README

The application is written in Java and works on IntelliJ. The additional library was mysql-connector-8.3.0. The application runs in the terminal. To start it, simply hit the run button that runs the main function for the Main file. Then, follow the application's instructions to log in or register an account. The main menu will pop up after logging in to the application. Normal users will be able to buy, sell, modify, and delete tickets for specific events. Host users will be able to create/delete events, create/delete stadiums, and create/delete stores in the stadium. To exit the program, return to the main menu and quit.

Technical Specifications

The project uses Java and MySQL workbench. We used packages from Java, including:

java.sql.*;

java.time.LocalDate;

java.util.*;

java.util.regex.Pattern;

Textual description of the current conceptual design

There are be 8 Entities in the UML diagram.

The event has an ID, name, date, and type. Type is an Enum, which could be one of "Sport," "Concert," "Art & Theater," "Family," or "Other."

The Stadium has an ID, name, capacity, and address that serves as the alternate key. An event must be hosted in one stadium, but a stadium can host many events.

A Seat is a weak entity that has a section, row, number, and type. Types include "General Admission," "Box Seats," "Club Seats," "Suites," "Accessible Seats," "Standing Areas," and "Other." A Seat must be in one stadium, and a stadium must have at least one seat.

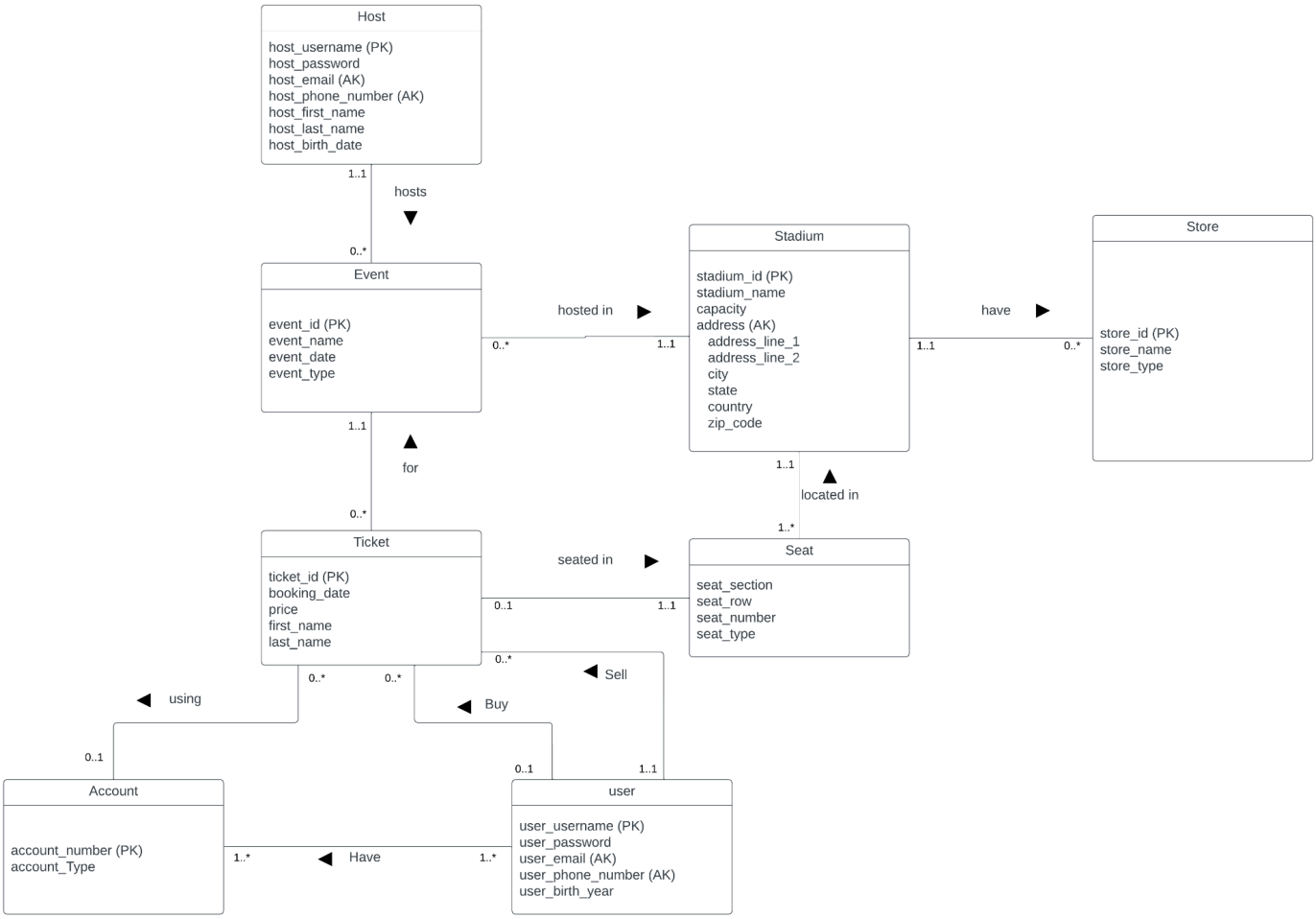
A store entity has an ID, name, and type. The type could be one of "Merchandise," "Food and Beverage," "Souvenir," "Fan Shops," and "Other." A stadium can have many stores, but a store can only be in one stadium.

A Ticket entity has an ID, booking date, price, and the ticket holder’s name. It must be related to one and only one seat and event. One event can have many tickets, and one seat is related to one and only one ticket for that event.

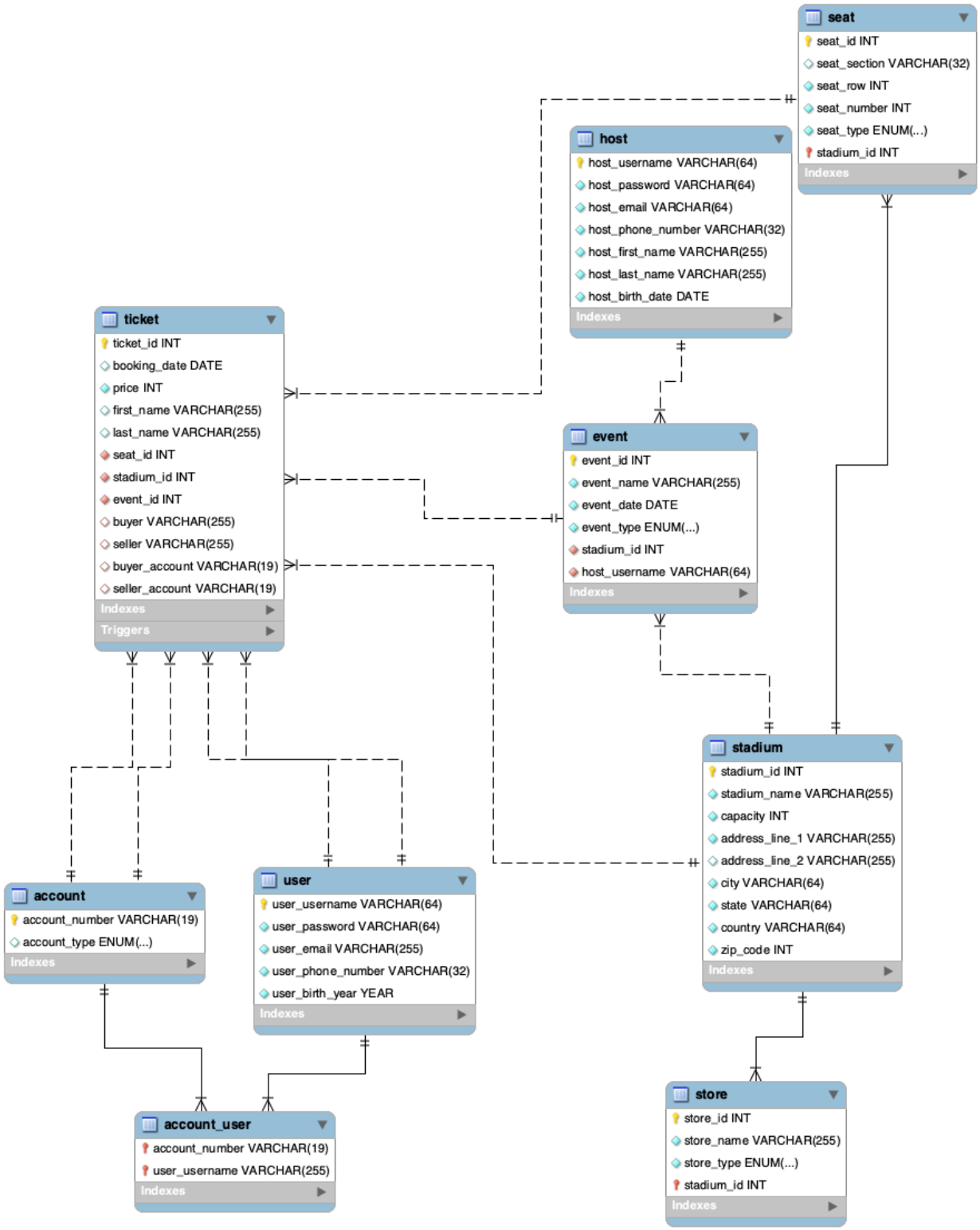
A user has an ID, username, password, email, phone number, and year of birth. A user can buy many tickets or sell many tickets, but a ticket can only have one buyer and seller as a user at a time.

A Host has a username, password, email, phone number, name, and birthdate. A host can host many events, and an event should have at least one host. Both host and user and login through username, email, or phone number combined with tier password.

The account entity has an account number and type, which is one of "Checking Account," "Savings Account," "Credit Card Account," and "Debit Card Account." A user must have an account related to them.

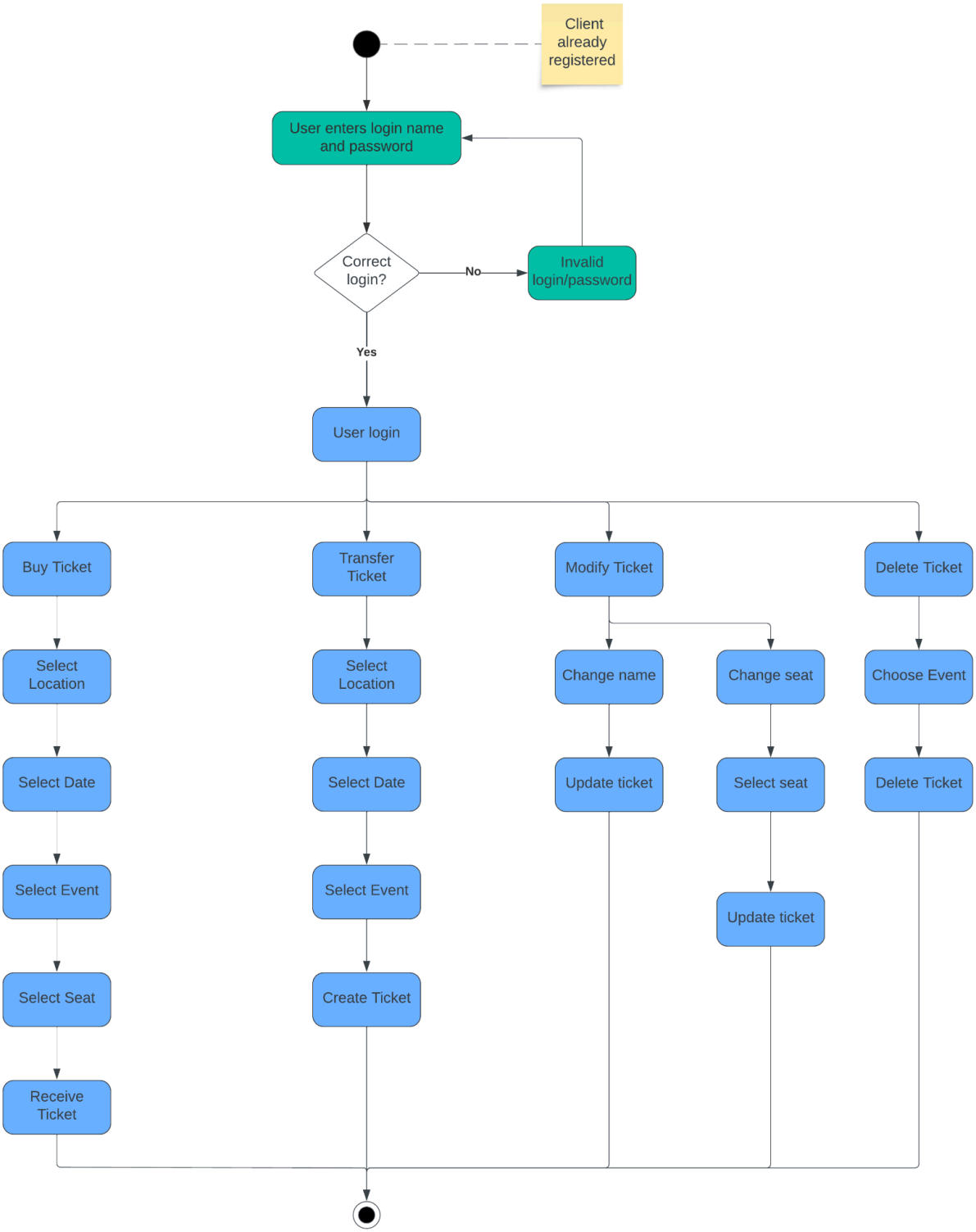


The logical design for the submitted database schema

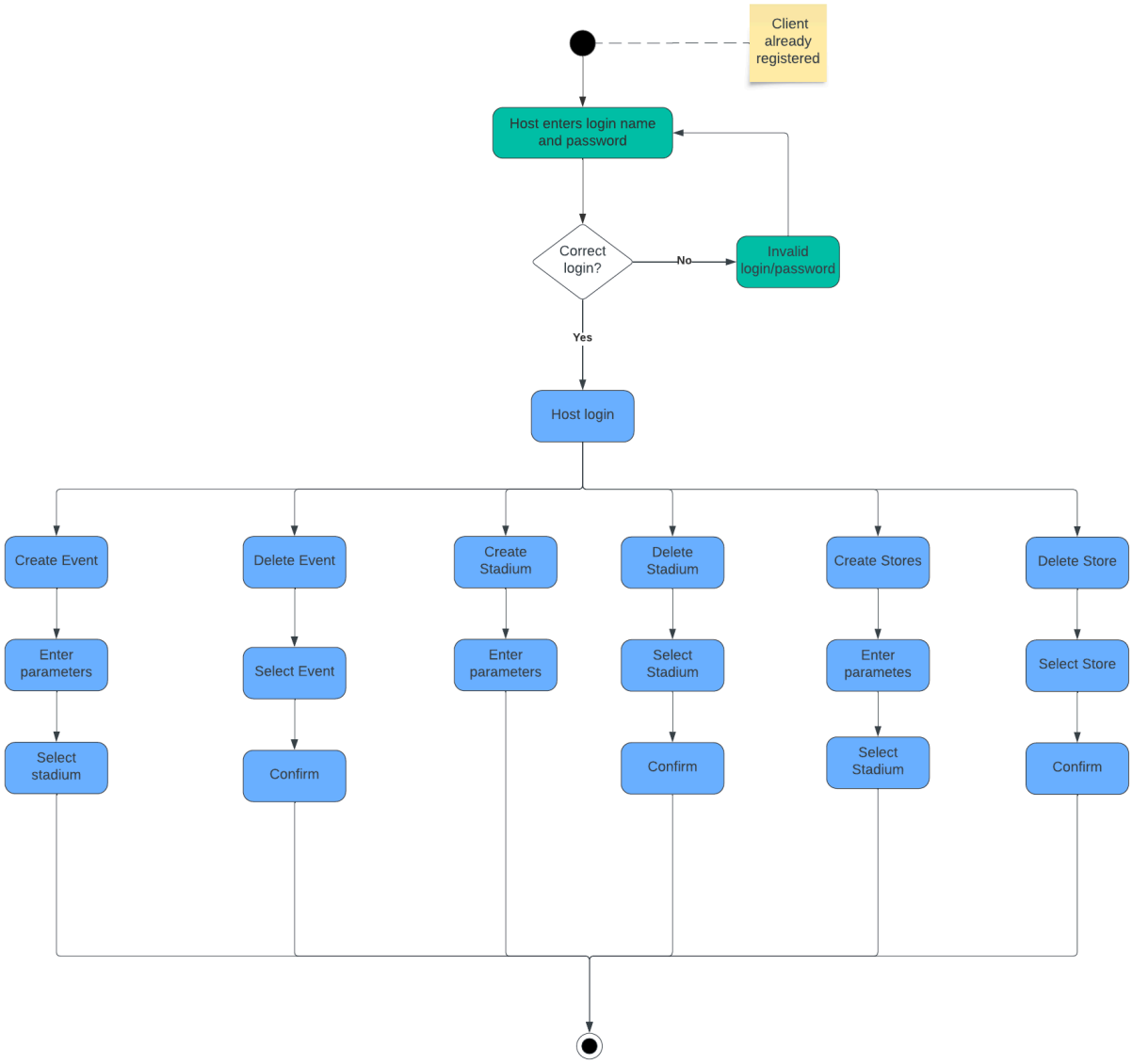


Final user flow of the system and commands

USER Activity Diagram:



HOST Activity Diagram:



Provide a description of any code not working in this section, and document any bugs you have encountered.

All functions seem to be implemented and are working as expected.

Lessons Learned

By integrating MySQL for robust data handling and Java for application logic, we developed a deep understanding of database architecture, SQL querying, and JDBC for database

connectivity. This experience enhanced our ability to design normalized database schemas and implement efficient data retrieval strategies. Additionally, working with Java deepened our skills in object-oriented programming and error handling. We were able to meet weekly and create and maintain a timeline.

At the start of the project, we had an availability status field for seats because we thought it would keep information and help determine if a user could buy a ticket. However, we ended up deleting the field because our database structure already allowed us to assign the role of keeping this information to the ticket table itself.

Currently, our database has a store entity, which is largely unexplored. We can expand upon our application to give the user greater access to viewing or interacting with the restaurants or use a different entity entirely.

Future work

This application enables users to effortlessly browse, reserve, purchase, and sell tickets for their favorite events at stadiums. Stadium Seat Booker aims to enhance the event-going experience for sports enthusiasts.

We are not able to implement a GUI and website for this database skeleton and have multiple users interact with the application. To do this we should host our database on a server rather than storing and editing the information locally. We could improve on the search engine to include more filters and autocorrect slight errors to match with the database information.

In addition, some specific functions, such as creating events or stadiums, limit the user to having square stadiums. It would be better to improve the amount of flexibility the user has in the future.