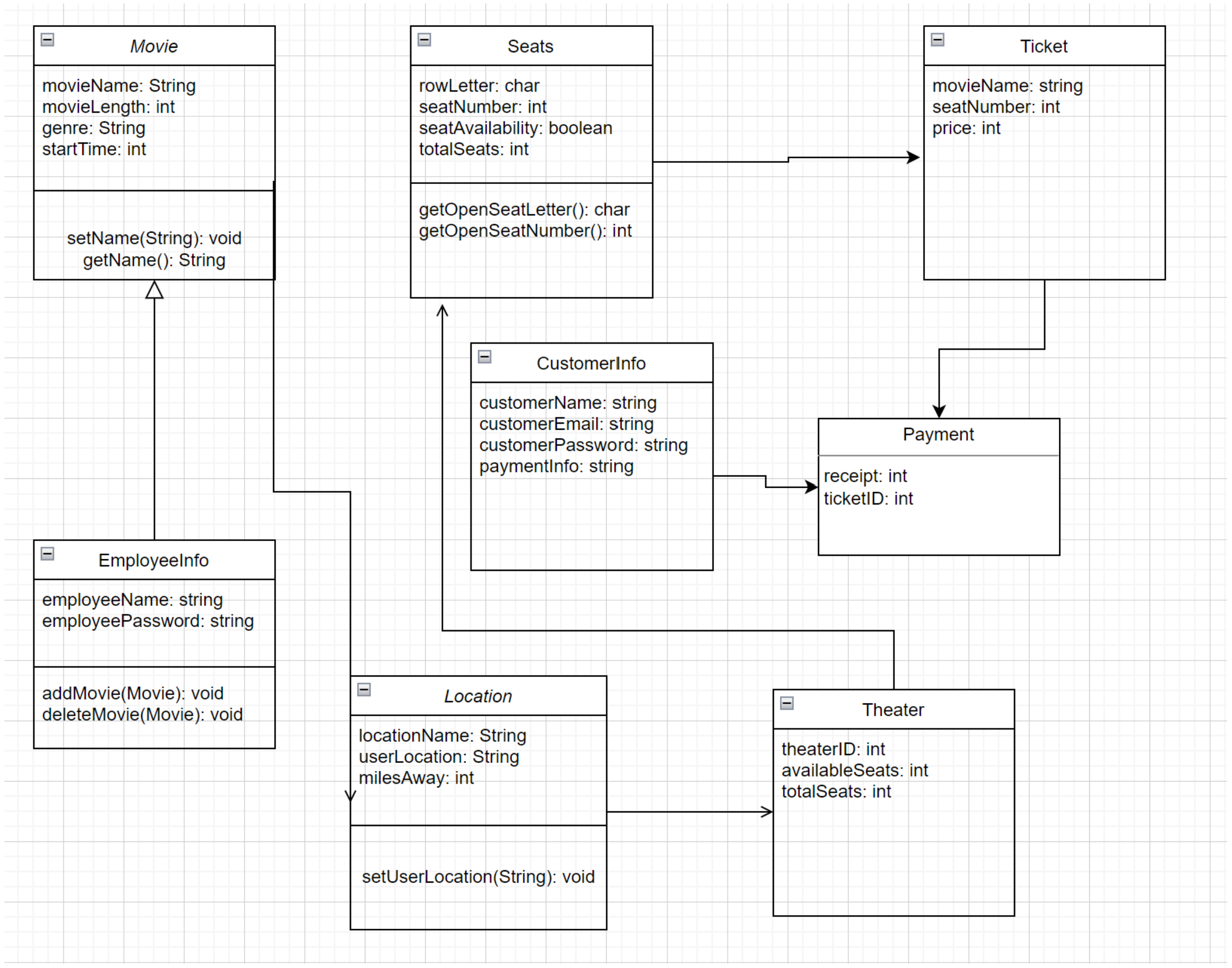
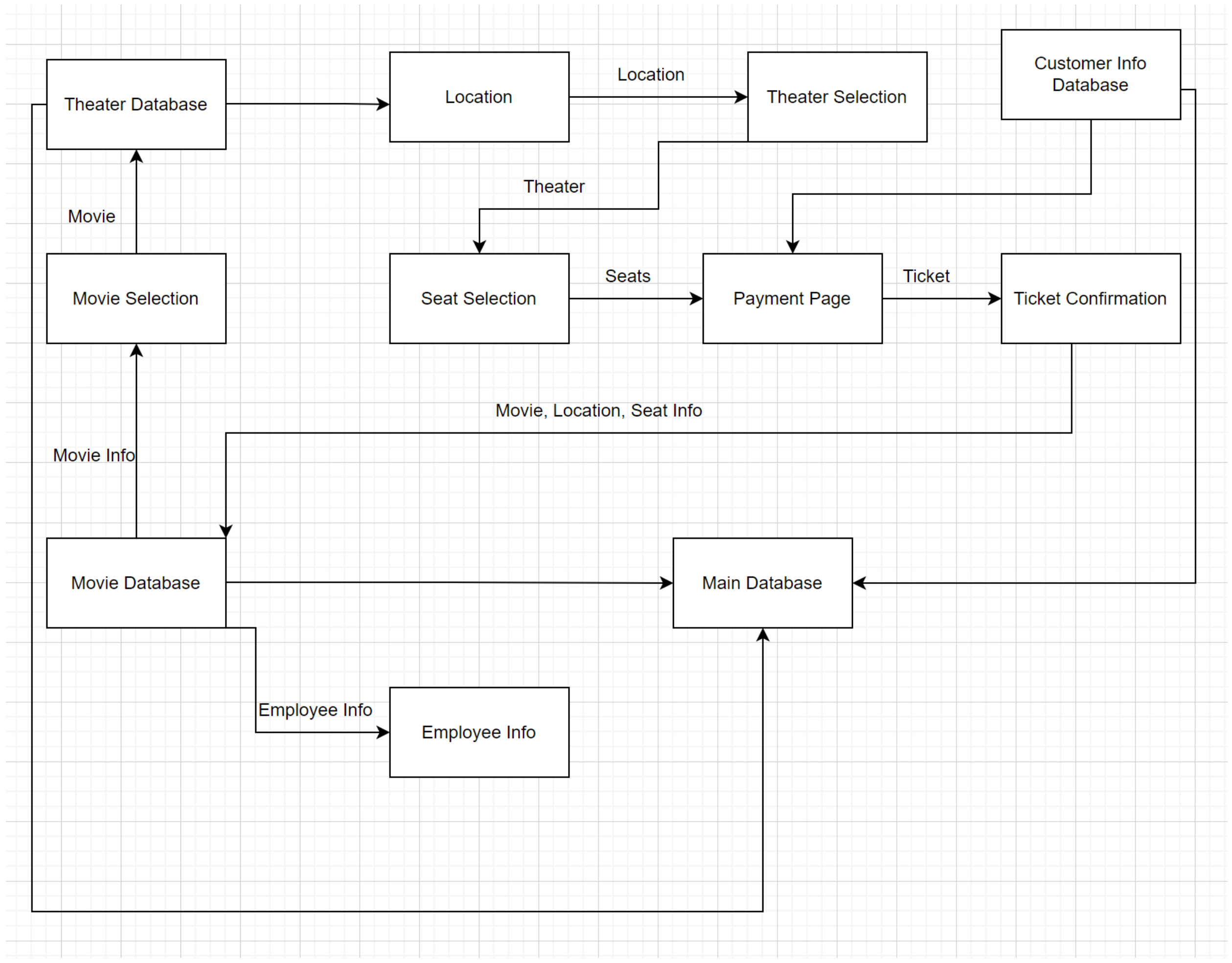
Design Specification Diagram Update:

UML Diagram



https://drive.google.com/file/d/14T24Y2ZmfYA9v__EwmtIC1d0nvIHJule/view?usp=sharing

Architectural Diagram



https://drive.google.com/file/d/1GczHMnxtBrQLPHaztREFe_aUGopSOfZG/view?usp=sharing

Design Specification Description Update:

We updated our diagrams to resemble an Architectural Diagram. In our updated diagram we can see what information will be used in each of the steps in the process of buying a movie ticket. Although compared to the Previous diagram we constructed it may seem as if there is less information on the Architectural Diagram with the new diagram we can better understand and visualize the connections that are being made between the steps which can help developers see what information they can use and how it can be accessed in order to make a working ticket buying program. Both diagrams are necessary when compiling the program as the UML diagram will show what classes and methods are needed along with some of the variables while the Architectural Diagram will show how they relate to each other.

Data Management Strategy Diagrams

SQL Theater Database

theaterID	availableSeats	location
01	13	miramesa
02	47	miramesa
01	32	lamesa
…	…	…

SQL Movie Table Example for Miramesa Location Database

movieName	movieLength	genre	startTime	totalSeats	theaterID
mario	92	adventure	8 pm	120	01
avatar	192	sci-fi	7 pm	100	02
…	…	…	…	…	…

SQL Customer Info Table Database

customerName	customerEmail	customerPassword	paymentInfo
John Smith	jsmith11@gmail.com	*********ve	**** **** **** 7310

Mary Brown	marybrown214@yahoo.com	******89	**** **** **** 5526
…	…	…	…

Data Management Strategy Description

Our data management strategy for the software systems revolves around what we know is persistent and sensitive data and is based on SQL. We have chosen SQL as our database management system as it provides a reliable, efficient, and scalable solution for storing and retrieving data.

Design Decisions:

We have divided our data across three databases: the Theater Database, the Movie Database, and the Customer Info Database. This approach enables us to efficiently manage and retrieve data specific to each entity while ensuring the security and integrity of our data. The Theater Database includes the theater ID, the number of available seats, and the location of the theater. The Movie Database includes the name of the movies, the movie length, the movie genre, the movie start time, the movie end time, and the theater ID. The Customer Info Database includes the customer name, customer email, customer password, and the payment info. By splitting up the data logically into three different databases, we can reduce the likelihood of data duplication and inconsistency. It also allows us to implement security measures specific to each database, ensuring that only authorized users have access to sensitive information. These security measures will help to reduce security risks and ensure the protection of our data.

Possible Alternatives:

Although we have chosen SQL as our database management system, we could have used a non-SQL system such as NoSQL. NoSQL is known for its scalability and flexibility, but it may not be the best choice for our specific needs. SQL offers strong consistency, which is essential for our software systems. A possible alternative for how we handled the organization of data is that we could have chosen to store all data in a single database. However, this could have resulted in a more complex system that would have been challenging to maintain and manage.

Tradeoff Discussion

SQL is the data management strategy that we will use for our project. The SQL approach is the most appropriate because SQL is vertically scalable and table-based, whereas NoSQL is document, key-value, graph, or wide-column stores. The table-based approach is superior at utilizing the Movie Ticketing System. SQL systems are better for multi-row transactions. The Movie Ticketing System is structured and NoSQL is better for unstructured data like documents. Although SQL is more expensive to scale, it better matches the objectives of this project. The data should be first normalized then optimized to meet customer requirements and performance needs. This is done to reduce inefficiency and redundancy. The database should use third normal form to include name, genre, and runtime. The database should include a high level of indexing for faster access time and slower write time, as we will be accessing the data more than writing it.