MiniTicketer

Assessment Project for Job Application @ IntegraDev

Proposal

The proposed project, named MiniTicketer, is a simple yet intuitive venue ticketing system which allows customers to book to see a short *movie* at a *specific venue* at a *specific date & time* for where there is a defined *seat allocation*. For the purposes of this project, the venues consist of short movie screenings at certain times over a specific date period. Models will now be looked at in more detail.

Models

Movie

- Movies are unique; there can not be more than one of the same movie.
- Movies have a name.
- Movies have a running time (duration in minutes).
- Movies have a release year.
- Movies have a specific genre.
- Movies are shown at **screenings**.
- Other aspects can be included but are not the focus of what is being assessed.

Venue

- Venues are unique; there can not be more than one of the same venue.
- Multiple **screenings** can occur at the same venue.
- Venues are identified by number, similar to cinema rooms.
- Venues have a specific amount of seating rows.
- Venues have a specific amount of seating columns.
- Other aspects can be included but are not the focus of what is being assessed.

Screening

- Customers can purchase a **ticket** to see a screening.
- A specific **movie** is shown at the screening.
- Screenings occur at a specific **venue**.
- Screenings happen over a specific date period. (start, end dates)
- Multiple screenings of the same **movie** can occur, just not in the same date period.
- Screenings can happen on specific days at specific times.
    - Once a screening's date period has finished, it can no longer be booked at.
    - Each screening must have a unique time period and inherently, a unique movie in this time period; this forms the primary key.
    - Can be uniquely identified by an ID number.

Ticket

- Ticket is purchased by a customer to see a **screening**.
- Ticket is purchased by a customer who is identified by a *username*.
    - There will be no separate model for users, to keep the complexity at a minimum.
- Ticket purchased for a **screening** must have a *valid* date & time.
- Ticket purchased for a **screening** must have a *valid* seat allocation.
- Uniquely identified by screening ID number, date, time, and seat selection.

Let's quickly summarise the four *main* data models in this program.

Movies are unique, short films that are shown a **screening**. **Screenings** can show the same **movie** more than once, and they happen at a **venue**. **Screenings** occur during a specified date period, on specified days, at specified times. *Validation must occur to make sure screenings do not conflict or overlap. This will be done by the program, and conflicts will be automatically corrected and logged*. **Venues** are where **screenings** take place, and each venue can have unique seat allocation dimensions (e.g. 10 rows, 20 columns = 400 total seats). Customers purchase a **ticket** to see a **screening** at a date and time that falls into the specified range defined by the **screening**, and can pick a seat allocation that falls into the specified dimensions defined by the **venue**.

The only thing that can be changed by the end-user, the customer, is the **ticketing**. Movies, venues, and screenings should all be set-up prior to a customer using the program to acquire a ticket. As this is only a small assessment project, sample movies, genres, venues, and screenings will be automatically provided when the data store is created, though a database operator can use a web tool such as *phpMyAdmin* to add more movies, venues, and screenings where applicable.

Model Assumptions/Validation

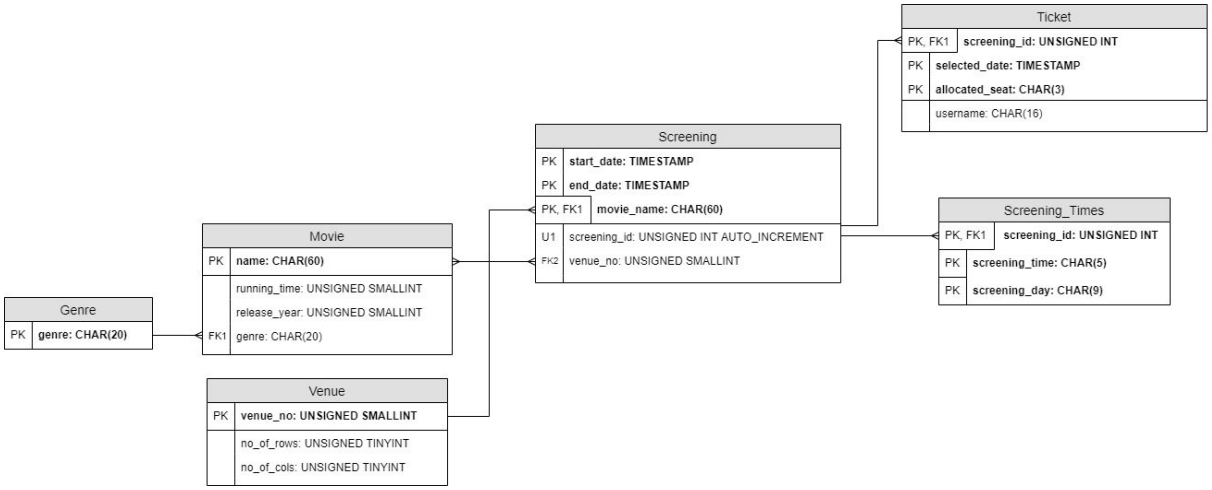
There are obviously countless more potential models that could be added (e.g. movie actors, actual user logins, venue locations) to make the application more complex & intuitive, but I feel that the current five are enough to adequately demonstrate my design and development skills. The *venue* attribute in the *screening* model is not part of the primary key as an assumption for the *screening* model is that a movie can only be screened once for the specified date period, and cannot have overlapping screenings *even if they occur at different venues*.

Validation of screenings are done primarily through the program, though correct data types and unique values will be enforced by the tables. Here are more assumptions, and validations that must be done when loading in screenings and designing the MySQL tables:

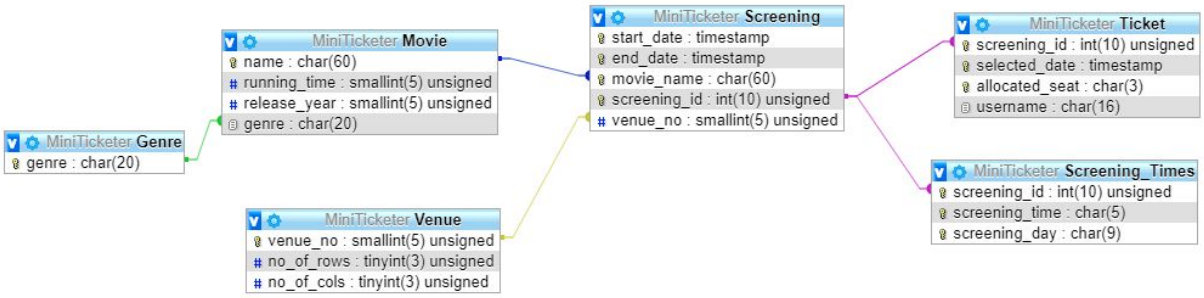
- The start date must not be after the end date. This can be enforced using a CHECK condition using SQL.

- The available days and times for a screening should not be duplicated, so they are delegated to their own table named *Screening_Times* to enforce this. Wednesday has the most characters, so the maximum character value is 9. The character count for military time is always 5.
- Screenings that show the same movie cannot have overlapping date ranges, if this is the case, the most recent one will be favoured.
- Screenings times should not overlap one another.
- Since tickets are created using only-valid data, they are assumed to always have valid dates & times, though they will be checked on each startup and removed if invalid:
    - Tickets sharing the same screening time should not have duplicate seats.
- Attributes in tables should have checks & default values where applicable, these can be viewed in the table creation SQL statements.

SQL Table UML Diagram



A higher resolution UML will be included in the code package upon completion. This table structure conforms to first, second, and third normal forms. It went through a couple of stages where transient dependencies between tables were removed in order to achieve 3NF. *SQL Statements* to create the tables will be included in the code package, though the program will automatically generate them if they do not exist. They have been tested and confirmed to work, though the CHECK conditions will not be enforced in MySQL versions before 8.0. It is recommended that you use MySQL 8.0 or later.



Here is the same table structure, presented by phpMyAdmin after running the table creation script.

Views/Screens

Initial Control Flow:
When the user first loads the program, it will attempt to establish a database connection to a MySQL server addressed at 127.0.0.1 (localhost) on default port 3306 with the credentials user=root and password=password, as this is the default configuration. If the connection fails, the user can change these settings through the configuration file generated in the same directory as where the JAR was executed. The program will not load until a successful connection is established with the MySQL server. If at any point connection is lost, the program should either stall until successful reconnection or simply close; this is not considered to be a main feature of the project but I will see how much time I get.

Main (Booking) Screen
After successful connection, the user will be greeted to a menu where all currently shown screenings are listed. The user can filter available screenings by genre. The user will be able to click on a screening to begin the booking sequence. The user should also be able to view a list of their booked tickets under their *username*, grouped by each unique screening time. There should be options to either delete tickets individually or bulk delete tickets for a whole screening. The booking sequence begins with selecting a time/date slot and entering the number of attendees. After this has been successfully done, the user is then brought to the allocation screen to select seats, either manually or automatically using a button. Note that the user should not be able to continue the booking process for a time slot that has been completely sold out.

Allocation (Seat Selection) Screen

Following up from a successful selection of a screening time from the *booking screen*, the user must then allocate all attendee seats by clicking on them. Clicking on a seat twice un-selects it. Allocation is done in a queue-like fashion, where the first selected seat goes to the first ticket holder, and so on. The dimensions of the seating grids is determined by the venue at which the screening will be shown. Rows are labelled by a letter, and columns are labelled by a number (e.g. seat E11). The maximum amount of rows should be 26, and the maximum amount of columns should be 99, realistically. After successfully selecting the seats, the user will be brought to a final confirmation screen. Seats that have already been booked should not be available for selection. A button should exist that automatically picks the first available seats for considerably large bookings.

Confirmation Screen

This screen serves the simple purpose of summarising the selections the user has made. This includes the screening, the day, the time, the number of attendees, and their specific seating allocations. To commit the booking, the user must supply a *username* to store the booking under so that their tickets can be managed later if needed. A receipt is shown with the booking summary once the user submits. Note that the user can return to any of the previous screens at any time.

Ticket Viewing Screen

Existing tickets under a username can be viewed and deleted using this screen. Tickets are sorted by movie (not screening, though it results in the exact same thing), and can either be deleted individually or in bulk per movie.

Screen Designs

Rough screen designs were sketched on paper.