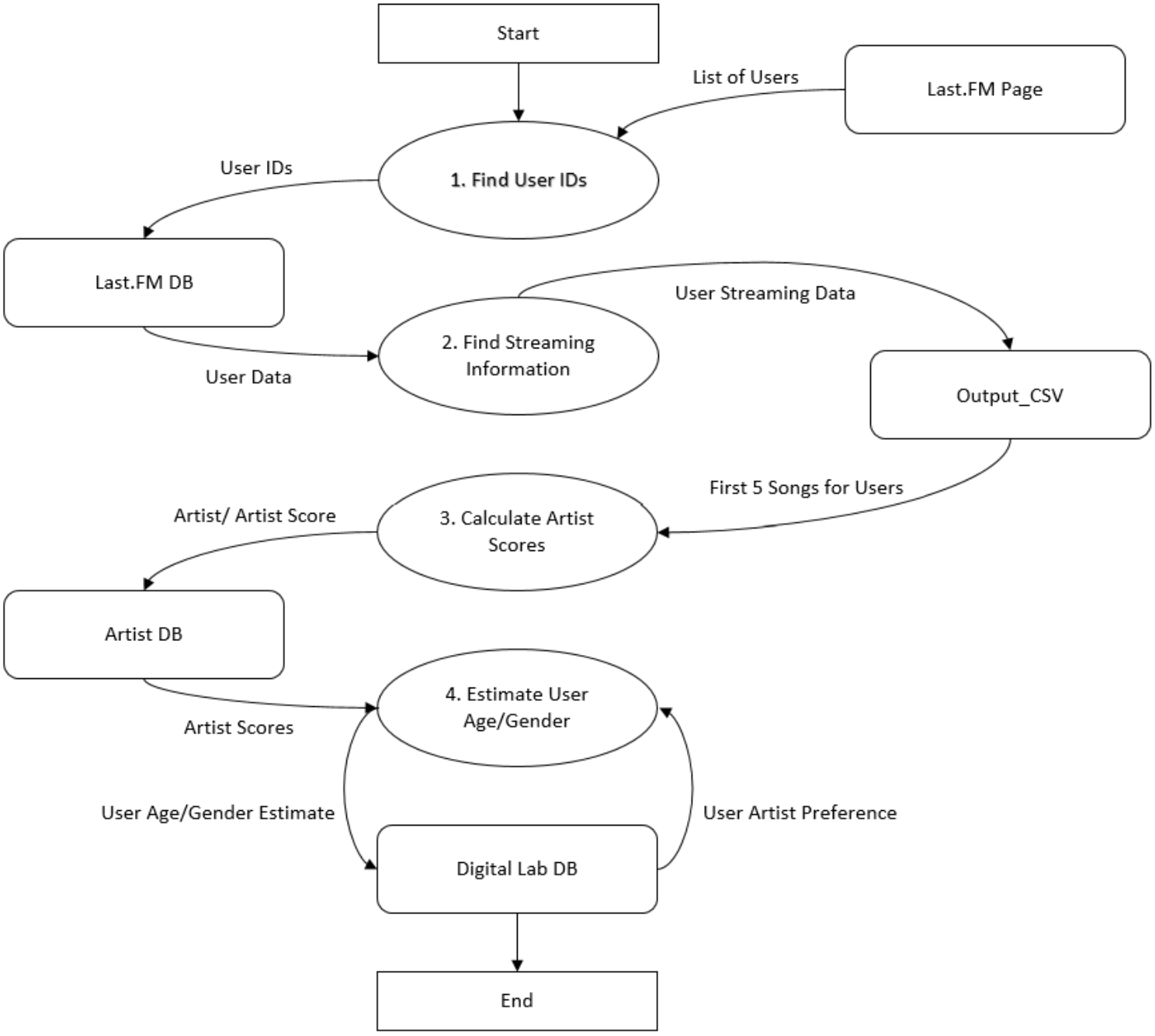

Figure 1: Getting usernames, user music preferences and determining user age and sex based on that information

6. Username Collection Module Overview

The following section includes modules from the main simulation program – StrokeModel.py

The following section includes information about the Last.fm username collection program – LastFMUsers.py

6.1 CLASS: Main Class

6.1.1 Description

Last.fm's API allows for data collection from users with their username, however it does not have any functions for gathering random usernames. Thus, this program was written to extract the username's from Last.fm's online users page. This program is run every 10 minutes for 24 hours to ensure distribution across all time zones.

6.1.2 FUNCTION: Main Function

6.1.2.1 Description

- Entire script is written into one function. It extracts usernames from all of Last.fm's active user pages.

6.1.2.2 Inputs

- Last.fm Active Users Webpage URLs

6.1.2.3 Processing

- The script creates a temporary HTML objects for Last.fm's active users pages
- Extracts the individual usernames from each page
- Outputs usernames to text file if not already found within the text file
- Outputs the current number of usernames in the text file

6.1.2.4 Outputs

- List of usernames into "userlist.txt"

6.1.2.5 Error Handling

- N/A

7. User Public Information Collection Overview

The following section includes information on the modules within the *current* main program - GetUserInfoAndSongs2.py (Note: GetUserInfoAndSongs.py is an initial test version of the program, not used for the actual project)

7.1 CLASS: Main

7.1.1 Description

This program uses Last.fm's API to collect publicly available user information that is specifically useful to the project and stores into CSV files for concatenation and visualization.

7.1.2 FUNCTION: Main Function

7.1.2.1 Description

- Entire script is written into one function. It extracts user information from Last.fm using the collected username list and Last.fm's free API.

7.1.2.2 Inputs

- CSV file location - *Output_CSV*
- Username list TXT file location - *User_TXT*
- Number of tracks to collect - *tracksToCollect*
- API Public Key and API Secret Key - *API_KEY* & *API_SECRET*
- Last.fm Developer Username/Password – *username* & *password_hash*

7.1.2.3 Processing

- The script starts by creating a connection to Last.fm's API and enabling a rate limit to ensure the number of calls per second does not surpass Last.fm's terms of service (5/second)
- The script then writes a header to the CSV file
- The script loops through all the usernames in the username text file (*User_TXT)* and collects some of their publicly available information
- This information includes:
    - Username
    - Age
    - Sex
    - Country
    - First tracks listened after signing up on Last.fm
- The year of release is used to determine the "age first heard" for each track
- The first tracks listened are reduced to the first *tracksToCollect* number of tracks (usually five)
- This information is outputted to the CSV file (*Output_CSV*)

7.1.2.4 Outputs

- List of user information into *Output_CSV*

7.1.2.5 Error Handling

- If there is an exception when the "age first heard" is being determined, the user is skipped and the program continues from the next user
- If there is an undetermined error at any time during the data collection of a user, that user is skipped and the program continues from the next user

8. References

[1] MySQL, **MySQL Community Server 5.6.22**,
http://dev.mysql.com/downloads/file.php?id=454673

[2] Crummy, **Beautiful Soup 4.3.2**,
http://www.crummy.com/software/BeautifulSoup/bs4/download/4.3/

[3] Python Software Foundation, **csv Library**,
https://docs.python.org/3/library/csv.html

[4] pylast, **pylast**,
https://github.com/pylast/pylast