**To Add To Requirements Write-Up**

**Music Informatics Write Up**

“…music informatics simply involves research in areas such as the automatic transcription of music, chords, and chord progressions; key detection; and music classification.” [1]. This simply means that music informatics is the analysis of music by means of the fundamental sounds that construct the music. In terms of Personal Informatics, this analysis can illustrate trends in qualities such as moods and behaviours. This is possible by analysing musical attributes such as “[…] melody extraction, chord recognition, beat tracking, tempo estimation, [..] and mood prediction” [2].

Moreover, in a forever growing music industry with apps such as Apple Music, Spotify and YouTube Music, there’s an increase in the interest people are taking in their own music tastes. This is becoming increasingly popular with features such as Obscurify or Spotify Wrapped “which reveals users’ favourite music from the past year, [2019], has been released to [Spotify’s] 243 million users” [3].

Finally, due to other qualities which can be associated with the type of music that you listen to, such as “instrument identification, music similarity, genre classification, […]” [2], it is also possible to recommend songs similar to those that you already listen to. An increasingly smart analysis of these trends is much requested in order to get more accurate song recommendations. It is evident that there is the need for more research into this growing area of personal informatics which is shown in the research that’s been carried out over the last decade which has revealed “chord recognition, genre recognition, and mood estimation—are each converging to performance plateaus below satisfactory levels”[2].

[1] <https://musicmindandbrain.wordpress.com/2013/01/27/music-informatics-a-modern-approach-to-studying-music/>

[2] Humphrey, E.J., Bello, J.P. & LeCun, Y. Feature learning and deep architectures: new directions for music informatics. J Intell Inf Syst 41, 461–481 (2013). <https://doi.org/10.1007/s10844-013-0248-5>

[3] <https://www.theguardian.com/culture/2019/dec/06/spotify-wrapped-users-alarmed-by-their-own-listening-habits>

**Description of Types of Users in Problem Domain**

There are several different types of users in our problem domain. Everyone who uses a music streaming service such as Spotify, Apple Music, and Deezer. According to a study taken in 2019, “89% [of the 34,000 participants], listen to music through on-demand streaming.” [1]. This evidently shows the large audience of stakeholders regarding music streaming services whom would be interested in knowing more about their music interests and discovering similar songs to the ones they are already listening to. It is the majority of music listeners whom would benefit from using a system like the one we are developing, novice and experienced users alike.

The more experienced music listeners stand to gain a deeper understanding of why they like the music they do, for example by seeing strong correlation between beats per minute and music danceability perhaps. A less experienced music listener may not appreciate this sort of information as much as the more avid music streamer, however, both types of users would benefit from discovering more songs that they might like through song recommendations based on mood, tempo, genre, etc…

All types of users may feel concerns regarding the ethics to a server listening constantly to the user’s phone / streaming device. There are some clear ethical implications here such as security and privacy. This may stunt the amount of potential stake holders, but by using hashing and salting of user passwords and storing only the necessary user data on a secure server may convince people to use our music informatic tracking tool. There also may be concerns about selling personal data to third parties or it being stolen, however, in order for our system to be functional, we only need to collect minimal personal data and it would never be sold to third parties. Many users would see that the ethical concerns aren’t too major and that the consequences are minimal in the rarity of anything happening.

[1] <https://www.ifpi.org/consumer-research.php>

**Use Cases Scenarios**

Use Case Scenarios are used for illustrating the expected, and alternate, flow of events for a given scenario. This can be extremely useful to developers when programming and testing a system as it’s apparent what should happen for any given circumstance. For any given exception, the system should handle it accordingly. Finally, if any preconditions aren’t met, then you should be able to see what happens for that given scenario.

**Scenario 1 – Logging In To System**

**Actor:** User

**Use Case:** Logging into the system

**Preconditions:**

* User has login already
* User has access to the internet
* Server is running

**Normal Flow of Events:**

1. User enters login details
2. Login details are checked against those held in the database
3. Server confirms login details
4. User is logged in

**Alternate Flow of Events:**

1. User isn’t able to enter log in details
2. System notifies user that they are offline, flow starts at step 1.
3. User doesn’t already have a login account
4. Server sends back error message saying user not found. Returns to log in screen for user to try again. Flow continues from step 1.
5. Server can’t confirm log in details due to not currently running
6. System notifies user that login is temporarily unavailable and should try again soon. User returns to login screen and flow continues from step 1.
7. User can’t be logged in
8. System notifies user of unsuccessful log in. Flow continues from step 1

**System State Upon Completion:**

The user is logged into the server and connection is maintained until the user disconnects from the server or the user has been idle during the session for 24 hours.

**Scenario 2 – Creating an Account**

**Actor:** User

**Use Case:** Creating a user account

**Preconditions:**

* User is connected to the internet
* Server is running
* User doesn’t already exist in the database
* Passwords entered match
* All fields filled in correctly in a valid format

**Normal Flow of Events:**

1. User enters new user details
2. All fields are filled in correctly
3. Server checks that the user doesn’t already exist
4. Account is made
5. User returned to log in screen

**Alternate Flow of Events:**

1. User can’t enter login details, user isn’t connected to the internet
2. System notifies user accordingly; flow continues from step 1
3. Fields are filled out incorrectly
4. A field(s) is left blank
5. System informs user, flow continues from step 1
6. Passwords entered don’t match
7. System asks user to retype the passwords, continues from step 1
8. A field(s) has invalid format
9. Blank spaces for username for example. User notified and asked to try again, flow continues from step 1
10. User already exists
11. System informs user that user already exists. User taken to login page
12. Account isn’t made; server isn’t running
13. System informs user that account creation was unsuccessful. Flow continues from step 1
14. User isn’t returned to login screen
15. System notifies user that they couldn’t be returned to login screen. Flow continues from step 1

**System State Upon Completion:**

User account is created, and user is returned to login screen with new login details. Database contains login details of a new user.

**Scenario – Requesting Data From the Server**

**Actor:** User

**Use Case:** User requesting data from the server

**Preconditions:**

* User is logged in
* User is able to request data
* Server is running correctly

**Normal Flow of Events:**

1. User sends request to server for some data
2. Server Connection is checked
3. Server handles request
4. Data is processed server side
5. Data is sent back to client
6. Data is displayed to user

**Alternate Flow of Events:**

1. Request isn’t sent to the server; user isn’t logged in / is offline
2. System notifies user they need to log in before sending requests. User sent back to login screen.
3. Server isn’t running correctly
4. System notifies user that server is offline. Flow continues from step 1
5. Server can’t handle request
6. Server notifies user that the request couldn’t be handled. Flow continues from step 1
7. Data can’t be processed
8. Error while processing data. System notifies user of failed request; flow continues from step 1
9. Data can’t be sent back to client
10. No data to send back to client. System notifies user and flow continues from step 1
11. Connection broken whilst sending data back to client. System notifies user of broken connection. User disconnected and returned to login screen
12. Data can’t be displayed
13. Invalid data sent back over to client can’t be displayed. System notifies user that request wasn’t completed. Flow continues from step 1.

**System State Upon Completion:**

Server has sent some processed data back to the client and it is displayed on the client’s screen.