Katie Kim (kjk5drb), Andrew Shin (ajs5qgm), Gabriella Woo (gdw5bv), Winston Zhang (wyz5rge)
CS 4750: Database Systems
8. September, 2023

CS 4750 Semester Project Proposal

Introduction

The semester project for CS 4750 is an assignment given to students as a means to offer a demonstration of conceptual and practical understanding of database systems and how they are integrated in software engineering solutions. The assignment asks students to design and develop an application, either web-based or not, which makes use of a relational database. The project is broken down into the following components, each with its own deliverable and deadline:

1. **Proposal and UI design:** A brief description of the intended project, as well as accompanying UI/UX proposals
2. **DB design:** Graphical design of database which the application will utilize, complete with attributes, relationships, dependencies
3. **DB setup and SQL:** Implementation of database design using MySQL
4. **Final deliverable:** Implementation of all functionalities, quality assurance, and final report

This document is our team's submission for component 1: Proposal and UI design.

The following requirements were enumerated in the assignment description for which the project application needs to satisfy:

1. Application must use a relational database, created in a MySQL server
2. Database must contain realistic data that is relevant to the domain of the application
3. Application may or may not web-based
4. Application must include dynamic behaviors, such that front end response to user actions and inputs
5. Application must user and interact with the database such that it allows for user actions to create databases, create tables, and retrieve/insert/delete/update data
6. Application must also allow users to sort, search/filter, export, or import data
7. Application must support multiple users simultaneously
8. Application must support returning users such that they can interact with existing data associated with their account

The remainder of this document is a description of the application that our team intends to build, and how it fulfills each of the above enumerated requirements.

Proposed Project

EchoShell is a text-based web application that seeks to emulate Spotify in a low-tech approach. Low-tech "demakes" and minimalism have become a common trend in the tech landscape, with prominent examples being Bloodborne PSX, a fan-made video game remake of the hit PS4 game Bloodborne, but reimagined in PS2 graphics, as well as the Light Phone, an "dumb phone" that is noticeably lacking in many features that mainstream phones boast, only allowing users to call, text, and listen to music. Low-tech tech solutions have become a sort of contra force to the feature creep of modern consumer products, as significant groups in the market have begun to become disillusioned with always-online, subscription based products-as-a-service that many, if not all, tech giants have begun to offer. Low-tech is also partly a result of nostalgia for older generations of technology, primarily among millennial consumers, which would explain the resurgence in previous obsolete technologies such as flip phones and vinyl records. EchoShell is our team's way of bringing the music listening experience into a more primitive form.

EchoShell will operate similarly to Spotify and other music streaming platforms, where users can browse through a library of songs and artists. Our app will allow users to create their own playlists and search through the database with different criteria, such as title, album, playlist, and genre. These playlists will be saved to the user's account and accessible to the user if they would like to modify them in the future. The difference between EchoShell and current music streaming applications out there is that EchoShell will be entirely operated through a Linux-like terminal, which offers a simple and minimalist approach for users to easily navigate the app.

Dataset

The domain of the dataset we will use for our project is within the music/entertainment domain. Some of the attributes we may use from our dataset are:

- Song Name
- Artist Name
- Album Name
- Publish Date
- Publisher Name
- Number of Total Listens

These attributes can be beneficial to obtain information about the songs actively available on Spotify. Since the main purpose of our project is to be able to freely create playlists, the size will be as big as we deem necessary. For the scope of this project, we may limit the pure number of songs added to the dataset by capping the data by publish year, e.g. songs after the year 2010. This ensures that the size of the dataset won't be astronomically big, sacrificing the efficiency of the project for potentially unnecessary coverage. We will obtain the dataset by using the Spotify Web API from the Spotify website, https://developer.spotify.com/documentation/web-api

Functionalities

Once users are logged into the application, they will be able to perform the following functions:

- Create and name playlists so that users can organize their music collections based on genre, mood, etc.
- Add song titles into the created playlists (song titles will either be from an existing music database (ex: Spotify Web API) or will be from our own manually made music database)
- Remove or delete song titles from the playlists as well as delete whole playlists if the user would no longer like to listen to those songs/playlists
- View all playlists as well as list all song titles in any specific playlist so that the user can see existing collections
- Edit or update any of the playlists after creation by adding more or removing song titles since music tastes/eras/trends are constantly changing

- Search for song titles in the existing music database so that users can explore and add new song titles into their playlists

User Interface Design

The proposed user interface will be quite simple, given the text-based nature of the application. The application would closely resemble a command terminal that is available on UNIX machines. Because the application imitates Spotify in functionality, the UI will be green and black to mimic the Spotify logo’s colors, although the possibility of color customization may be a possible feature that could be implemented in later stages of development.

When the user opens up the application, this could be the view that they expect to be greeted with:

```
Welcome to EchoShell!
Logged in as: [username]

For a list of navigation commands, please use the ‘help’ command
For a list of EchoShell commands, please use ‘echoshell --help’

[user@echoshell] $
```

The user would then be able to navigate the system as if it were a UNIX filesystem:

```
[user@echoshell] $ ls
1. Homework Music
2. Chill Vibes
3. Gym Mix

[user@echoshell] $ cd Chill Vibes
[user@echoshell] $ ls
1. Dancing Queen - ABBA
2. Hotel California - Eagles
3. Bohemian Rhapsody - Queen
4. Rasputin - Boney M
5. Take On Me - A-ha

[user@echoshell] $
```

The user would be able to use a combination of commands to perform actions that would, behind the scenes, interact with the database such that data is being added/changed/deleted, which would be reflected in the terminal by executing the same UNIX shell commands:

```
[user@echoshell] $ echoshell --search Beggin
Search results:
Beggin’ - Madcon, song id: [some_number]
Beggin’ - Maneskin, song id: [some_number2]

[user@echoshell] $ echoshell --add [somenumber_2]
[user@echoshell] $ echoshell --remove Hotel California
[user@echoshell] $ ls
1. Dancing Queen - ABBA
```

```
2. Bohemian Rhapsody - Queen
3. Rasputin - Boney M
4. Take On Me - A-ha
5. Beggin’ - Maneskin
```

Honor Pledge

On our honor as students, we have neither given nor received unauthorized aid on this assignment.

Katie Kim (kjk5drb),
Andrew Shin (ajs5qgm),
Gabriella Woo (gdw5bv),
Winston Zhang (wyz5rge)

8. September, 2023