CSE348 DATABASE MANAGEMENT SYSTEMS
SPRING 2025
TERM PROJECT
DUE: 24.05.2025 23:59
DEMO SELECTION DEADLINE: 25.05.2025 23:59

MUSIC PLAYER

In this project, you need to develop a simple music player web application. To accomplish this, you will need a database to store user data, an HTML file to provide access to the web application which is your music player through a browser, and PHP code to handle interactions between browser and the database.

Each required file is explained below:

- *index.html:* The initial page of the music player application. It includes a button to initialize the database. After clicking this button, the database needs to be created, and the login page should be displayed to the user.
- *install.php:* Contains setup-related code for the database such as creating database and tables, filling these tables etc. After this file is executed, the database needs to be ready.
- *generate_data.php:* Used to generate datasets for the specified entities.
- *login.html, login.php:* Handles user authentication. If the user is authenticated, a session will be started, and the user needs to be redirected to the homepage.
- *homepage.html, homepage.php:* Displays the homepage of the user which includes playlists, music history and artist information. All functionalities are described in the Part III.
- *playlistpage.html, playlistpage.php:* Allows the user to view the songs in their playlists.
- *currentmusic.html, currentmusic.php:* Displays the currently playing music information.
- *artistpage.html, artistpage.php:* Shows artist details, including artist information, their albums and most listened songs. All functionalities are described in the Part III.
- *generalSQL.html, generalSQL.php:* The page about genre-related and country-related operations.

PART I: CREATING DATABASE AND TABLES OF THE PROJECT

Prepare an installation page with PHP (the outline of it is given in install.php file) and when clicked on the installation button via HTML page, it should create the tables described below and fill these tables with the information described in Part II. Do not change given table names (they are written in bold).

(i) Create a database which is named by your name (eg. name_surname)
(ii) Create a table for keeping **USERS**, **PLAY_HISTORY**.
(iii) Create a table for **ARTISTS** and **ALBUMS**.
(iv) Create a table for holding information about **SONGS**.
(v) Create table for holding **COUNTRY**s.
(vi) Create table for **PLAYLISTS** and **PLAYLIST_SONGS**.

PART II: INSERTING ELEMENTS TO THE TABLES

The required attributes for each table are listed below. If necessary, you may add additional attributes. You must insert the minimum required data before the system can function properly. Do not change given attribute and table names.

- USERS (minimum 100 user): country_id, user_id, age, name, username, email, password, date_joined, last_login, follower_num, subscription_type, top_genre, num_songs_liked, most_played_artist, image
- PLAY_HISTORY (minimum 100 history): play_id, user_id, song_id, playtime
- ARTISTS (minimum 100 artists): artist_id, name, genre, date_joined, total_num_music, total_albums, listeners, bio, country_id, image
- ALBUMS (minimum 200 albums): album_id, artist_id, name/title, release_date, genre, music_number, image
- SONGS (minimum 1000 songs): song_id, album_id, title, duration, genre, release_date, rank, image (needs to be same image with the album)
- PLAYLISTS (minimum 500 playlists): playlist_id, user_id, title, description, date_created, image
- PLAYLIST_SONGS (minimum 500 songs): playlistsong_id, playlist_id, song_id, date_added
- COUNTRY (include all countries): country_id, country_name, country_code (like ‘TR’ for Türkiye)

Write a *generete_data.php* script that generates data for each entity. This php code will read data from the text file and by using given values, it will create different combinations. The size of the input file needs to be given wisely. For instance, if we create 100 people, at least we need 80 first name. Create a .sql file that includes INSERT INTO statements. The input/output example is given below:

```
// input.txt for names – read this file as an input line by line
Senem
Ayşen
Eda
Bahar
Ömer
Emre
…
```

```
// output.sql for names – after creating this file, read this to insert those statements into created table
Person_ID, Names
INSERT INTO Persons VALUES (1, “Senem”);
INSERT INTO Persons VALUES (2, “Ayşen”);
INSERT INTO Persons VALUES (3, “Eda”);
INSERT INTO Persons VALUES (4, “Bahar”);
INSERT INTO Persons VALUES (5, “Ömer”);
INSERT INTO Persons VALUES (6, “Ayşen”);
```

PART III: DESIGNING WEBSITE

GENERAL RULES

- You are free to design your pages creatively. Well-designed pages and additional useful features may earn some bonus points; however, you are expected to come up with ideas on your own.
- When designing the application, make sure to guide the user properly. For example, if certain conditions are not met, display appropriate error messages.
- In addition, any operation that results in an update to the data must be handled correctly.

INITIAL PAGE

When the system starts, this page should display only a single button. When this button is clicked, it should initialize the database. The page title must be your name. After initialization, the login page should be displayed.

LOGIN PAGE

The system must have a login page where users can authenticate with their username and password. After authentication, each user must be welcomed by his or her name “Hello, Name!”, which is the title of the page.

AFTER AUTHENTICATION

After successful user authentication, (all) playlists along with their images should be displayed on the left side of the page. On the right side, the upper section should show the user’s last 10 played songs. The lower section should display 5 artists from the user’s county, ordered by their number of listeners.

Add a search input area to the top-left side of the page and open a new page based on the search result. If a playlist name is entered, the corresponding playlist page should open. If a song name is entered, the song’s page should be opened instead. The functionalities of these pages are explained below.

Add another search bar to the artists section to search for new artists. If the searched artist exists in the application, the artist’s page should be opened and a button to follow the artist should be displayed. If the user clicks to the follow button, related parts of the application should be updated accordingly.

A plus (+) button should be displayed in the top-right corner of the playlists section, allowing the user to add a new playlist.

To open a song from the history section, add a search bar at the top of the history area where the user can enter a song name. If the song exists in the database, the corresponding song page should be shown, and the history should be updated accordingly.

The search areas need to take input from user and create proper SQL queries in the background.

PLAYLIST PAGE

All songs included in the selected playlist should be displayed. In addition to this, the country information of each song’s artists should be displayed next to the song.

To add a new song to a playlist, include a search bar at the top of the page. If the song exists in the database, its page should be displayed along with an add button. When the user clicks the add button, the relevant parts of the application should be updated accordingly.

CURRENTLY PLAYING MUSIC PAGE

On this page, you should design a display area for played music information.

THE PAGE FOR THE ARTIST

On the left side, the artist's image and information should be displayed. The top side section should include the artist's last five albums, and the bottom section should show their top 5 most listened songs.

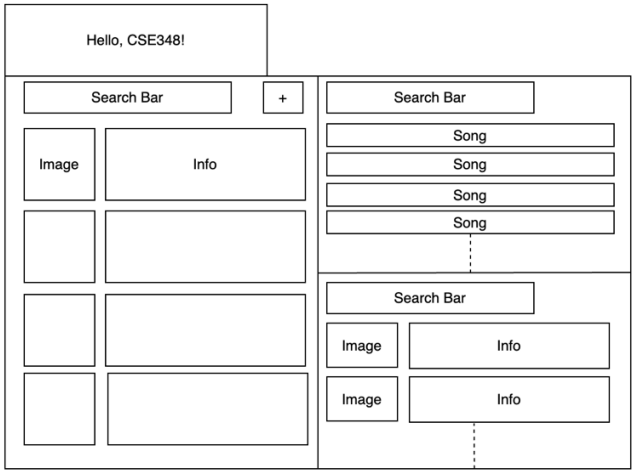
If the user selects a song, the currently playing music page should be displayed. If the user selects an album, it should be similarly to the playlist page; however, adding new songs to the album should not be allowed. You are expected to come up with additional criteria on your own.

ADDITIONAL GENERAL OPERATIONS

Create an additional page for genre-related and country-related queries. Include an input area for custom SQL queries. The results should fill 5 data slots in the page. For example, the page can display the top 5 genres listened to by users of this application or the top 5 most listened songs based on country.

EXAMPLE PAGE DESING LAYOUTS

This part is up to you - design it on your own. I will provide an example layout for the homepage after authentication, which you can use as a reference. Make sure to use additional wrapper/container tags such as to properly structure and divide the page.



DEMO AND SUBMISSION RULES

- You should design the database with the given entities and attributes.
- Create the ER diagram of the database (You can use Ampps for this operation, we did it in the lab). You should create your tables according to the ER diagram you draw. You should justify your design choices.
- Create a visual representation that illustrates the action flow of your design.
- The demo days will be announced on Yulearn **on 24.05.2024**. You need to select your demo slot until **25.05.2024 (23:59)**. You must attend the demo **on time**, if you don't attend even you send a project your grade will be 0.
- Submit an ER diagram (FullName_StudentID_ER.pdf), Action Flow (FullName_StudentID_ActionFlow), SQL, PHP and HTML files → (FullName_StudentID.zip)
- Add comments that includes your name and brief explanations in each file that you created.
- Submit your own work. Plagiarism will not be tolerated.