Project Final Report
Group 17: Danny Gimeno, Jack Wilkin, Lindsay Hauser
BanditPro

README: This project is built using PHP, CSS and HTML, XAMPP, and MySql.

Setting up Front End
XAMPP download: https://www.apachefriends.org/download.html
Follow the instructions to download and to start the localhost (apache server).

Navigate to Applications/XAMPP/htdocs and drag the project folders into htdocs. The project folders is called "Database-Final-Project"
Open a web browser and navigate to localhost/Database-Final-Project to interact with the UI.

Database-Final-Project/model/database.php contains 2 main folders: model and view. The model folder contains the functions for each page. The view folder contains the code for the formatting of the site. This includes any buttons, text or formatting for each page.

Setting up Back End
The database connection from frontend to backend is configured in the file: Database-Final-Project/model/database.php. We ran this project locally on our computers. We used connection 127.0.0.1 with root 3306 and created a new user called project_user with password: password123 in our MySql database. Currently, the database.php file is configured with this information, but this configuration can be changed to whatever database is being used. As long as the details in Database-Final-Project/model/database.php match the MySql configuration, the PHP code should connect with the database.
A script of the database information is provided in Database-Final-Project/dbsetup/database-final-project.sql. Running this script in MySql will cause all of the databases to be created, along with any constraints, procedures, triggers and functions we created. Additionally, it will also insert some sample data into the database to be used.

Program Functionality:
Use Cases (separated between the 3 different views: Customer, Artist, Venue
Customer:

1. Customer signs up for the site
    a. Open the site
    b. Click register
    c. Enter username and password, select "customer" and enter coordinates for location
    d. click register.
    e. Expected: The user is created in the database in the User table, and the profile page is navigated to on the UI.
2. Customer logs into the site (assume we already has an account)

   a. Open the site
   b. Click login
   c. Enter the username and password created in use case 1 and login
   d. Expected: If the username and password are entered correctly, we navigate to the profile page where show data is displayed. If the username and password are entered incorrectly, the user will not login and will have to try again.
3. Update password
   a. Login as customer (case 2)
   b. See update password field after login
   c. Enter a new password, click update password button
   d. In database in User table, the new password has been updated.
4. A user wants to purchase a ticket for a show that is not sold out.
   a. Login as a customer (see use case 2).
   b. Query show data and find a show with tickets still left
   c. Click purchase "Buy" button
   d. See that the ticket has been purchased (it will show up in tickets purchased section)
5. A user wants to purchase a ticket for a show that is sold out.
   a. Login as a customer (see use case 2).
   b. Query show data and find a show with no more tickets left
   c. Attempt to buy a ticket for this show by clicking "Buy".
   d. Confirm that you could not buy a ticket because the show was sold out (no ticket was added to your tickets). Top says "Could not buy ticket - show sold out!"
6. A user wants to see shows filtered by venue after a certain date (or after today)
   a. Login as a customer user
   b. Navigate to the "Filter" section
   c. Choose Venue, "after today" or input a date to search after that date, and search
   d. Expected: All shows with venues as the selected venue with shows after today or after the inputted date are retrieved.
7. A user wants to see shows filtered by venue before a certain date
   a. Login as a customer user
   b. Navigate to the "Filter" section
   c. Choose Venue, "before today" or input a certain date to search before, and search
   d. Expected: All shows with venues as the selected venue with shows before today or before the given date are retrieved.
8. A user wants to see shows filtered by artist before today, or before a certain date.
   a. Login as a customer user
   b. Navigate to the "Filter" section
   c. Choose artist, "before today" or enter a date to search for before that date, and search
   d. Expected: All shows with the selected artists before today or before the given date are retrieved.
9. A user wants to see shows filtered by artist after today or after given date

    a. Login as customer user
    b. Navigate to “Filter” section
    c. Choose artist, “after today” or input a date, and search
    d. Expected: All shows with the selected artist after today or after the inputted date are retrieved.

Artist

1. Artist wants to sign up for the site.
    a. Naviage to the site and click register
    b. Enter username and password and choose artist from list.
    c. Add coordinates for location.
    d. Enter information
    e. Expected: An artist user is created in the User table in the database, the user is now created
2. Artist logs into the site (already has an account).
    a. Navigate to the site and click login
    b. Enter the username and password that were created in the previous use case.
    c. Expected: If the username and password are entered correctly, it navigates to the profile page where a list of shows are shown and where shows can be queried. If the username and password are incorrectly entered, the user will not be logged in.
3. Artist wants to add a song
    a. Login as artist
    b. Navigate to input bar that say “Add Song”
    c. Add a song name
    d. Expected: Song is added to Song table in database
4. Artist wants to remove a song
    a. Login as an artist
    b. Navigate to “Song Name” section
    c. Click “Delete” button
    d. Song is removed from Songs list.
5. An artist wants to add a song to their set list for a specific show.
    a. Login as an artist
    b. Go to list of performances table, click “Edit Show” button
    c. Add a song
    d. Expected: Song is added to the set list, and in the database SetListShow record is added.
6. Update password
    a. Login as artist (case 2)
    b. See update password field after login
    c. Enter a new password, click update password button
    d. In database in User table, the new password has been updated.
7. An artists wants to update a song to their set list for a specific show.

    a. Login as an artist
    b. Navigate to Add Song section
    c. Enter song name, click “Add Song”
    d. Expected: The song is added under “Songs” Section and also added to Songs table in database
8. An artist wants to remove a song from their set list for a specific show.
    a. Login as an artist
    b. Click “Edit Performance” button next to a show
    c. Expected: a list of Song names shows up, click “Delete” button
    d. Expected: Song is removed from set list.
9. A artist wants to edit a past show that already occured
    a. Login as an artist user
    b. Navigate to “Past Performances” area
    c. Expected: The “Edit Show” Button does not appear, because you cannot edit a past show.

Venue

1. A venue wants to sign up for the site
    a. Navigate to site
    b. Click register
    c. Enter username and password, and choose venue from the list of options
    d. Click register.
    e. Expected: A new user is added to the User table. On the site, the user successfully registers and is brought to a profile page.
2. A venue logs into the site (already has an account).
    a. Navigate to site
    b. Click login
    c. Enter coordinates for location.
    d. Enter username and password of the venue user just added and click login. Expected: If the username and password are entered correctly, it brings the user to a profile page. If they are entered incorrectly, then the user is not logged in.
3. Update password
    a. Login as a venue (case 2)
    b. See update password field after login
    c. Enter a new password, click update password button
    d. In database in User table, the new password has been updated.
4. A venue wants to add a list of artists to a specific show.
    a. Login as a venue user.
    b. Pick a show to edit (in the table)
    c. Click “Edit Show”
    d. Add a new headliner/opener to the show by inputting an artist and choosing headliner/opener values.
    e. Expected: the artist is added to the show as headliner or opener

5. A venue wants to remove a artists from a specific show.
    a. Login as a venue user
    b. Pick a show to edit (in the table)
    c. Click "Edit Show"
    d. Remove an artist from the show by clicking "Remove Artist"
    e. Expected: the artist is removed from the show.
6. Venue creates a show.
    a. Login as a venue user
    b. Navigate to Add show area, enter show name, date, and ticket price.
    c. Expected: A record is added to Show database table.

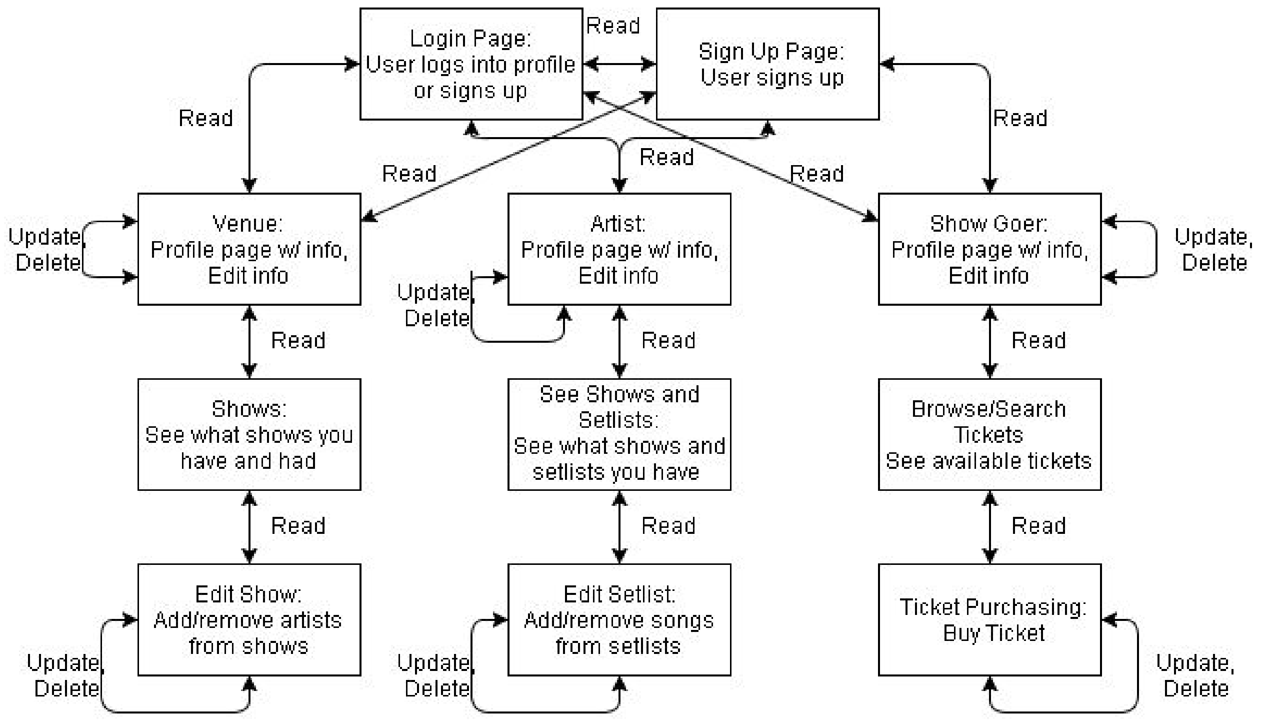
Database Requirements:

1. User should be able to create new objects in the database.
    - When a user logs into the page, if they are not registered yet (do not have a username and password yet), they can register. They register a username and password, and this gets added to the database user table. A user can either be a venue, an artist, or a show goer.
    - When a ticket is bought, ticket records are created.
    - When songs are added, song records are created
    - When shows are added by venues, show records are created.
2. User should be able to delete data from database.
    - An artist can remove songs from setlists. This deletes data off of the SetListSong data table.
    - If a venue wants to remove an artist from a show, this deletes data off of the Performs data table.
3. User should be able to read data from database.
    - When a user searches for shows and performances, data is read from the database to display to the user
    - A user can search for shows by date, and this queries show data from the database
    - The site shows all of the tickets that have been bought by a user, and this data is read from the database.
4. User should be able to update data in the database.
    - If a user wants to change their password, the new password is updated in the database.
    - If an artists wants to update a song in a set list for a show, any changes are updated in the database.
5. Completeness of operations provided to the user.
    - All of the use cases we set out to complete have been implemented
6. Modularization of system (use of SQL user defined Procedures, functions, triggers, and events)
    - A list of all procedures, functions and triggers is below.
7. Error handling system (testing of arguments)
    - There are triggers in place to take care of error handling (these are explained below)

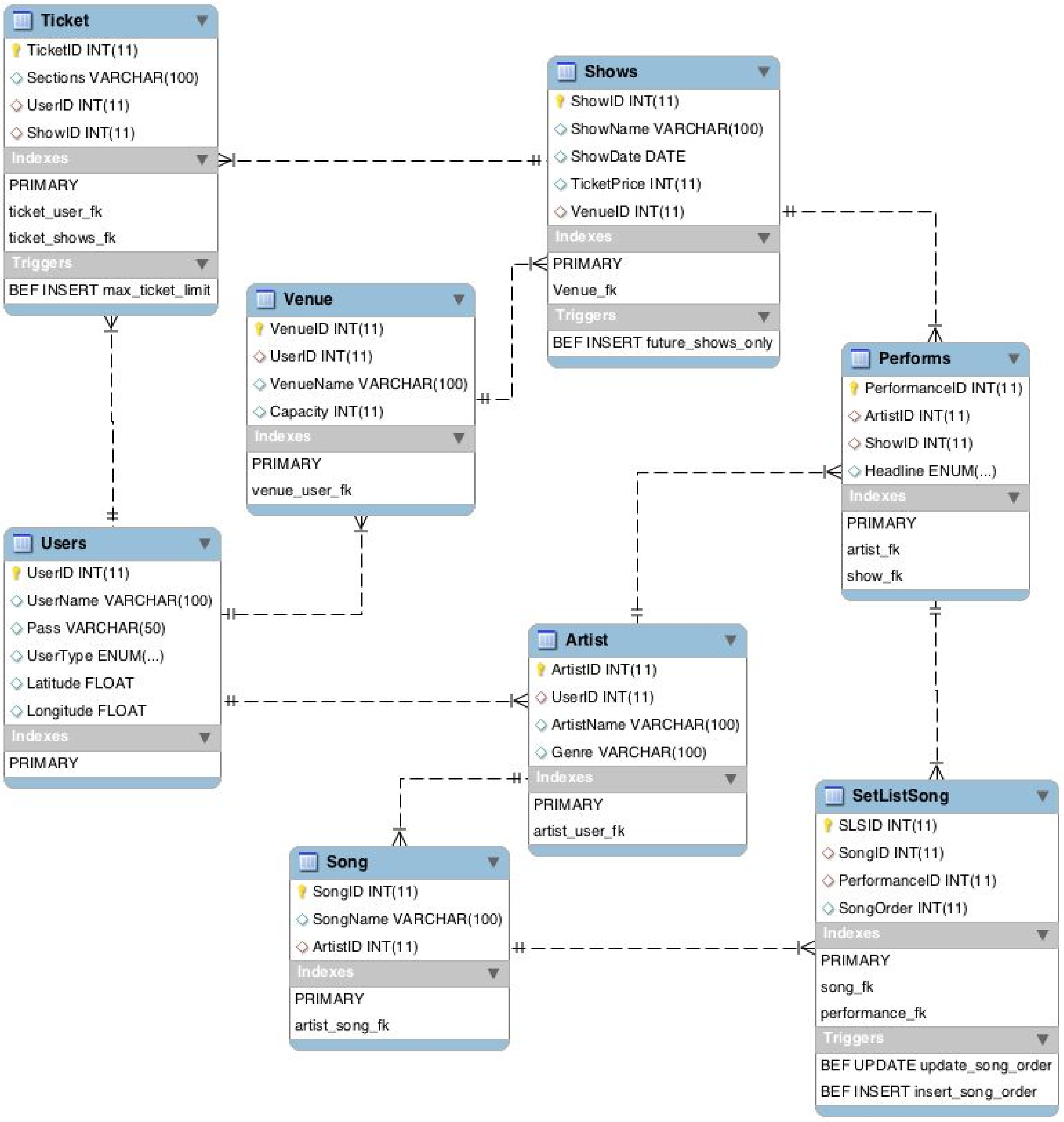
Technical Specifications:

The gui and client is in PHP for the backend, Javascript for the middle-ware, CSS and HTML for the front end. The machine must be able to connect to the internet to access the client. XAMPP is used for the cross platform web server. MySql is used as the database.

Final User flow of System/ Final EER diagram



Interaction with System & Commands
A user logs into the page. Either the user logs into a profile, or signs up. There are three users: venues, artists and show goers. A venue is able to create a venue profile, and update or delete information about a specific venue. The database must have read, update and delete functionality because a venue may want to see data about the venue, edit it or delete it. Venues can creates shows, and add or remove artists from show. An artist user has the ability to change profile information about an arts, see shows and setlists, and edit songs from setlists for shows. These are all CRUD operations as if Artists edit profile information or edit set lists, the database must be updated with CRUD operations. A show goer is able to edit profile information, browse and search tickets, and then purchase a ticket as well. This also requires all CRUD operations, as the database needs to read information in order to show the user, delete and update data if tickets are bought, and create data if a new user signs up.



Lessons Learned:

1. Technical expertise gained:
   - Lindsay & Jack gained some exposure/familiarity with PHP. Danny had some previous experience but expanded his knowledge of PHP.
   - Everyone: practiced writing and reading SQL queries, stored procedures, triggers and functions (everyone created queries, others would verify that it was correct)
   - Everyone practiced using git for source control
   - Everyone learned how to design a database with multiple tables and the appropriate relations between these tables.

2. Group work insights, time management insights, data domain insights etc:
    - We met in person a lot - we met to create the project proposal, to do the progress report, and also worked together when creating the actual project. Scheduling times was sometimes difficult because of busy schedules. All groups members respected each others time when meeting.

3. Realized or contemplated alternative design / approaches to the project:
    - Our schema changed slightly as we created the project. When we first envisioned the project, we had customers, venues and artists as completely separate users, however we ended up setting customers, venues and artists as different types of users. This means that all users are one of the three (we used an enumeration).
    - When the project was started, we weren't sure how to represent location in the database for each user. We ended up adding a latitude and longitude coordinate field to the user to track their location.

4. Document any code not working in this section
    - In the Frontend, all fields must be filled out when entering information, otherwise this may cause issues. Thus, the code isn't broken, but it hasn't been tested 100% with every edge case.

5. Provide a "Future work" section
    - An additional way to query data for a user could be: a user finds upcoming shows based on previous shows they have attended. This is something that could be expanded upon in the future.

Briefly describe the contribution of <u>each group member</u> to the design and development of your project.
We all worked together on coming up with use cases, designing the database, creating the project proposal, progress report and presentation. In terms of development, Danny worked on the front end linking PHP with database, creating the flowchart and working with the SQL queries. Jack & Lindsay worked on creating database stored procedures, functions, and creating the final report.

Please include a list of all the procedures, triggers, functions, transactions, error handling you have implemented as part of the project report, with a brief & concise explanation of their role/purpose. One line explanations are sufficient.

Triggers/Error Handling:

- future_shows_only
    - This trigger is fired when new shows are added. It makes sure that when a show is added, that only future shows (shows with a date greater than today) can be added. You cannot add a new show that takes place on a date in the past.
- max_ticket_limit

    - This trigger is fired when a new Ticket is created. It ensures that the number of tickets sold does not exceed the capacity of the venue. If the tickets for a show are sold out and another ticket for that show is purchased, then the ticket is prevented from being purchased.
- insert_song_order
    - This trigger is fired when an artists adds a song to a set list - it makes sure that when an artists puts a song order number, that it does not clash with an existing song order value in that set list.
- update_song_order
    - This trigger is fired when an artists updates a song in a set list - it makes sure that when an artists puts a song order number, that it does not clash with an existing song order value in that set list.
- insert_user
    - This trigger is fired when an artist or a venue user is added to the database. When an artist is created, it adds an artist record to the Artist table, or when a venue user is created, a venue is added into the database Venue table.

Procedures & Transitions: (for detailed steps, see use cases above)

- remove_song_from_set_list (delete)
    - Allows an artist to remove a song from a set list.
- add_song_to_set_list (insert)
    - Allows an artist to add a song to a set list
- get_tickets_sold_by_showId (select)
    - Allows a venue to see how many tickets have been sold at a specific show
- create_ticket (insert)
    - Creates a ticket when a customer buys one.
- tickets_purchased_by_user (select)
    - Returns how many tickets have been purchased by a specific user.
- get_user (select)
    - Returns the user given by the specified username and password
- get_user_no_password (select)
    - Returns the user given by the specified username
- get_show_by_showId (select)
    - Returns the show specified by the given show ID
- get_shows_by_month (select)
    - Returns all shows in a given month
- add_artist_to_show (insert)
    - Allows a venue to add a specific artist to a specific show
- remove_artist_from_show (delete)
    - Allows a venue to remove a specific artist from a specific show
- shows_for_artist_before_given_date (select)
    - Returns all shows an artist's has before a given date.

- find_similar_artists (select)
    - Returns similar artists (by genre)
- number_times_headlined (select)
    - Returns the amount of times the given artist has headlined
- shows_before_today (select)
    - Returns the shows that have taken place before today
- shows_after_today (select)
    - Returns the shows that are taking place after today (including today)
- get_average_venue_capacity (select)
    - Gets the average venue capacity of the venues
- add_show (insert)
    - Allows a venue to create a show
- get_headliner (select)
    - Returns the headliner from a specific show
- get_opener (select)
    - Returns the opener for a specific show
- get_lat_and_long (select)
    - Returns the latitude and longitude coordinates of 2 users (in order to compare them for location)
- get_filtered_shows (select)
    - Returns a query of shows based on a given user id, filter type (artist or venue) and a given date, and then a variable specifying if we want shows after or before that date. If the filter type is artist, it queries all shows for a certain artist after or before a given date. If the filter type is venue, it queries all shows for a venue after or before a given date.

Functions:

- get_tickets_by_user
    - This function calculates the amount of tickets a user has bought

Things that were demoed in class:

- Creating a new customer user during login (create/insert operation)
- Querying show data once a user has logged in, reading the amount of tickets left in order to purchase (read operation)
- Letting a user change their password (update operation)
- User buying a ticket (with available tickets left)(update operation)
- User trying to buy a ticket with no more available tickets left over (system prevents user from buying a ticket)

Things that were added after the demo:

- During the demo, most of the queries were written on the front end side (PHP was directly calling database queries). Since then, these queries have been converted into stored procedures in the mysql database and are called as procedures.

- There were no triggers yet implemented during the demo. They have been added for error handling.
- There were no functions in the database for the demo. They have been added.
- During the demo, only the view for a customer had been implemented. Views for artists, and venues have been added.
    - With the addition of artist and venues, a lot more functionality has been implemented.
    - Venues and artists can sign up/login to the site.
    - Venues can add shows, add performers (artists) to shows.
    - Artists can add, edit and remove songs from setlists.
- Location is now an attribute to users, so that a user can search for shows based on location.
- Venues can search for similar artists based on an artist based on genre (if they wanted to find a opener for a show or something similar to that).
- Implemented all of the search filters for customers - on the customer page, there is now a “filter” section where users can now search for shows by artist, venue, date (before and after), shows before today, or shows after today.