Prestige Worldwide Entertainment

Clinton Lohr, Charles Maddux

Team: Data Base Camp

Prestige Worldwide Database Link

INTRODUCTION TO DATABASES (CS_340_400_U2022)

Project Step 5 (Portfolio Assignment)

08.05.2022

Executive Summary

Throughout the design and implementation of our database, our original idea for what we wanted to accomplish with this project has remained largely unchanged. Though, with the help of peer feedback from both students and teaching assistants, as well as team discussions, modifications have been made along the way to improve both the logic and organization of our database.

We started this project with a clear idea of what we wanted this database to be. We began by brainstorming all of the entities that we thought we would need to include and what attributes those entities would need, including details and relationship attributes. This worked in our favor in that we were very thorough in what we needed to include, but it became apparent by the second week of peer reviews that we may have been too ambitious with everything we hoped to include. We received a few notes about how complicated our project seemed to be. In an effort to simplify things, we decided to cut the ‘Customers’ entity from our database. What this meant was that rather than our database being used to store performance dates as well as customer purchase information, we would instead just focus on storing details for upcoming performances at the five live performance venues.

As we began writing the data definition queries and the data manipulation queries, we started to discover some holes in the relationships between entities that resulted in having to create dummy entries. Though we thought that we had normalized the results, it turned out that there were foreign keys that needed to be added to the performance entity to avoid having to make these dummy entries. This was also confirmed in our project 3 review by our peers. This proved to be a relatively easy fix and we were able to achieve normalization.

Things really started to fall into place once we started building the web application. Because of the thorough planning and input from our peers, the tables for our project were easy to design and implement. We started with a bare bones html layout so that we could get the CRUD functionality established and then added css to make our application more visually appealing. Though we appreciated the responses of our peers for the final peer review, by the time we had received those responses, we had already addressed the suggestions that were made, including adding description and improving readability.

Project Outline

Prestige Worldwide Entertainment is the owner and operator of five live performance venues located in major cities across the United States with sights set on expanding internationally. Each venue features between one and three separate rooms, where different artists can be performing simultaneously. Between all venues, PWE puts on 400 events annually.

The concert schedule for all the venues is maintained using a single backend database. Employees from each venue can use a web application to select, insert, update and delete artist information from the database. The database holds the date of the event, the venue where the performance will be held, the room/stage within that venue, the band/group that is playing, instruments that a band/group will require from the venue, information about the tickets for the performance and an invoice of tickets for a performance.

Database Outline

- **Venues:** Records information about an individual venue that is not expected to change
    - venue_id: INT, PK, not NULL, auto increment
    - name: VARCHAR(50), not NULL
    - address: VARCHAR(100), not NULL
    - city: VARCHAR(100)
    - state: VARCHAR(2)
    - zip: INT(5)
    - phone: VARCHAR(12)
    - num_rooms: INT, not NULL
    - total_capacity: INT not NULL
    - Relationships:
        - 1:M relationship with Rooms. Each venue can have one or more rooms in the establishment. The Foreign key ‘venue’ used in the Rooms entity identifies the venue that the room is in.

- **Rooms:** Records the details of a room in a venue. Room capacity can be used if tickets are sold to individual performances, but are otherwise a part of total capacity. This table is most important for managing bookings and associating them with an individual room for a specific date.
    - room_id: PK, not NULL, auto increment
    - name: VARCHAR(50)
    - capacity: INT, not NULL
    - over_21: TINYINT(1), default=0
    - venue: INT, FK, not NULL
    - Relationships:
        - M:1 relationship with Venues. Because each venue can have multiple rooms, a ‘venue’ FK is used to identify the venue.
        - 1:M relationship with Performances. One room can have multiple performances, but one performance cannot take place in multiple rooms. The “room_id” FK represents the room in which the performance takes place

- **Performances:** Tracks performance details, combines artist information, room assignment
    - performance_id: INT, PK, not NULL, unique, auto increment
    - performance_date: DATETIME, not NULL
    - tickets_available: INT, not NULL, default=0 [total_capacity - tickets_sold]
    - room_id: INT, FK, not NULL
    - artist_id: INT, FK, not NULL
    - ticket_id: INT, not NULL
    - Tickets_ticket_id, FK, not NULL
    - Relationships:
        - M:1 relationship with Artists. One artist can play multiple performances, but once performance cannot have multiple artists
        - M:N relationship with Tickets, facilitated by an intersection table holding the ticket_id and the performance_id as foreignkeys. This relationship is facilitated by the intersection table “Ticket_Invoices”. Many tickets are sold to each performance, and a ticket can be good for multiple performances within a single venue (depending on the number of rooms in a venue).

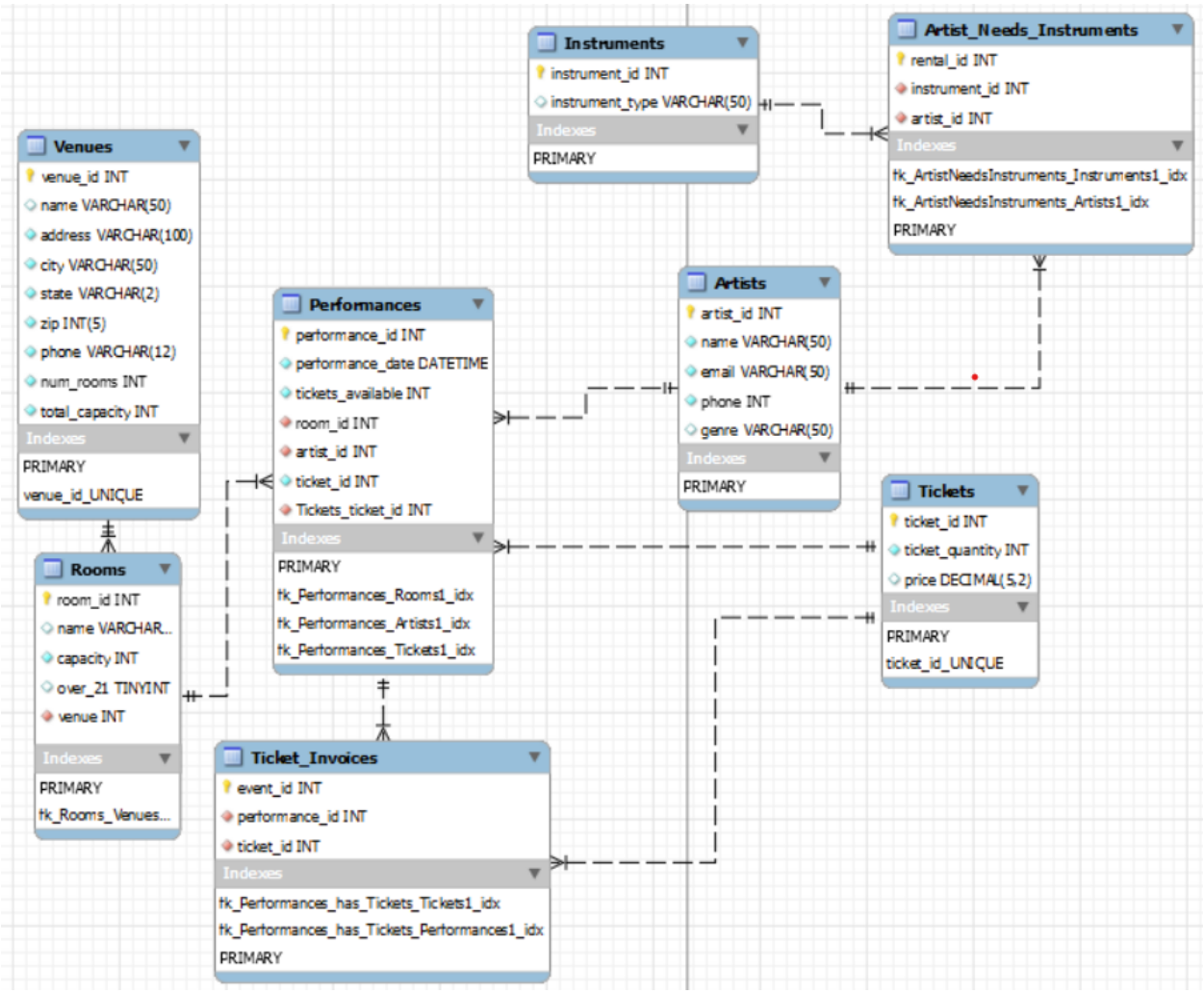
- **Artists:** Records the details of the artist that is playing at a venue. Maintains contact information and aids in pairing music of similar genres.
    - artist_id: INT, PK, not NULL, auto increment
    - name: VARCHAR(100), not NULL
    - email: VARCHAR(100), not NULL
    - phone: VARCHAR(10), not NULL
    - genre: VARCHAR(50)
    - Relationships:
        - 1:M relationship between Performances and Artists as one artist can play multiple performances, but multiple artists cannot play the same performance
        - M:N relationship between Artists and Instruments, facilitated by the intersection table “Artist_Needs_Instruments” holding the artist_id and the instrument_id as FKs. An artist may request many types of instruments, and multiple artists may request one type of instrument.

- **Instruments:** Contains a list of instruments that are available for request by artists performing at a venue.
    - instrument_id: INT, PK, not NULL, auto increment
    - instrument_type: VARCHAR(50)

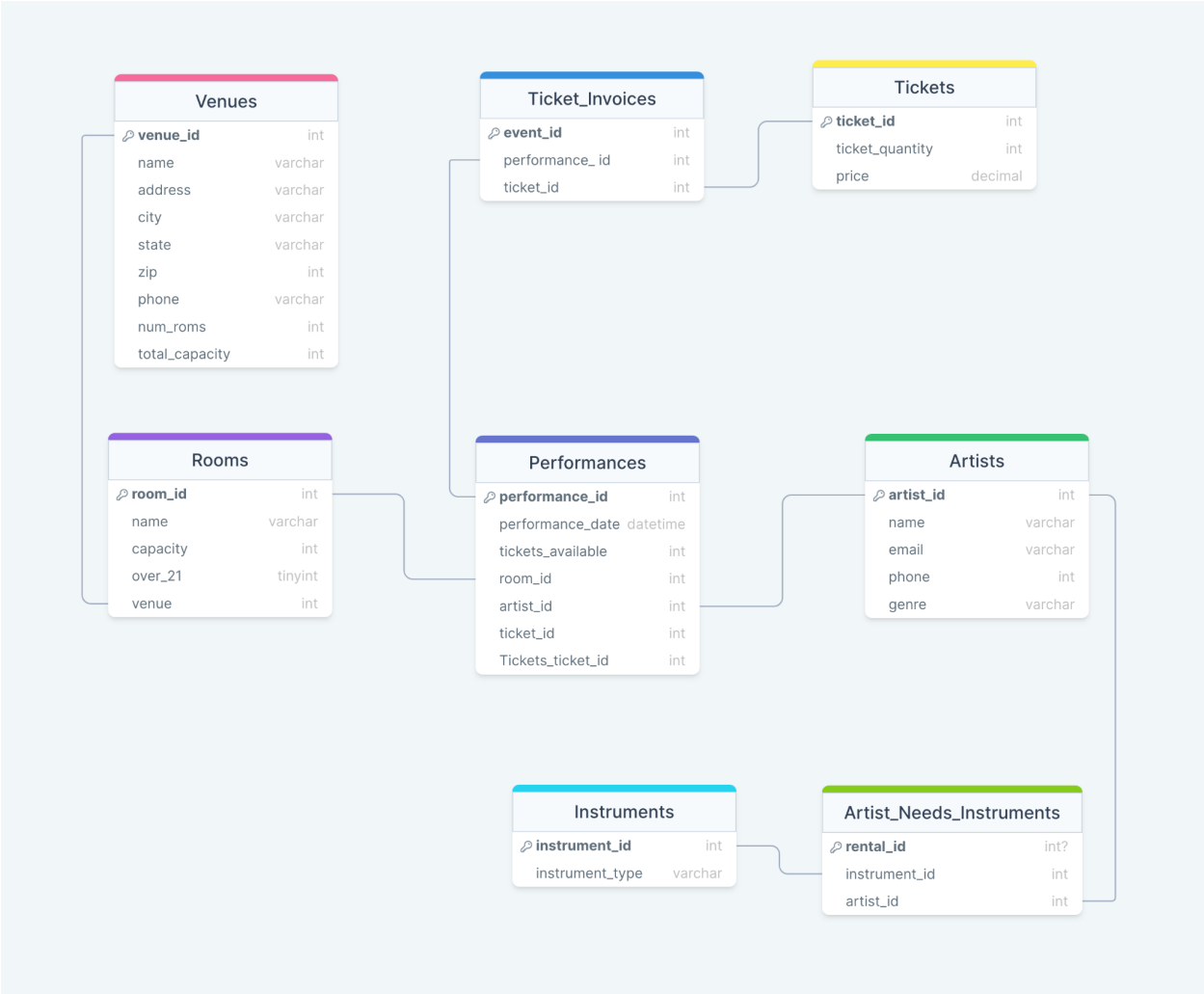
    - Relationships:
        - M:N relationship between Instruments and Artists, facilitated by the intersection table “Artist_Needs_Instruments” holding the artist_id and the instrument_id as foreignkeys. One type of instrument may be requested by multiple artists, and multiple types of instruments may be requested by one artist.

- **Tickets:** Records information about the admission ticket being sold for a performance
    - ticket_id: INT, PK, not NULL, UNIQUE auto increment
    - ticket_quantity: INT, not NULL
    - price: DECIMAL(5,2)
    - Relationships:
        - M:N relationship between Tickets and Performances, facilitated by an intersection table holding the ticket_id and the performance_id as foreignkeys. A ticket can admit an attendee to multiple performances in a venue and multiple tickets can be sold for any performance.

ER Diagram


Instruments
instrument_id INT
instrument_type VARCHAR(50)
Indexes
PRIMARY
Artist_Needs_Instruments
rental_id INT
instrument_id INT
artist_id INT
Indexes
fk_ArtistNeedsInstruments_Instruments1_idx
fk_ArtistNeedsInstruments_Artists1_idx
PRIMARY
Venues
venue_id INT
name VARCHAR(50)
address VARCHAR(100)
city VARCHAR(50)
state VARCHAR(2)
zip INT(5)
phone VARCHAR(12)
num_rooms INT
total_capacity INT
Indexes
PRIMARY
venue_id_UNIQUE
Performances
performance_id INT
performance_date DATETIME
tickets_available INT
room_id INT
artist_id INT
ticket_id INT
Tickets_ticket_id INT
Indexes
PRIMARY
fk_Performances_Rooms1_idx
fk_Performances_Artists1_idx
fk_Performances_Tickets1_idx
Artists
artist_id INT
name VARCHAR(50)
email VARCHAR(50)
phone INT
genre VARCHAR(50)
Indexes
PRIMARY
Tickets
ticket_id INT
ticket_quantity INT
price DECIMAL(5,2)
Indexes
PRIMARY
ticket_id_UNIQUE
Rooms
room_id INT
name VARCHAR...
capacity INT
over_21 TINYINT
venue INT
Indexes
PRIMARY
fk_Rooms_Venues...
Ticket_Invoices
event_id INT
performance_id INT
ticket_id INT
Indexes
fk_Performances_has_Tickets_Tickets1_idx
fk_Performances_has_Tickets_Performances1_idx
PRIMARY

Schema Diagram



Sample Data

Venues								
venue_id	name	address	city	state	zip	phone	num_rooms	total_capacity
1	Broken Spoke	3201 S. Lamar Blvd.	Austin	TX	78704	5124426189	3	250
2	SOMA	3350 Sports Arena Blvd.	San Diego	CA	NULL	6192267662	1	110
3	The Crocodile	2505 1st Ave.	Seattle	WA	98121	206-441-5555	2	310
4	The Joint	3730 N Clark St	Chicago	IL	60613	7735494140	1	250
5	The Ogden	935 E Colfax Ave	Denver	CO	80218	3038321874	1	550

Rooms				
room_id	name	capacity	over_21	venue
1	Pony Room	75	0	1
2	Oregon Trail	50	1	1
3	The Clydesdale	125	1	1
4	Calamity	45	1	2
5	Mind Freeze	65	1	2
6	Nevermind	100	0	3
7	Bleach	120	0	3
8	In Utero	90	0	3
9	The Main Room	250	0	4
10	Blue Room	550	0	5

Performances					
performance_id	artist_name	performance_date	room_id	price	tickets_available
1	Modest Mouse	07/23/2022	1	60	45
2	System of a Down	07/30/2022	2	70	10
3	James Taylor	08/06/2022	3	90.99	99

Instruments	
instrument_id	instrument_type
1	electric guitar
2	acoustic guitar
3	bass guitar
4	drum set

Artists				
artist_id	name	email	phone	genre
1	Modest Mouse	MM@gmail.com	1234567890	indie rock
2	System of a Down	sysdown@gmail.com	2345678901	progressive metal
3	James Taylor	jtaylor@aol.com	3456789012	folk
4	Stitched Up Heart	goth_girl@tragic.com	8005555555	Emo

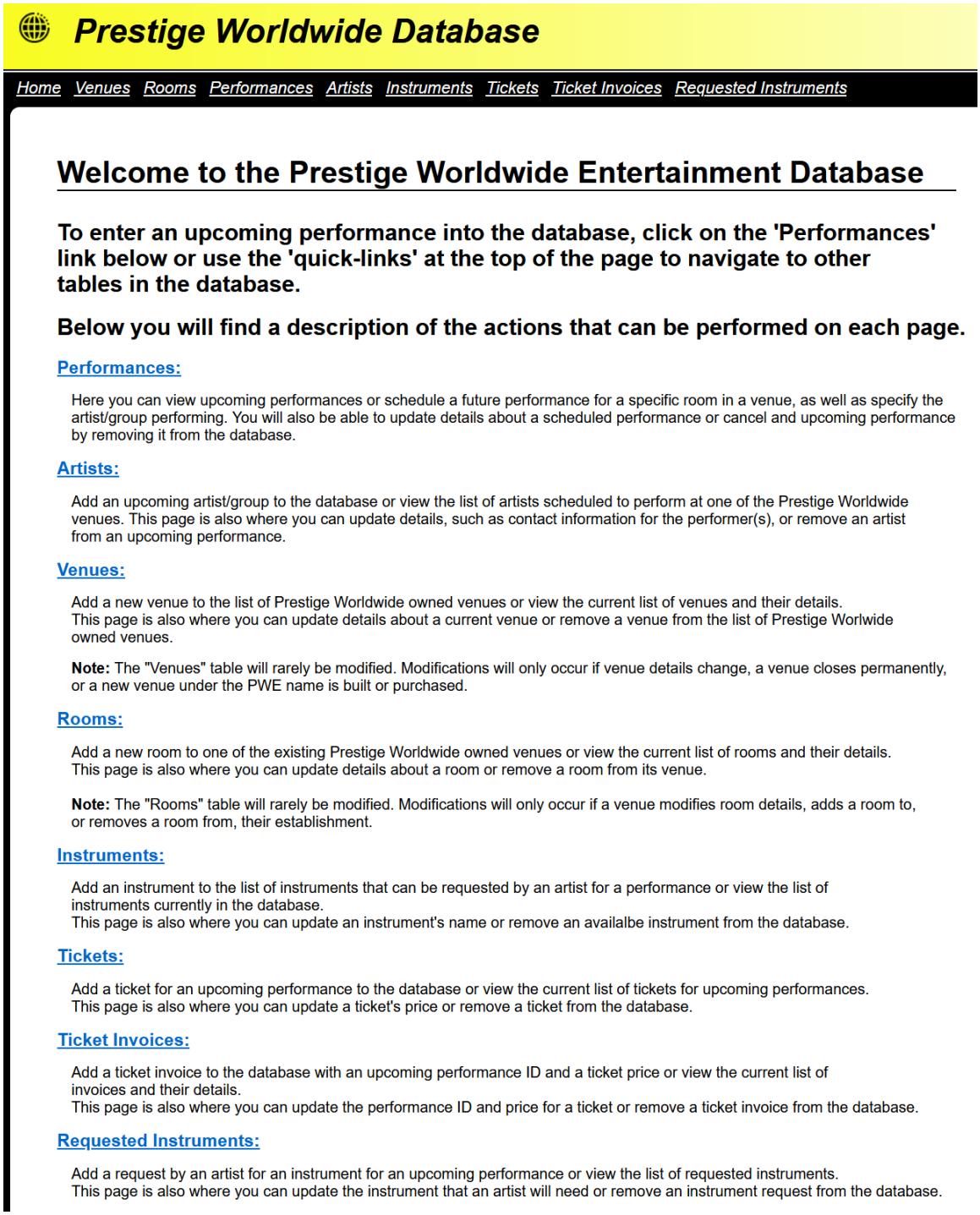
Tickets	
ticket_id	price
50	50.00
2	60.00
3	70.00
4	80.00
5	90.00
6	100.00
7	120.99

Ticket_Invoices					
event_id	Performer	Location	Room	Date	Price
1	Modest Mouse	Broken Spoke	Pony Room	2022-07-23	70
2	System of a Down	Broken Spoke	Oregon Trail	2022-07-30	60
3	James Taylor	Broken Spoke	The Clydesdale	2022-08-06	90

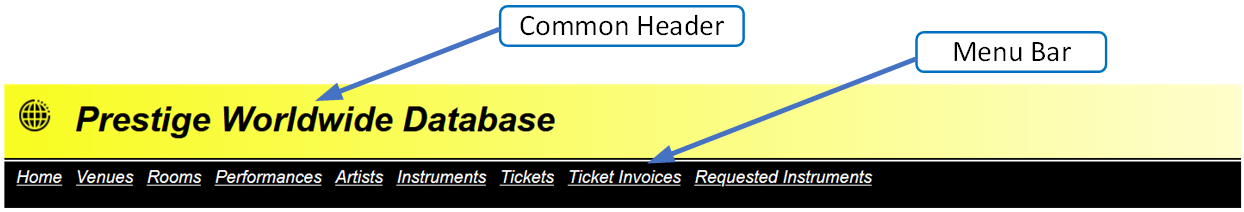
Artist_Needs_Instruments		
rental_id	Band	Gear
1	James Taylor	electric guitar
2	System of a Down	bass guitar
3	Modest Mouse	electric guitar

Screenshots

The user experience begins with a landing page that serves as a means of orientation and navigation. The menu bar across the top of the page will take the user to individual pages where they can perform the operations desired. Paragraphs below give a brief summary of the options available on each page as well as another convenient link to that specific page.

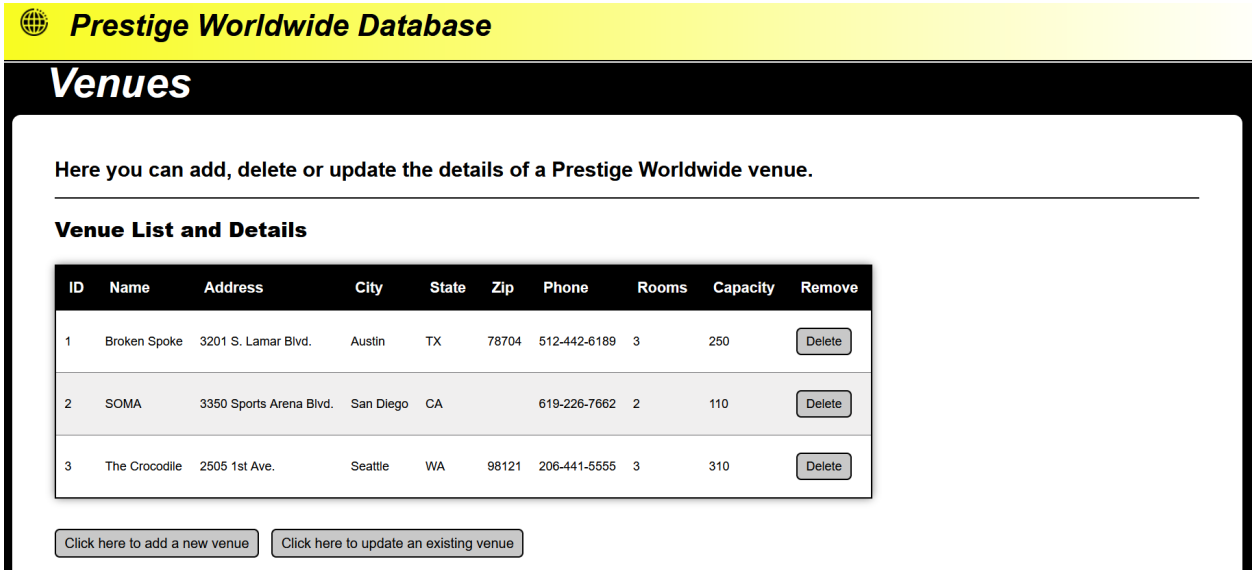


A common heading at the top of each page is provided to give the user familiarity with the layout when navigating and bringing the look of the site together.



Every database landing page has been enabled with CREATE/READ/UPDATE/DELETE CAPABILITY, as well as a search function for existing performances. Full CRUD implementation is displayed on a single page for each table, with CREATE and UPDATE interfaces hidden until called on by the user. Fields are added or updated using a variety of input fields, including text and number boxes, dynamically linked pull-down menus, and embedded calendars. DELETE buttons are embedded in each READ table entry.

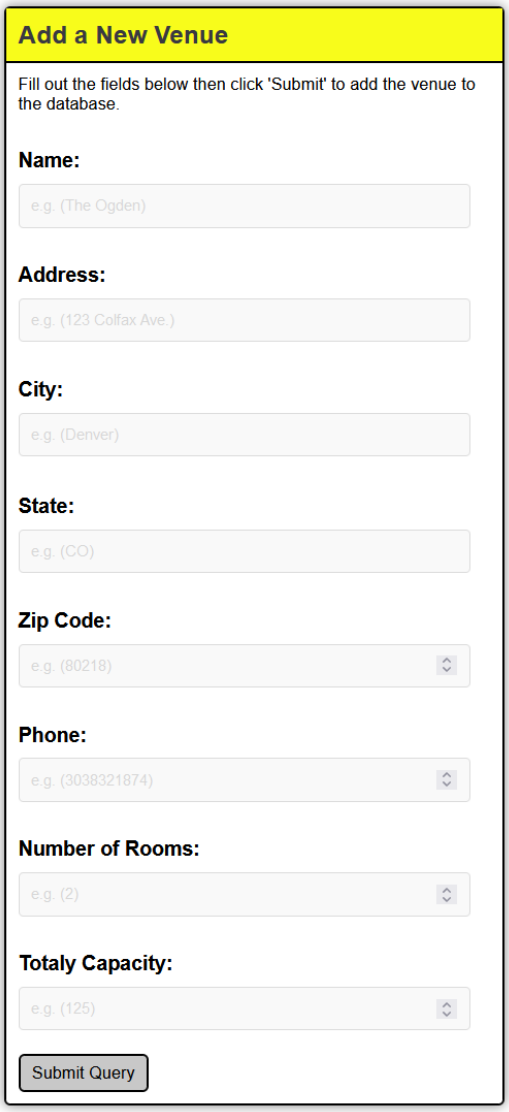
Venues



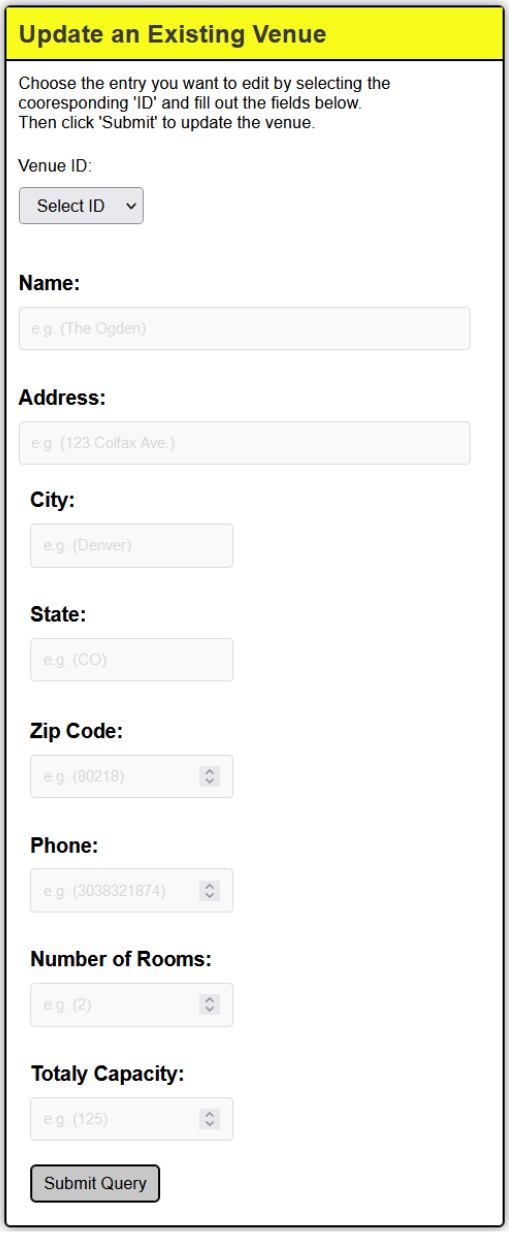
The Venues table contains most of the important details of each venue - name, address, number of rooms, and total capacity of the building. The top of the page displays the Read table of current entries with Delete buttons embedded for each. This is considered to be an object table since details will rarely be modified, but the delete button is present in case a venue is sold off.

The links below the read table display the Add form, the Update form, or both.

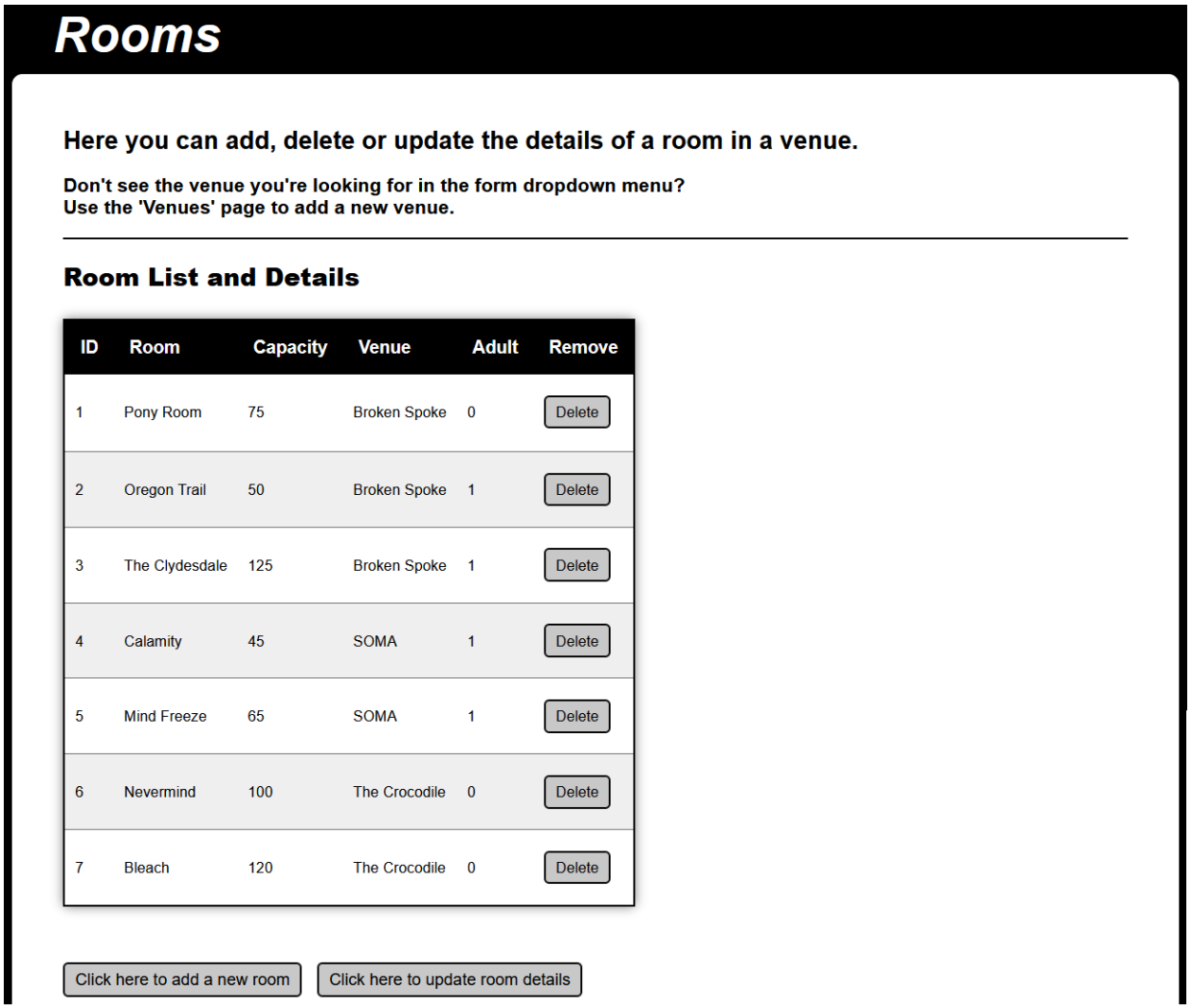
If a venue is added all information is entered via text/number boxes. The number of rooms is set to a default value of zero and updated as rooms are added. Total building capacity is determined by local fire code, and – along with the name - is the most likely entry to be updated. A pll-down menu is provided for choosing the venue to be updated, and all other details are filled in using the text boxes.



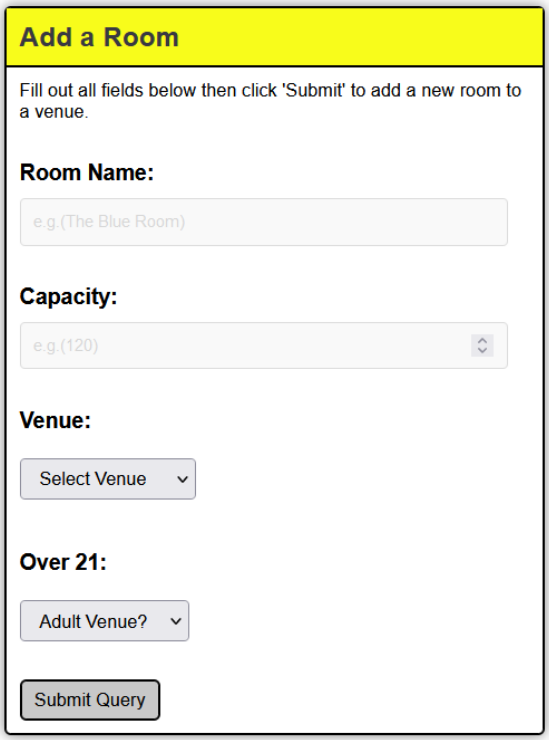
Click here to update an existing venue

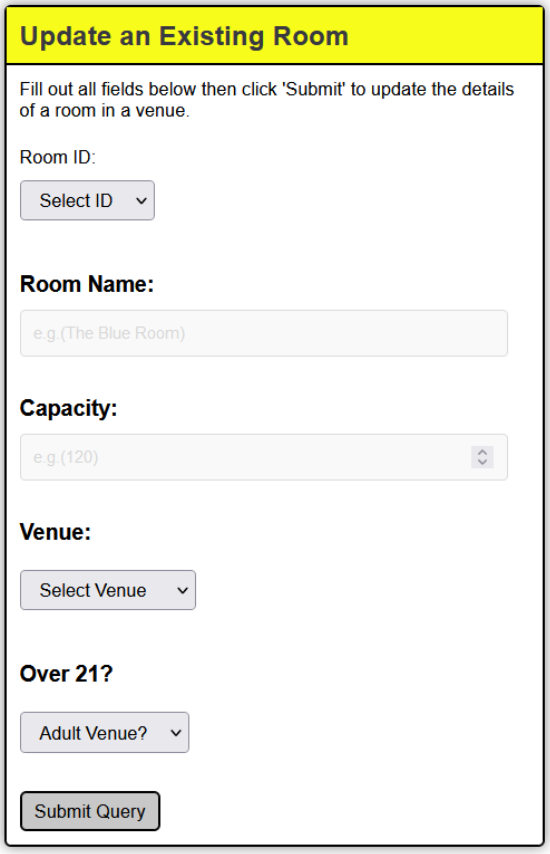


Rooms



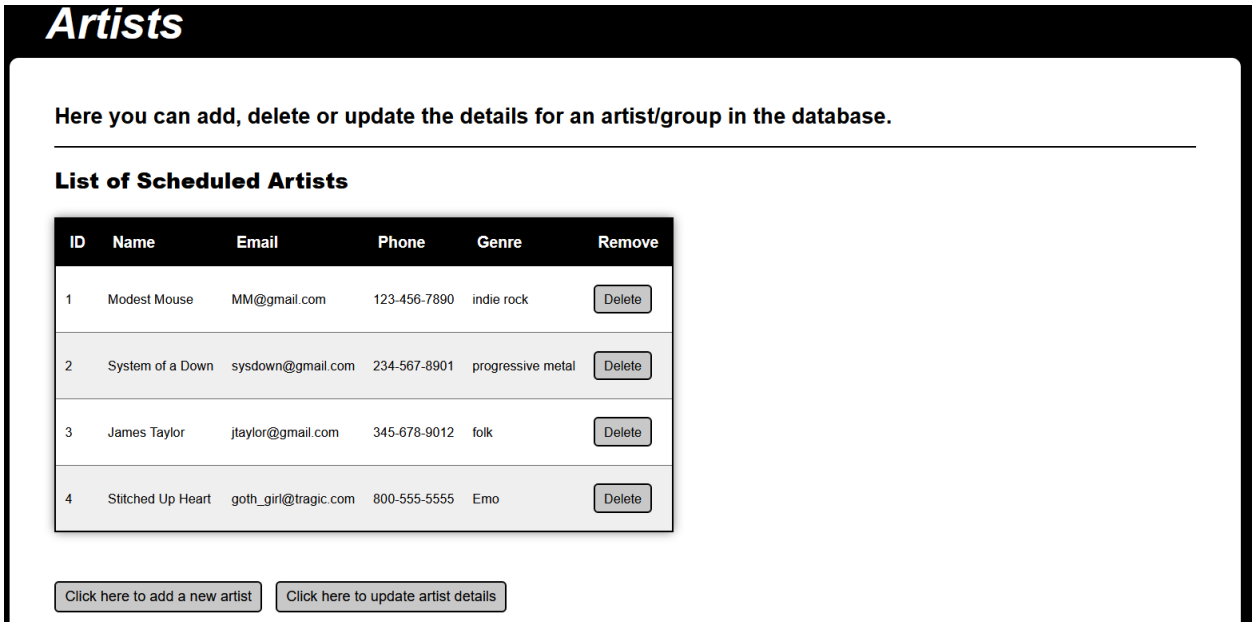
The Rooms table associates a particular room with a name and the venue in which it is located, as well as whether or not it is capable of holding an “over 21” event (alcohol sales). The Delete function is unlikely to be used except in the case of venue sales (or conversion to another function). Create and Update forms are provided in the links below the read table.



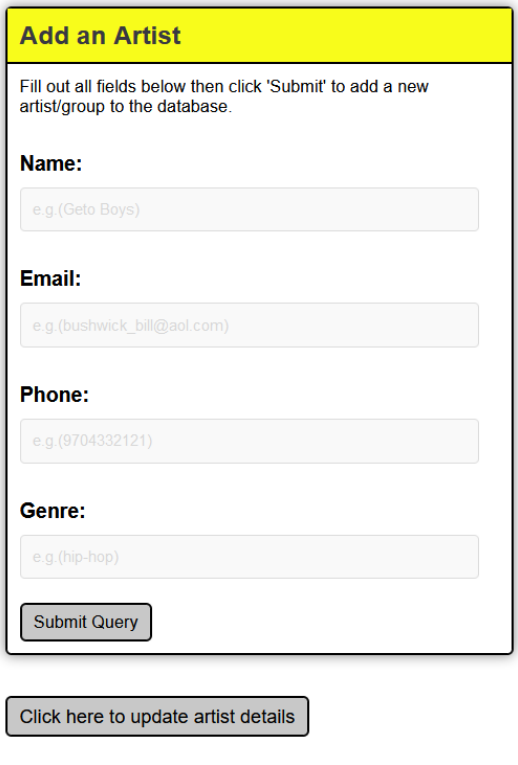


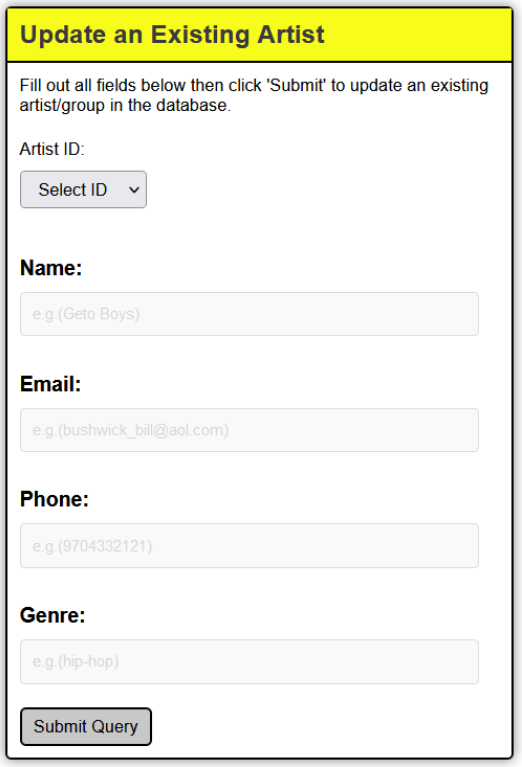
Room name and capacity are via text box. While still an object table, these values may be changed slightly more often than Venues - the name in an attempt to “keep it fresh” while room capacity can be changed in accordance with the fire code or the “feel” of the room. Venue would likely only be ever changed in the event of an error made in the initial entry. The adult venue may be changed if a ber were added or removed and a ‘Yes/No’ pull-down automatically converts to 1/0 values.

Artists



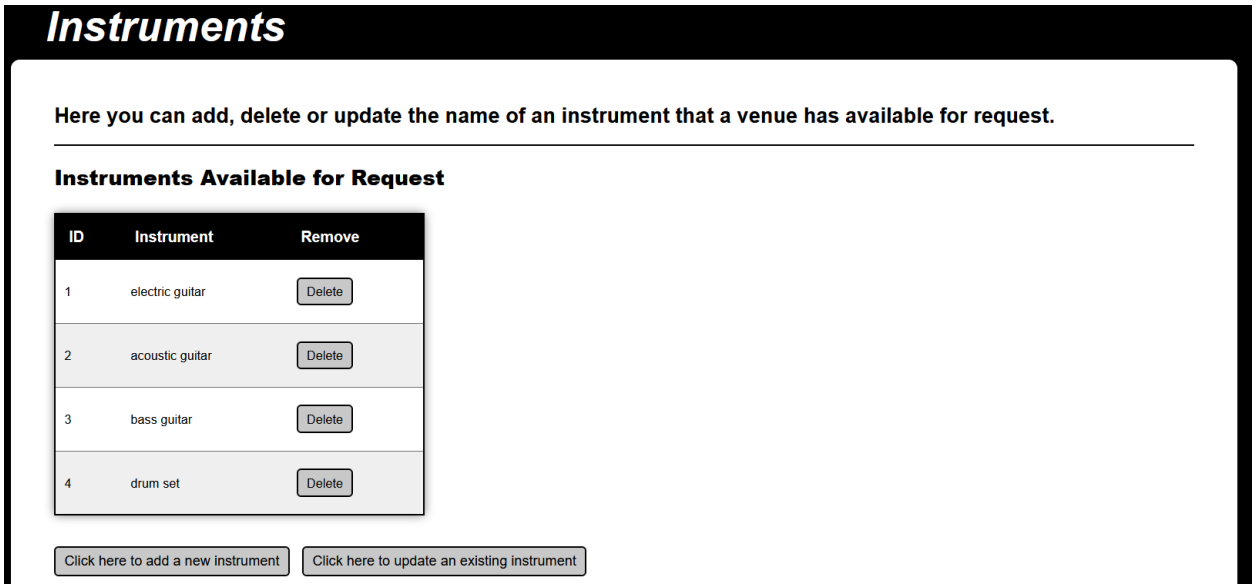
The Artists table maintains primary contact information for the artists so as not to risk loss if an event is canceled, as well as genre for help in placing a performance into a properly themed room. If an artist is permanently removed front he line-up (banned, for example), their information can be deleted. Create and Update forms are provided in the links below the read table.



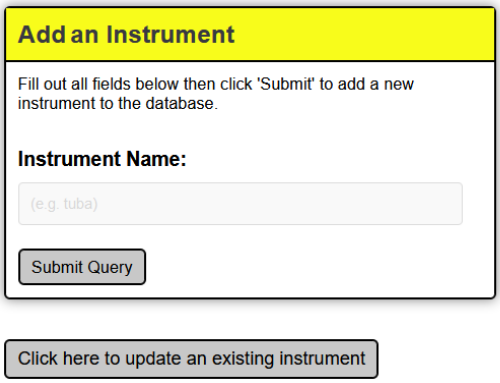


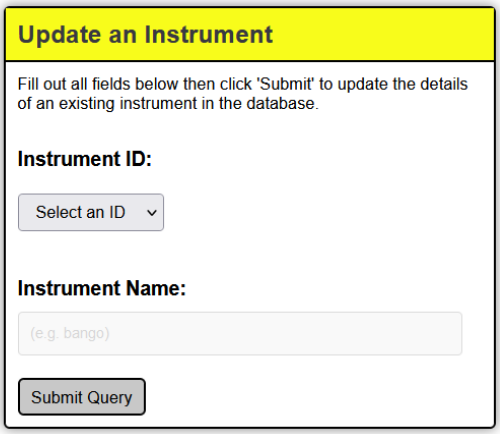
This is another object table, but may be somewhat more likely to be modified than Rooms or Venues as bands can be a bit more fluid. Any of the values in the artist table can be added or modified, including name (Lady Antebellum to Lady A, The Artist Formerly Known as Prince, etc.). A pull-down menu is provided for selection of the artist id.

Instruments



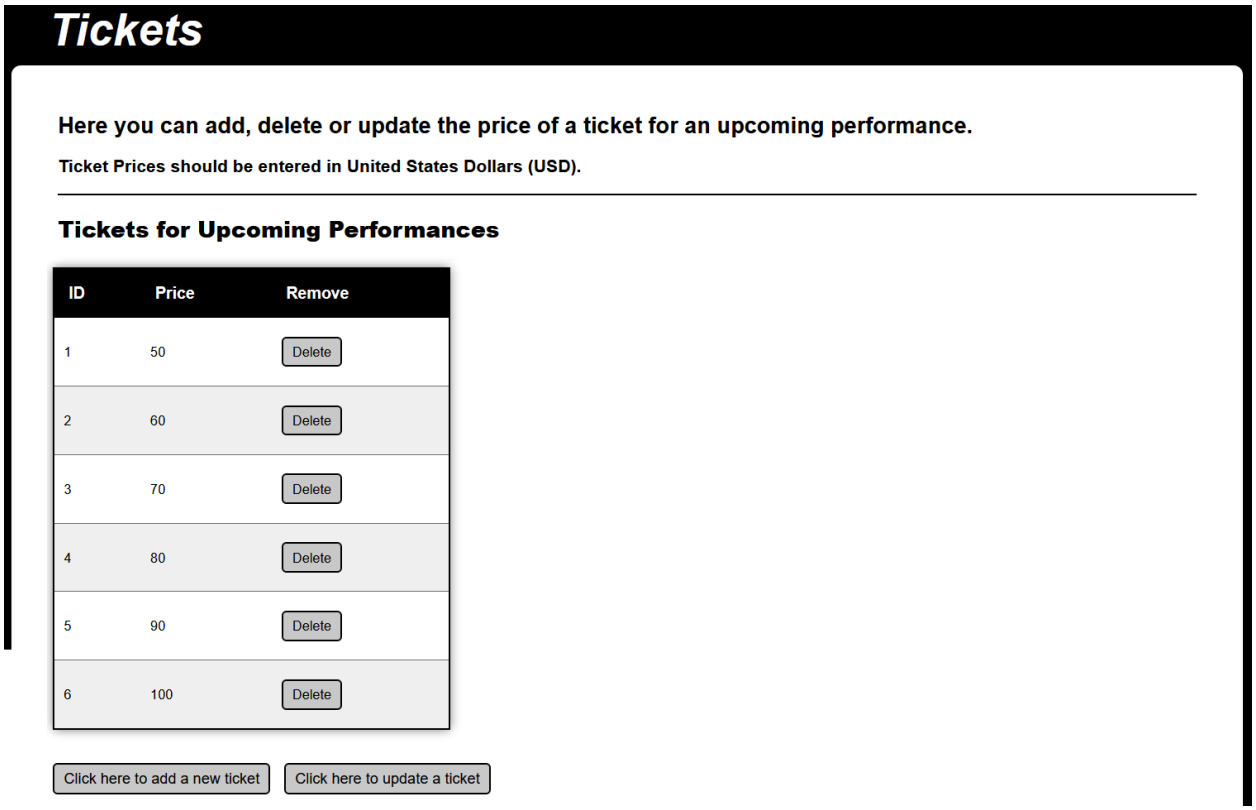
The Instruments table provides a list of the instruments that can be made available to an artist during their performance. These would be common to all venues so that their location would not need to be tracked. If an instrument becomes permanently unavailable, a Delete button is provided. Create and Update forms are provided in the links below the read table.



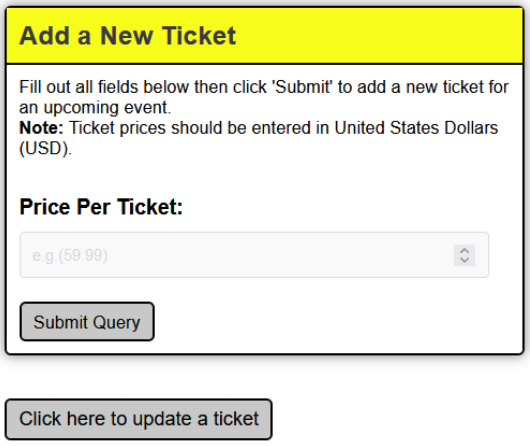


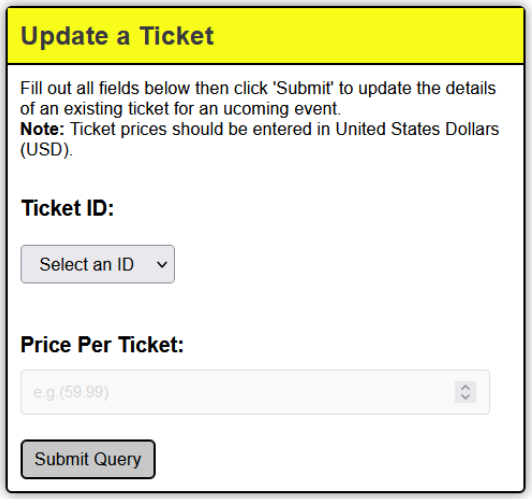
Adding an instrument is performed by simply filling in the text box provided and would be used as either more instruments become available or specific instruments are commonly requested. Updating this table could occur if more than one instrument of a specific type were available and a distinction needed to be made. For example, if two brands of electric guitar were available, each with a distinctive sound, then the first would need modification before entering the first. This would preserve transaction records that occurred before the change was made. A pull-down menu is provided for selection of the instrument id.

Tickets



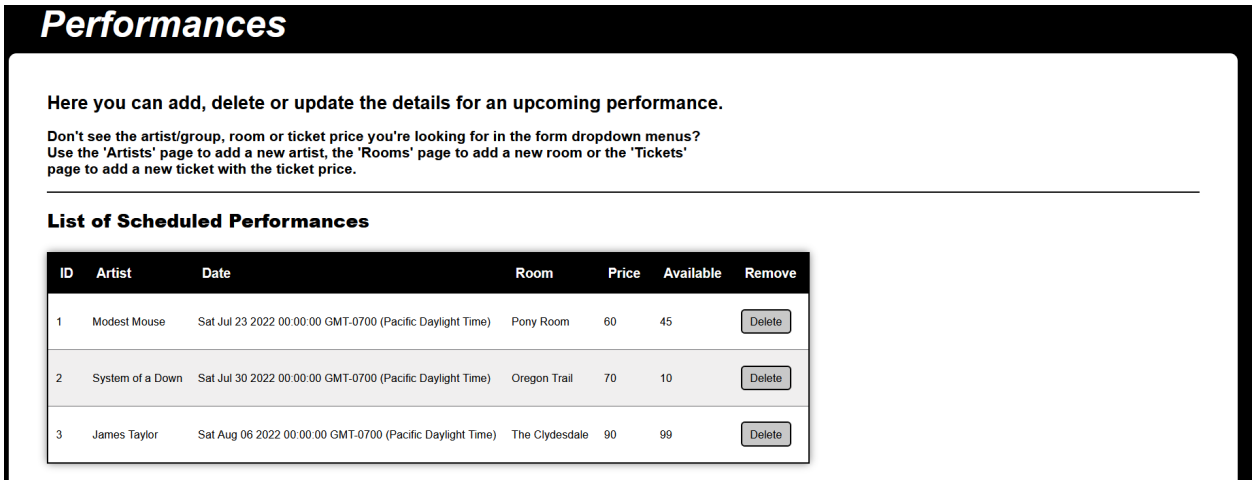
The Tickets table is a category table that sets a base price point for performances. A ticket price could be removed if, for example, inflation made it unlikely that “cheap” tickets would be sold in the future. Create and Update forms are provided in the links below the read table.



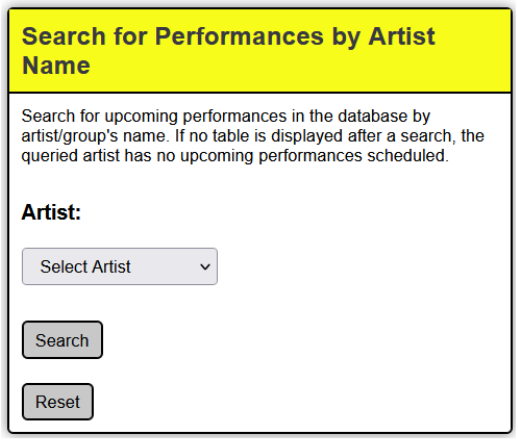


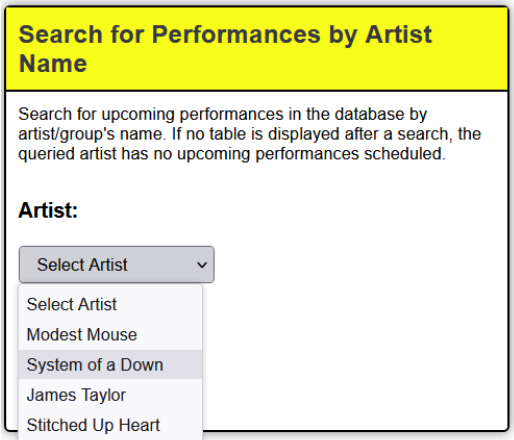
A pull-down menu is provided for selection of the ticket id.

Performances



Performances is an intersection table that brings Artists, Rooms, and Tickets together in one location and acts as an invoice for the company's hosted performances. From the read table at the top of the page, an employee can see all the performances that are scheduled for the company as a whole (across all venues). A key was added directly to Tickets in order to eliminate the need for a dummy invoice when setting the base price of the performance. As the number of performances grows, it may become difficult to find a specific performance from a complete table view, so a search function is provided to narrow performances to a specific artist selected from a pull-down menu. The process is shown below.

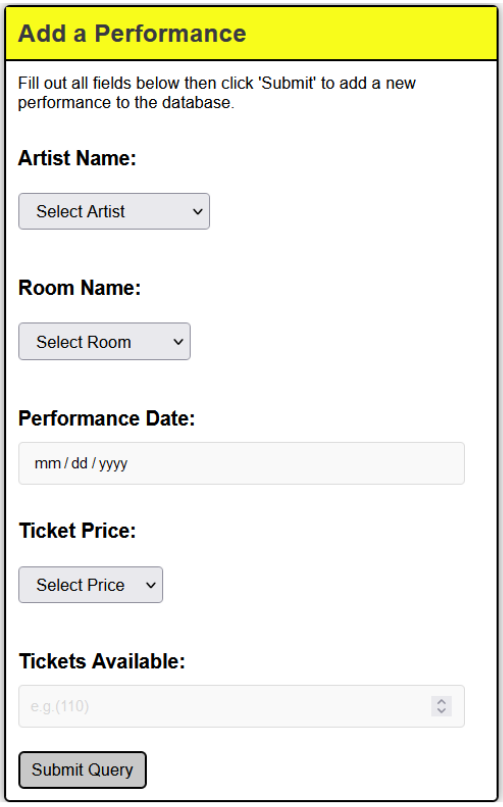


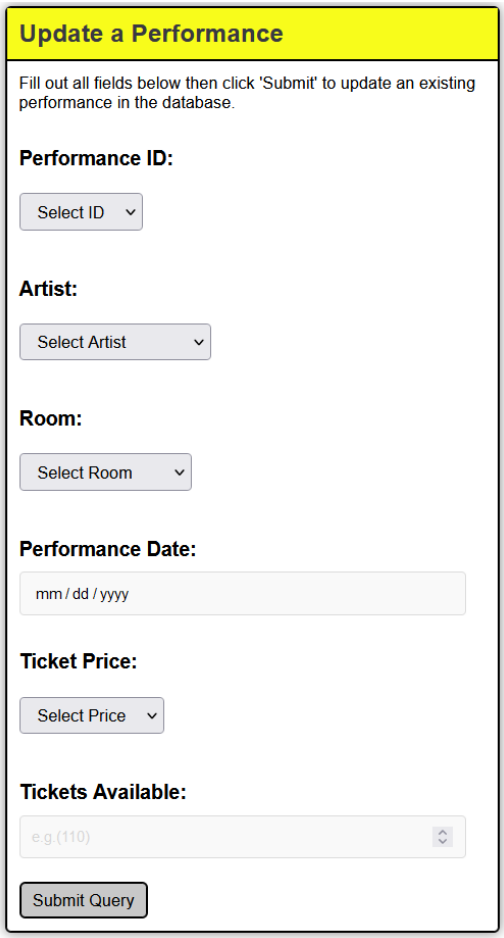


List of Scheduled Performances

ID	Artist	Date	Room	Price	Available	Remove
2	System of a Down	Sat Jul 30 2022 00:00:00 GMT-0700 (Pacific Daylight Time)	Oregon Trail	70	10	Delete

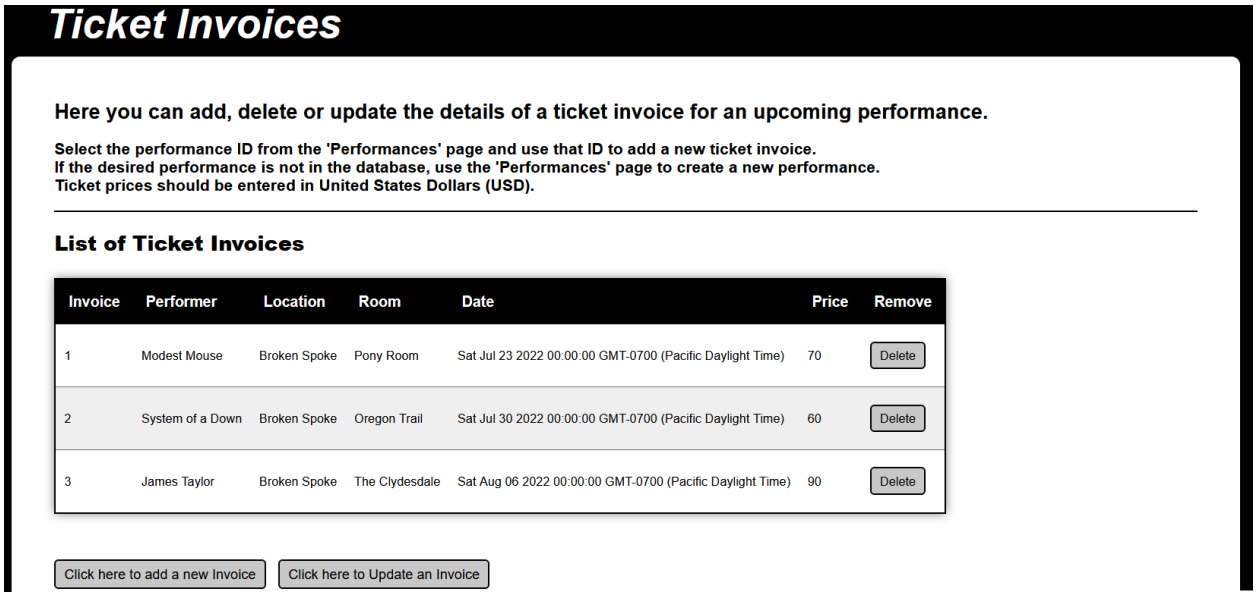
Create and Update forms are provided in the links below the read table and search bar.



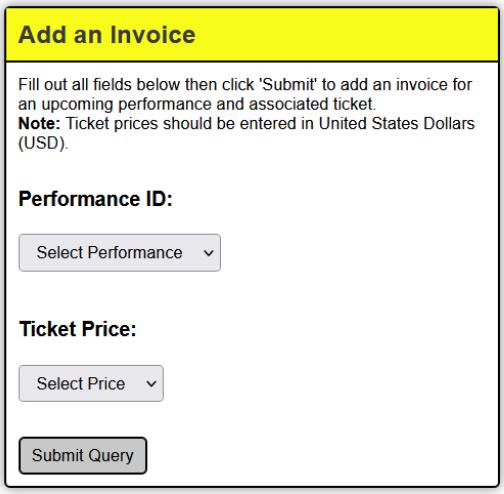


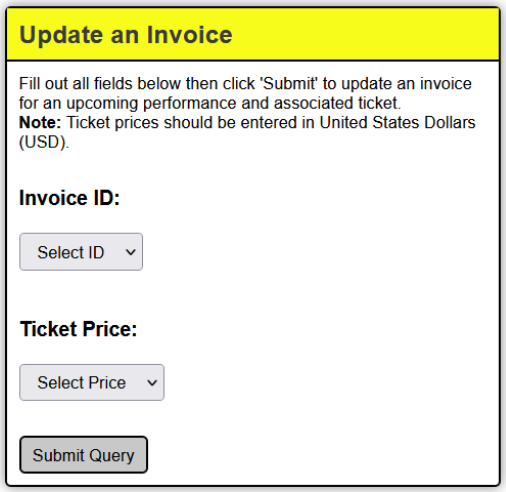
It is expected that performances will be added regularly and modified on occasion, so steps were taken to ensure that the process is as simple as possible. Performances are added from previously entered artists into previously entered rooms using previously authorized prices (dynamically populated pull-down menus). The performance date can be selected or modified from a pop-up calendar. A pull-down menu is provided for selection of the performance id.

Invoices



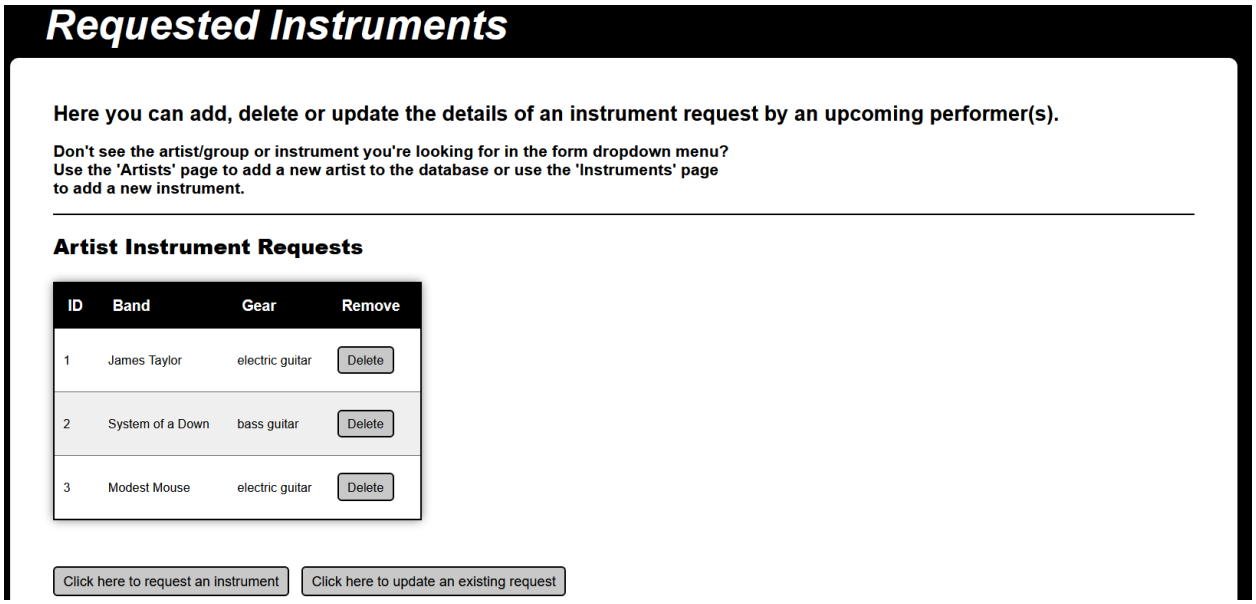
Invoices is an intersection table between the performances scheduled and the customers purchasing tickets. In this assignment, customer information is not tracked, so the table acts as a simple invoice table for tickets sold to a particular performance. This way, multiple tickets can be sold to a particular performance to different customers and at non-standard price points (based on demand, promotions, etc.). It was decided that a single invoice represents an individual ticket sale so that the Delete function represents a complete refund of the transaction. Create and Update forms are provided in the links below the read table.



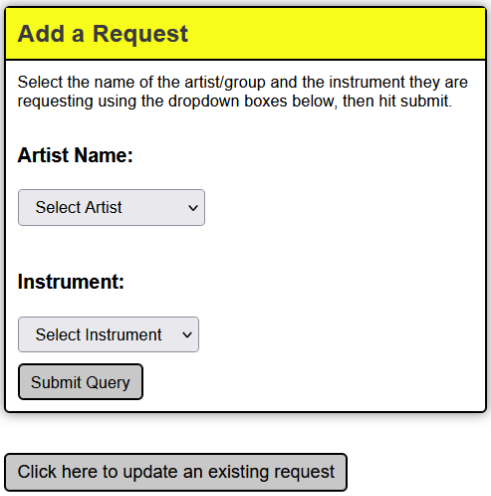


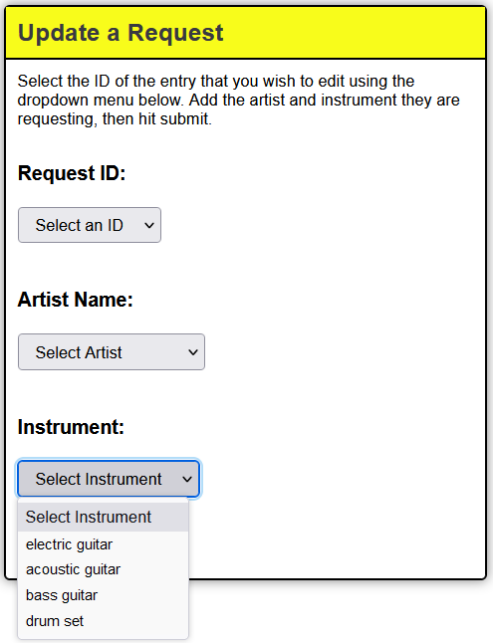
Adding an invoice is simplified for fast processing A pull-down menu is provided for selection of the performance id and ticket price. Updates to the invoice are limited to modification of the ticket price and are performed using pull-down menus.

Requested Instruments



Requested Instruments is an intersection table that serves as an invoice to track the instruments that are requested by a specific artist. It is assumed that this artist will need this equipment for all performances if more than one are scheduled. This could be modified to include a performance date in the future. Currently, delete buttons are provided if the instrument is no longer required by the artist, but some utilization information will be lost in the process (not currently tracked). Create and Update forms are provided in the links below the read table.





Instrument requests are handled using pull-down menus. For modifications to a request, a pull-down menu is provided for selection of the request id.