**Class Diagram-:**

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

The importance of class diagram is:-

1. Shows the static structure of classifiers in a system.
2. Diagram provides basic notation for other structure diagrams prescribed by UML
3. Helpful for developers and other team members too.
4. Business analysts can use class diagram to model systems from business perspectives.
5. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.
6. It describes the attributes and operations of a class and also the constraints imposed in the system.

The Notation used in class diagram are-:

Association

Aggregation

Class Diagram used in this process are shown below-:

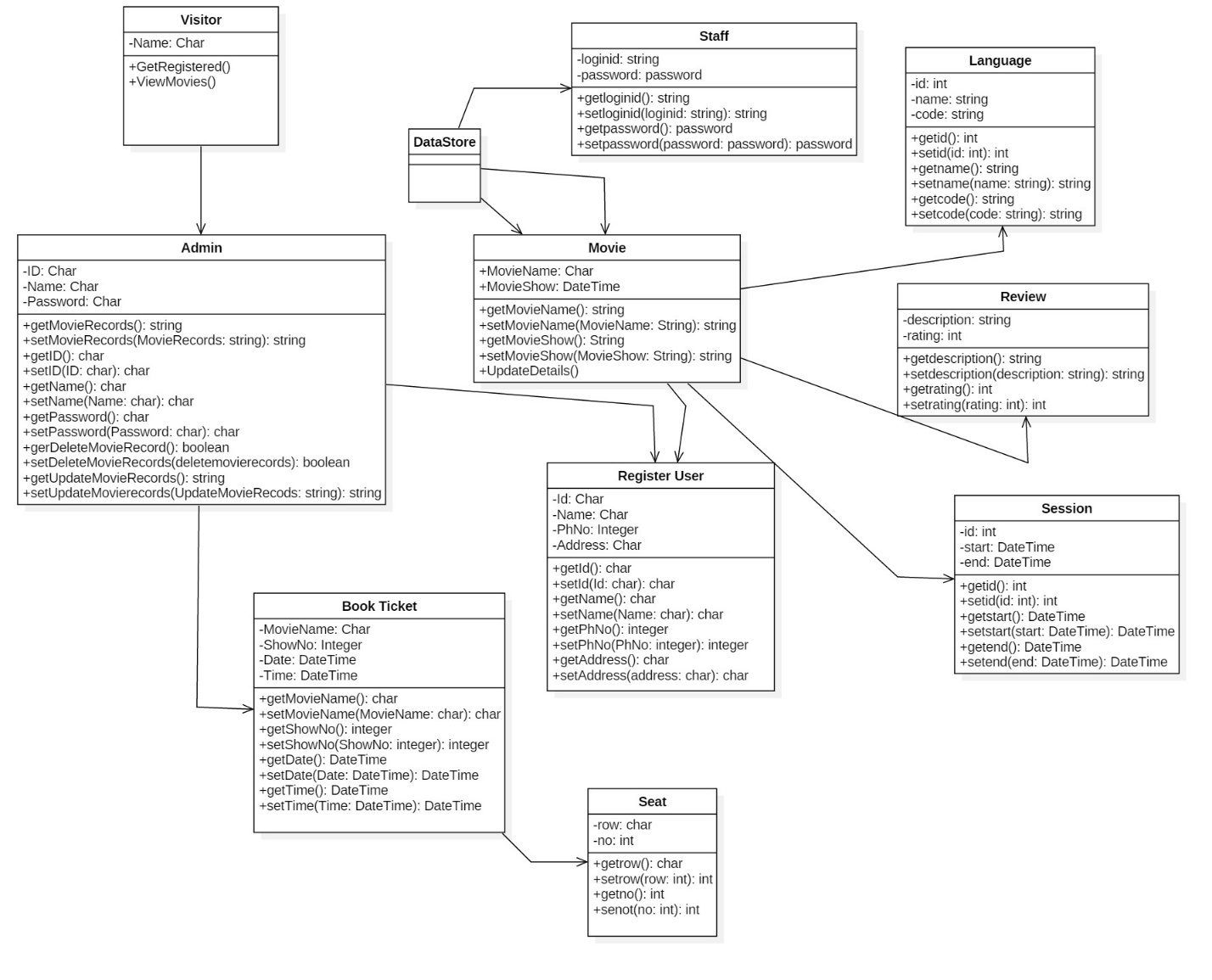


Figure 1 Class Diagram

**Flow Chart-:**

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

Importance of flowchart in the system are-:

* Communication: Flowcharts are better way of communicating the logic of a system to all concerned or involved.
* Effective analysis: With the help of flowchart, problem can be analysed in more effective way therefore reducing cost and wastage of time.
* Proper documentation: Program flowcharts serve as a good program documentation, which is needed for various purposes, making things more efficient.
* Efficient Coding: The flowcharts act as a guide or blueprint during the systems analysis and program development phase.

The Notation used in the flow chart are-:

* An end or beginning
* A step in the flowcharting process
* Directional flow
* Diamond: - decision

Flow Chart for the Project are shown below:

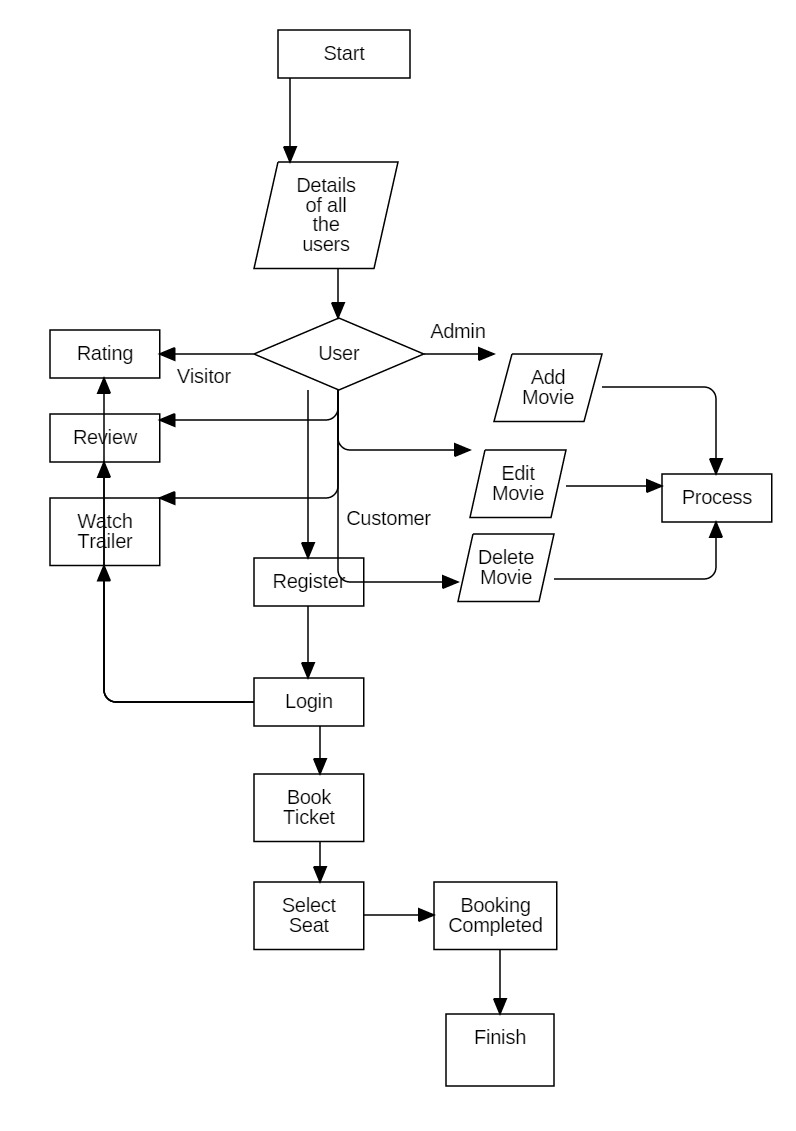


Figure 2 Flow Chart

**Behavioral Design-:**

1. **Activity Diagram-:**

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join.

Importance of Activity Diagram are-:

1. It is used by programmers as it is advanced version of flowchart to depict workflows.
2. It is used by developers to understand the flow of programs on a high level.
3. It also enables them to figure out constraints and conditions that cause particular events.
4. Activity diagram is used to show message flow from one activity to another.
5. It captures the dynamic behaviour of the system.

The Notation used in the diagram are shown below:-

1. Activity

1. Action
2. Control flow It shows the sequence of actions.
3. Object flow
4. Initial node
5. Activity Final Node: -
6. Decision node
7. Merge node

1. Fork node
2. Join node
3. Swim lane and Partition

Activity diagram used in this project are shown below:

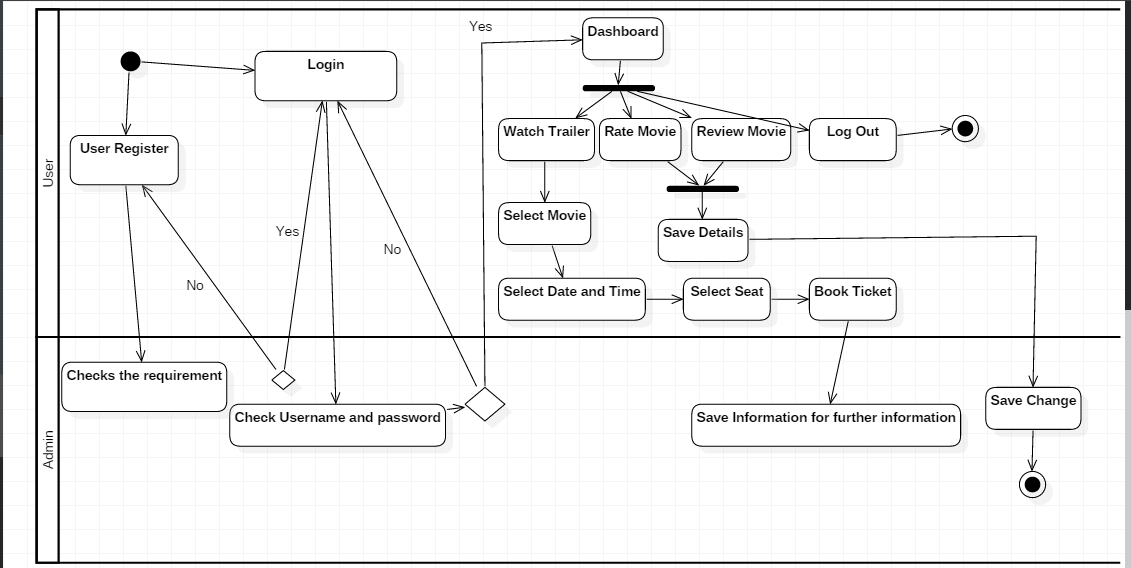


Figure 3 Activity Diagram

**Sequence Diagram-:**

UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

The reason the sequence diagram is so useful is because: -

1. It shows the interactions logic between the objects in the system in the time order that the interactions take place.
2. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.
3. Sequence diagram are more space efficient, simple to modify, allows vertical expansion rather that the horizontal which is the case for sequence diagram.
4. It is an essential component used in processes related to analysis, design and documentation.
5. In the context of UML, it represents object collaboration and is used to define event sequences between objects for a certain outcomes.

Notation used in this diagram are shown below-:

1. Lifeline: - it represents each instance in an interaction.
2. Activate: - it is used to denote participant activation. Once a participant is activated, its lifeline occurs.
3. Objects: - they are model elements that represent instances of a class or of classes.
4. Classes: - In UML, it shows architecture and features of the designed system.
5. Message: - it is an element that defines a specific kind of communication between instances in an interaction.

Sequence Diagram used in this project are shown below:-

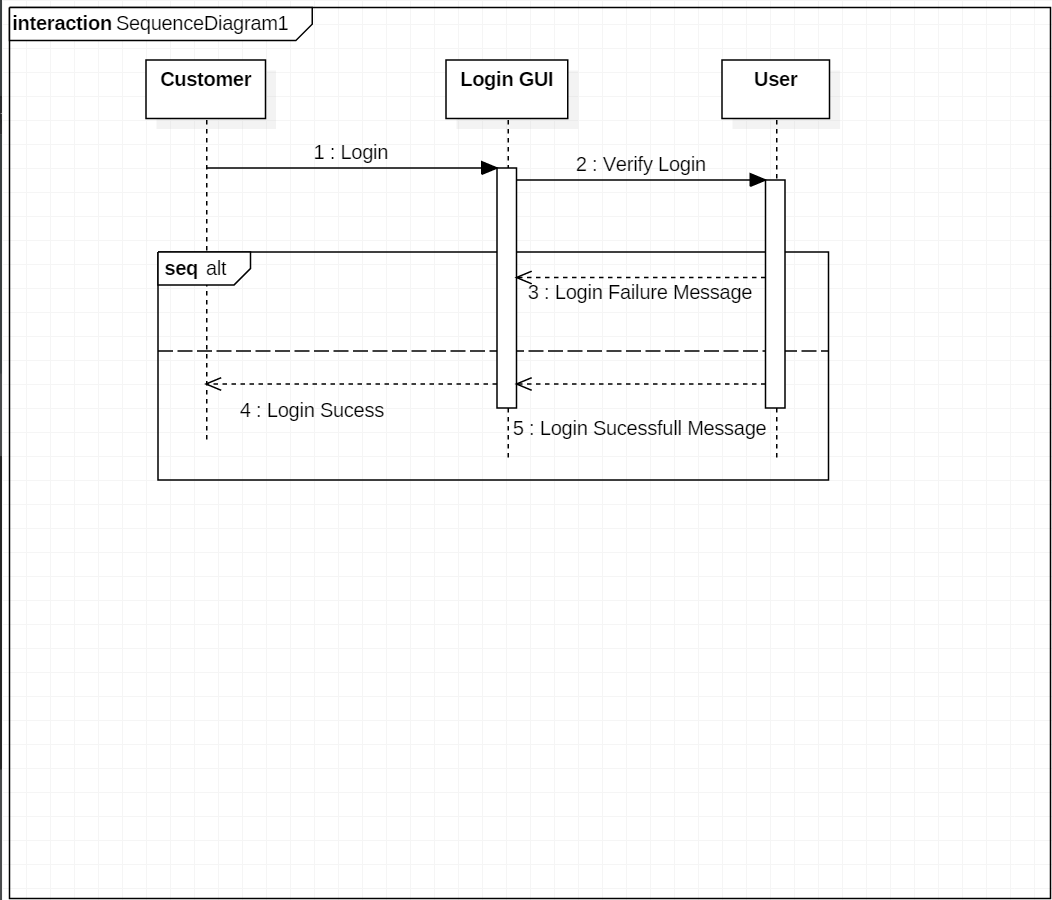


Figure 4 Sequence Diagram for login

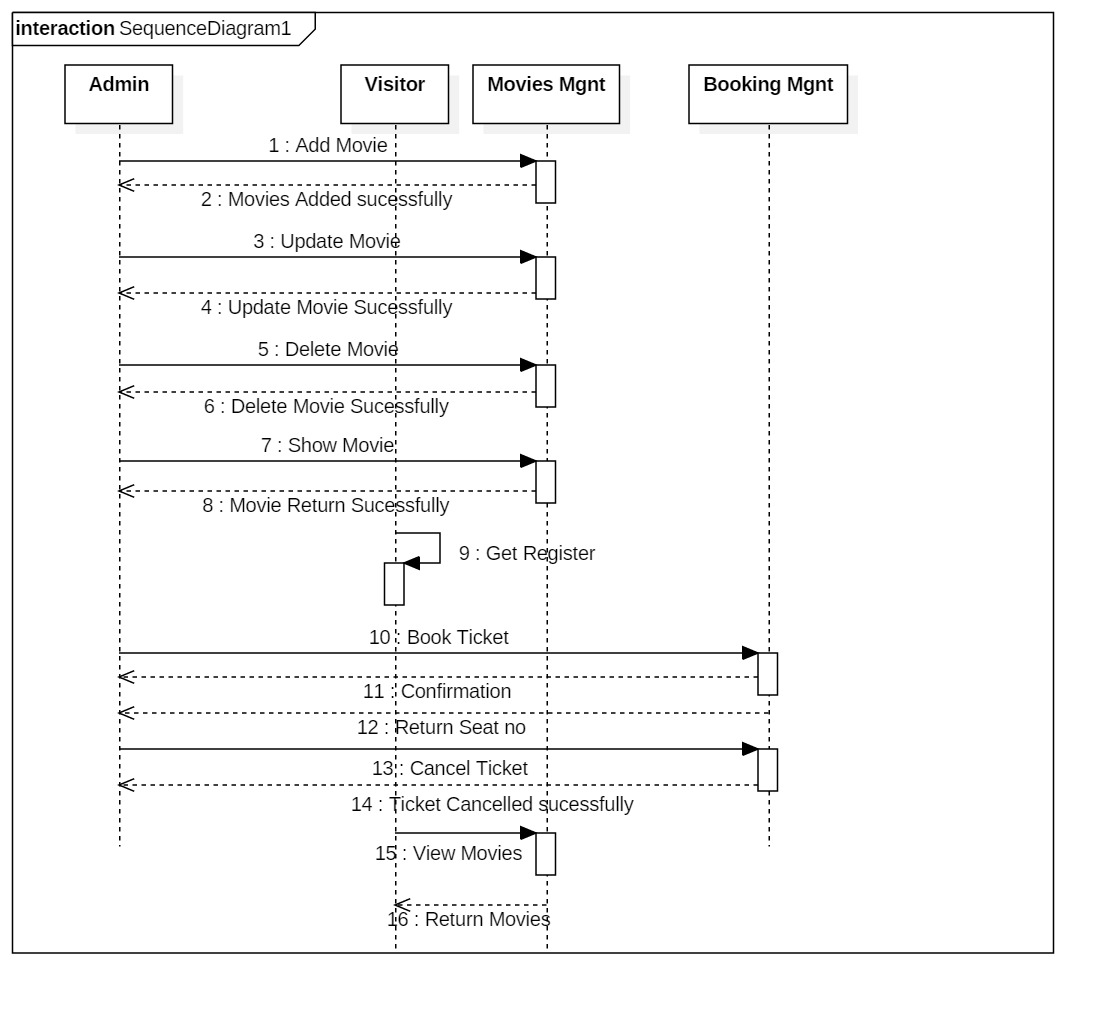


Figure 5 Sequence Diagram for Project

**Data Base**

1. Data Dictionary -:

A data dictionary is a file or a set of files that contains a database's metadata. The data dictionary contains records about other objects in the database, such as data ownership, data relationships to other objects, and other data. The data dictionary is a crucial component of any relational database. Ironically, because of its importance, it is invisible to most database users. Typically, only database administrators interact with the data dictionary.

Data Dictionary for this project are shown below:-

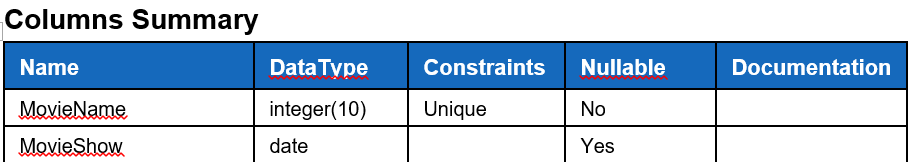


Figure 6 Movie

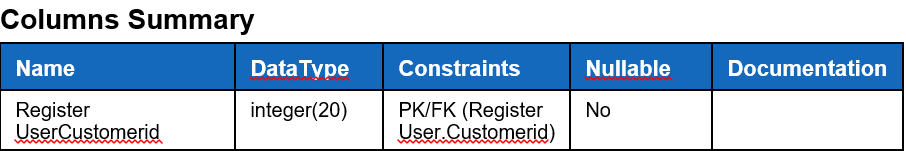


Figure 7

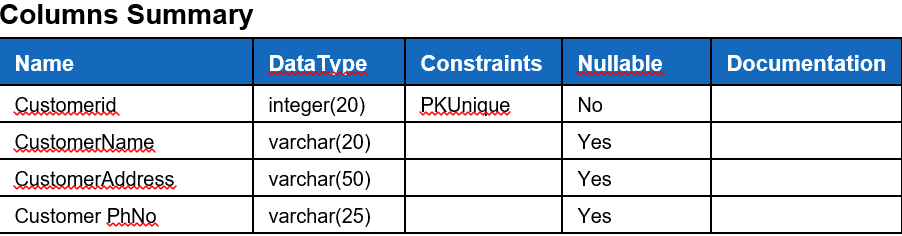


Figure 8

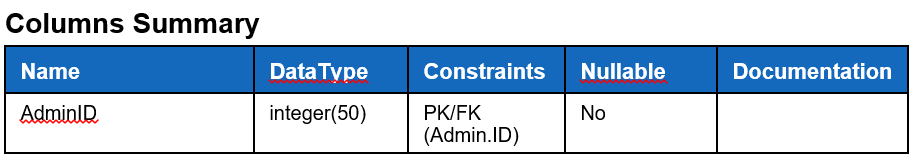


Figure 9

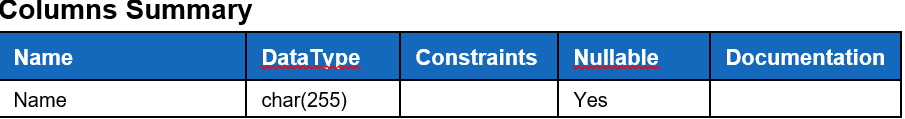


Figure 10

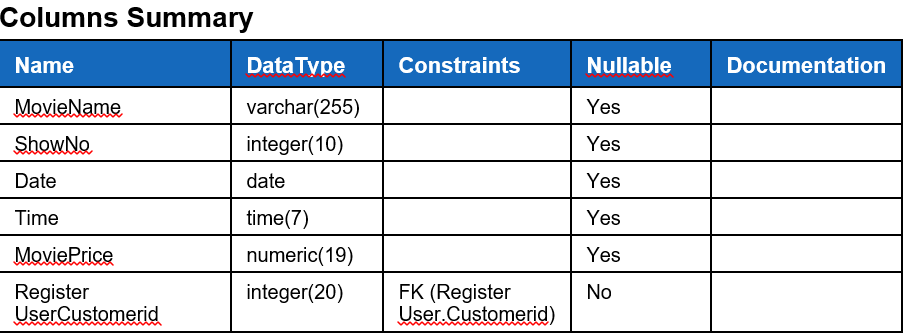


Figure 11

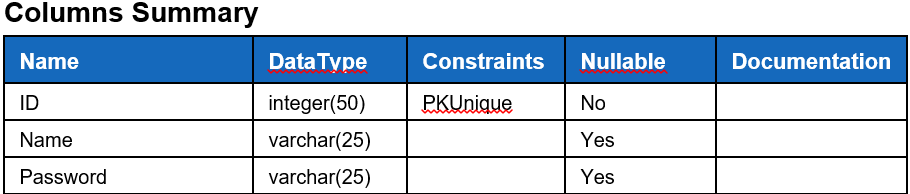


Figure 12

1. ER Diagram-:

Database is absolutely an integral part of software system. To fully utilize ER Diagram in database engineering guarantee you to produce high quality database design to use in database creation, management and maintenance. An ER model also provides a means for communication.

ER Diagram for this project are shown below-:

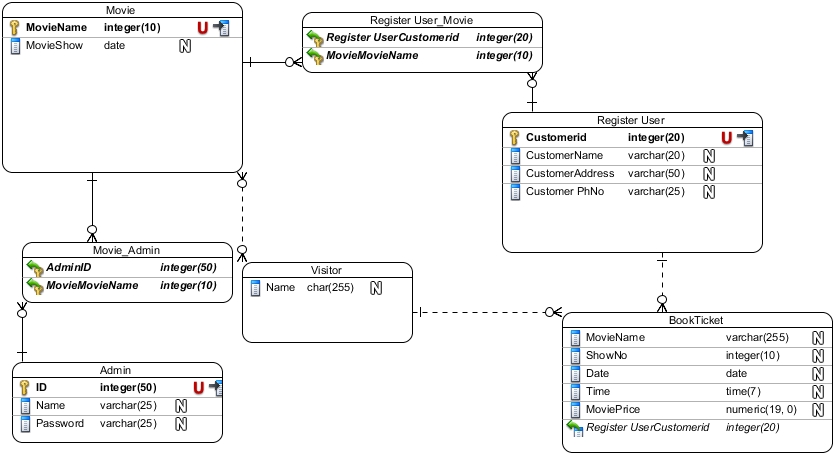


Figure 13 E R Diagram

**Prototyping-:**

**Digital Prototype-:**

A prototype is a digital simulation or demo of a product or service that enables you to test assumptions and virtually explore a product before it’s built. Digital prototyping can save you time and money by ensuring that product development is based on data from the outset, rather than untested assumptions.

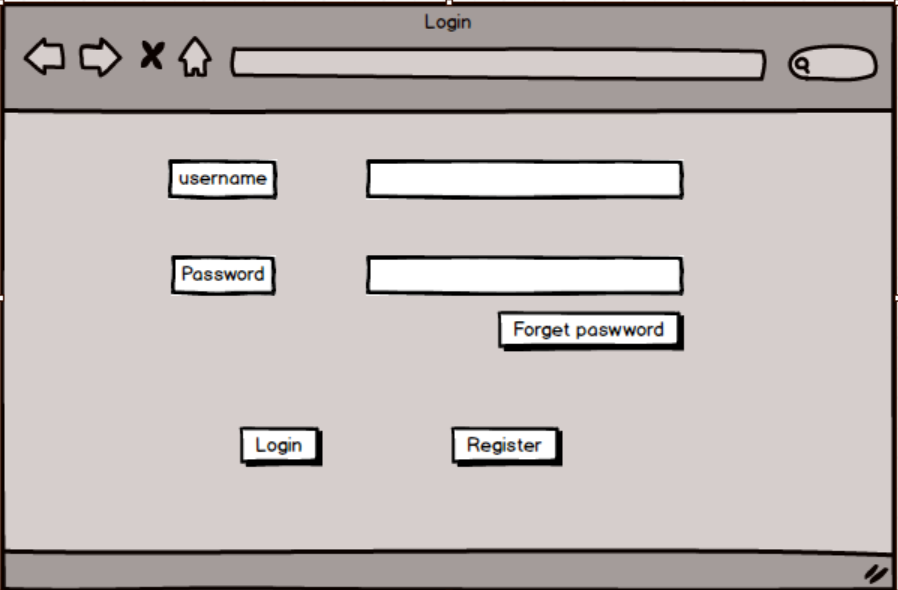
****

Figure 14 Login

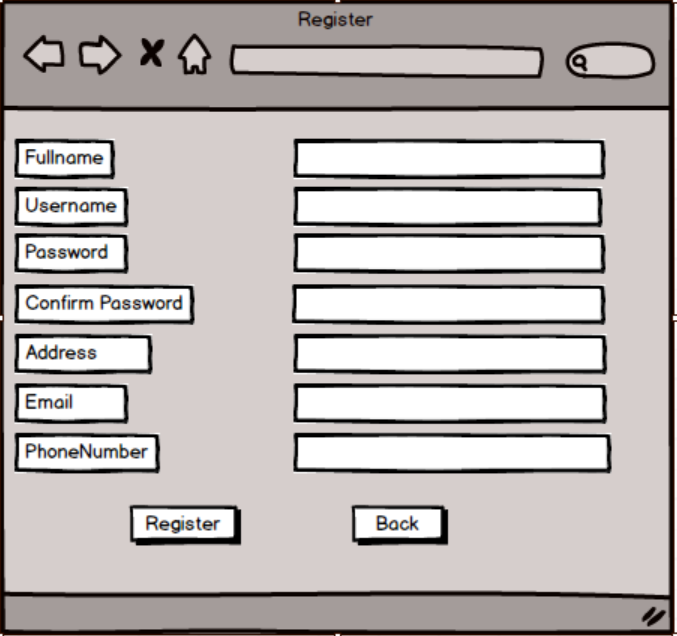


Figure 15 Register

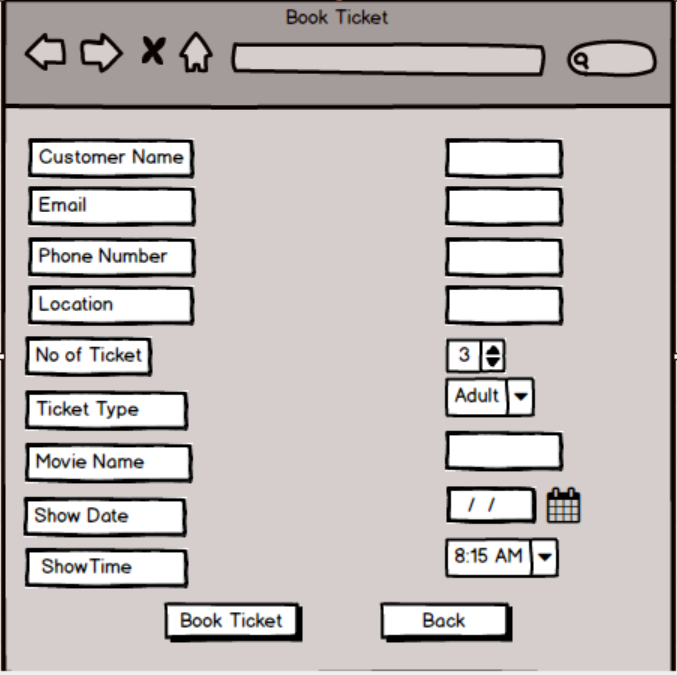


Figure 16 Book Ticket

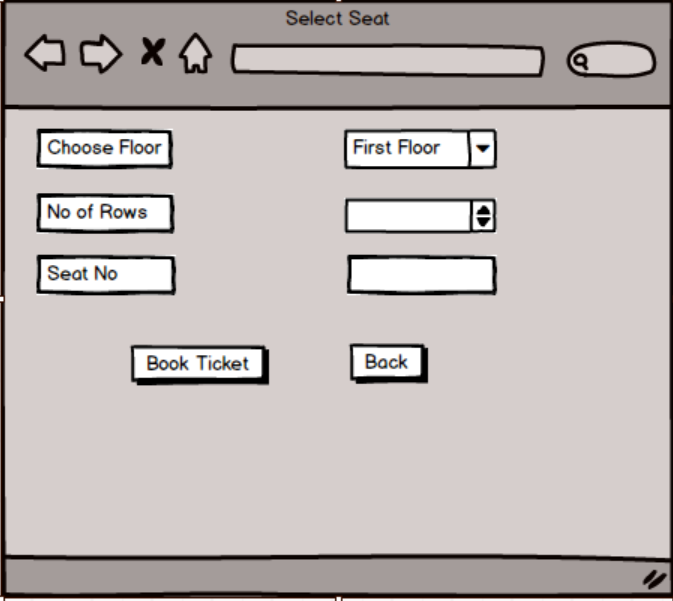


Figure 17 Select Seat

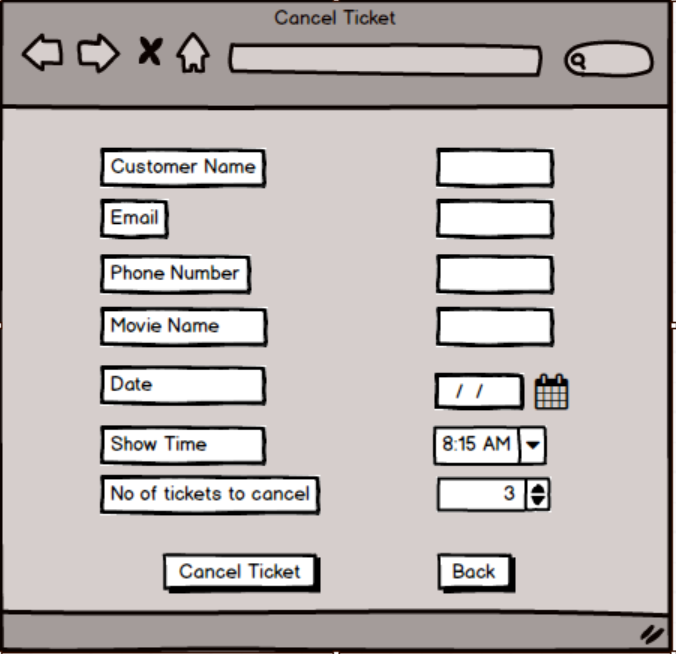


Figure 18 Cancel Seat

**Architecture-:**

In this project I have used 3 –tier structure. A 3-tier architecture is a type of software architecture which is composed of three “tiers” or “layers” of logical computing. They are often used in applications as a specific type of client-server system. 3-tier architectures provide many benefits for production and development environments by modularizing the user interface, business logic, and data storage layers.

3 tier structure used in this project are shown below:-

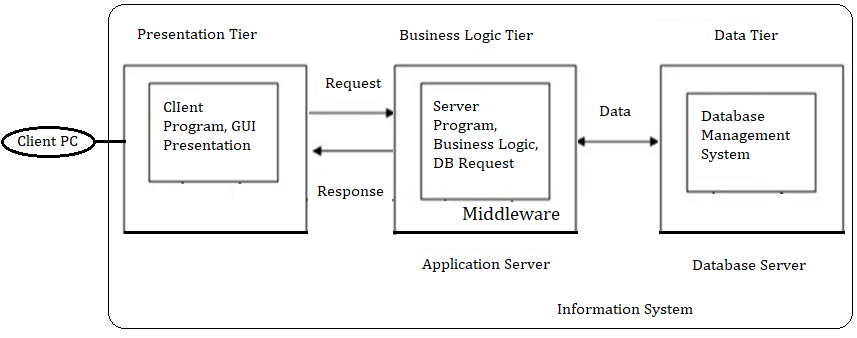


Figure 19 3-tier structure

1. **Presentation Tier-:**

The presentation tier is the front end layer in the 3-tier system and consists of the user interface. This user interface is often a graphical one accessible through a web browser or web-based application and which displays content and information useful to an end user. This tier is often built on web technologies such as HTML5, JavaScript, CSS, or through other popular web development frameworks, and communicates with others layers through API calls.

1. **Application Tier-**The application tier contains the functional business logic which drives an application’s core capabilities. It’s often written in Java, .NET, C#, Python, C++, etc.
2. **Data Tier-**The data tier comprises of the database/data storage system and data access layer. Examples of such systems are MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MongoDB, etc. Data is accessed by the application layer via API calls.

Importance of 3 tier architecture in this project are shown below:-

1. It gives the ability to update the technology stack of one tier, without impacting other areas of the application.
2. It allows for different development teams to each work on their own areas of expertise. Today’s developers are more likely to have deep competency in one area, like coding the front end of an application, instead of working on the full stack.
3. It helps to able to scale the application up and out. A separate back-end tier, for example, allows you to deploy to a variety of databases instead of being locked into one particular technology. It also allows you to scale up by adding multiple web servers.
4. It adds reliability and more independence of the underlying servers or services.
5. It provides an ease of maintenance of the code base, managing presentation code and business logic separately, so that a change to business logic, for example, does not impact the presentation layer.