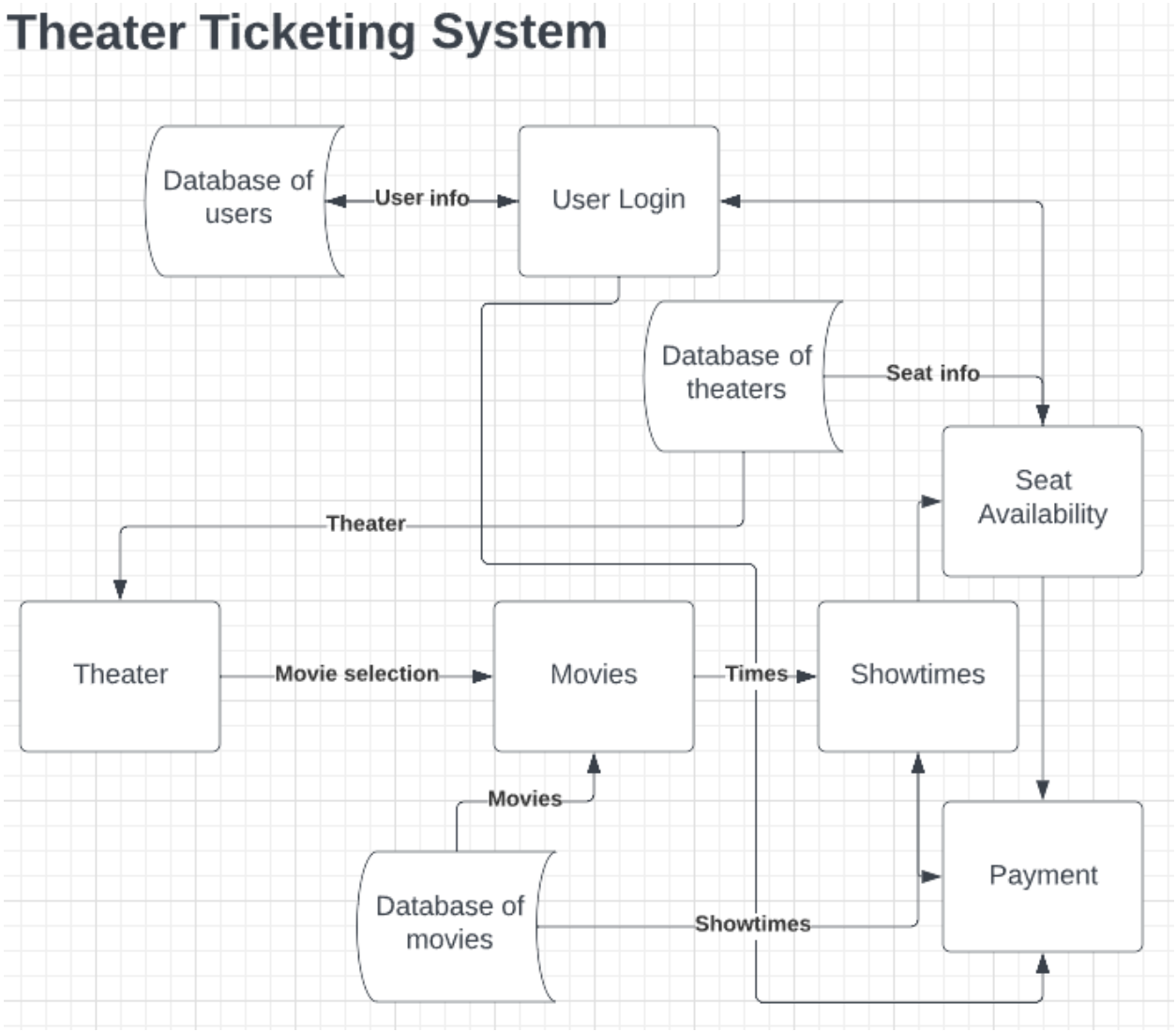
T.T.S Software Design 2.0

Presented by: Robert Albon

Software Architecture Diagram

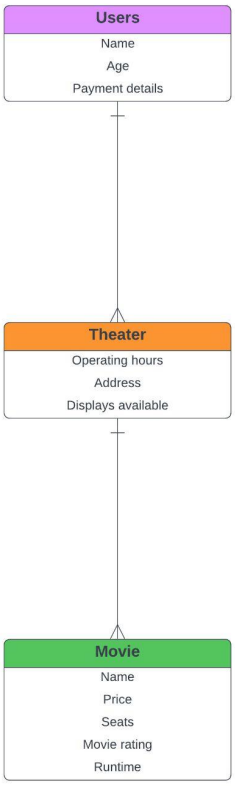


The software architecture diagram displays how an interaction between a user and the software would most take place; the SWA will show how a user reaches their desired end goal through the use of the software. The software employs the use of few databases to help with its overall use by a user.

Data Management Strategy

Regarding the data management strategy for the T.T.S, I believe that SQL would be the most effective and is best suited for the type of program that needs to be built. As displayed in the aforementioned SWA diagram and description the T.T.S would employ the use of a total of three databases including, movie DB, user DB, and a theater DB.

T.T.S Database Diagram (SQL)



Data Dictionary (Users)	Data Dictionary (Theater)	Data Dictionary (Movie)
Name: Name *Description:* T.T.S user *Type:* Attribute/String *Name:* Age *Description:* User's age *Type:* Attribute/Integer *Length Min:* 1 *Length Max:* 3 *Range of values:* 0-200 *Name:* Payment details *Description:* User saved/current payment details *Type:* String *Length Min:* 16 *Length Max:* 32	*Name:* Theater *Description:* Theater time/location *Type:* Table *Primary Index:* Address *Secondary Index:* Operating hours *Name:* Address *Description:* Address of theater *Type:* Attribute/Alpha *Length Min:* 5 *Length Max:* 35 Name: Operating hours Description: Hours of operation for theater Type: Attribute/String Format: HH:MM *Name:* Displays available *Description:* Amount of displays at theater *Type:* Attribute/Integer *Range of values:* 0-100	*Name:* Movie *Description:* Movie attributes *Type:* Table *Primary Index:* Name *Secondary Index:* Runtime *Third Index:* Seats *Name:* Name *Description:* Name of movie *Type:* Attribute/Alpha *Length Min:* 1 *Length Max:* 20 *Name:* Runtime *Description:* Length of movie in minutes *Type:* Attribute/Integer *Range of values:* 0-240 *Name:* Seats *Description:* Number of seats for movie *Type:* Attribute/Integer *Range of values:* 0-200 *Name:* Movie rating *Description:* Rating of the designated movie *Type:* Attribute/String *Length Min:* 1 *Length Max:* 1 *Name:* Price *Description:* Price of movie *Type:* Attribute/String *Format:* Dollars:Cents

---

This SQL diagram demonstrates the relationship between the entities at work with the T.T.S and how it correlates to the overall software architecture.

The primary reason that SQL was chosen is due to the overall efficiency of its use, especially in relation to heavy transactional use as well as its fit with simpler attribute architecture (i.e. strings, integers, etc.). Additional reasons that SQL was chosen in favor of a non-SQL design:

- Given the nature of the program, and given the functionality of non-SQL data management, the software will be heavily centered around transactions between a user and the system, thus the use of an SQL design is preferred.

- Furthermore, SQL holds an advantage over a non-SQL design in terms of its consistency as well as durability, which would prove to be beneficial for the software that is being designed.

- However, non-SQL holds the advantage of ease of access especially in terms of data insertions, whereas SQL databases require you to make changes across data.

- SQL databases also have an advantage over a non-SQL design due to the fact that SQL databases are relational databases whereas a non-SQL prioritizes non-relational databases.

- non-SQL classifications are also primarily better suited for a variety of uses, depending on the type of data being stored, whereas SQL databases are open/closed source in general.