# Design

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system.

Systems design implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach, but either way the process is systematic wherein it takes into account all related variables of the system that needs to be created—from the architecture, to the required hardware and software, right down to the data and how it travels and transforms throughout its travel through the system. Systems design then overlaps with systems analysis, systems engineering and systems architecture.

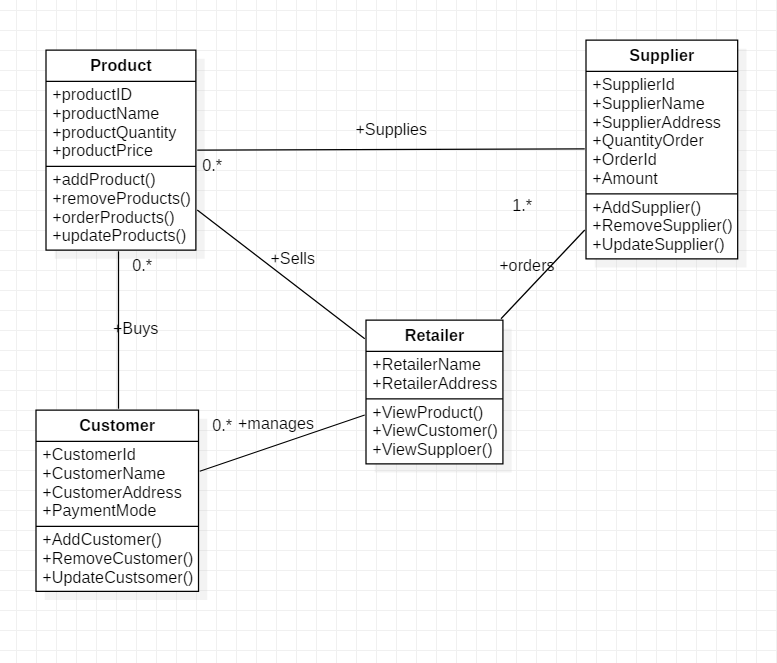
## Structural Design

Structural patterns ease the design of a program by identifying a way to realize relationships between entities. The difference between class patterns and object patterns is that class patterns describe how inheritance can be used to provide more useful program interfaces. Object patterns, on the other hand, describe how objects can be composed into larger structures using object composition, or the inclusion of objects within other objects.

### Parts of Structural Design are

#### Class Diagram

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity.

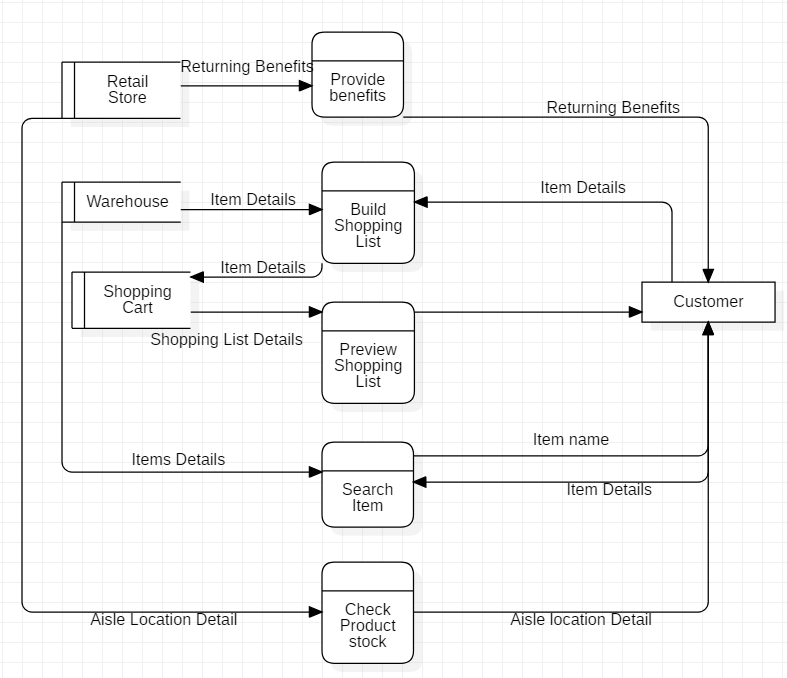


**IMAGE CLASS DIAGRAM**

**Explain Image**

**Data Flow Diagram**

A data flow diagram (or DFD) is a graphical representation of the flow of data through an information system. It shows how information is input to and output from the system, the sources and destinations of that information, and where that information is stored.



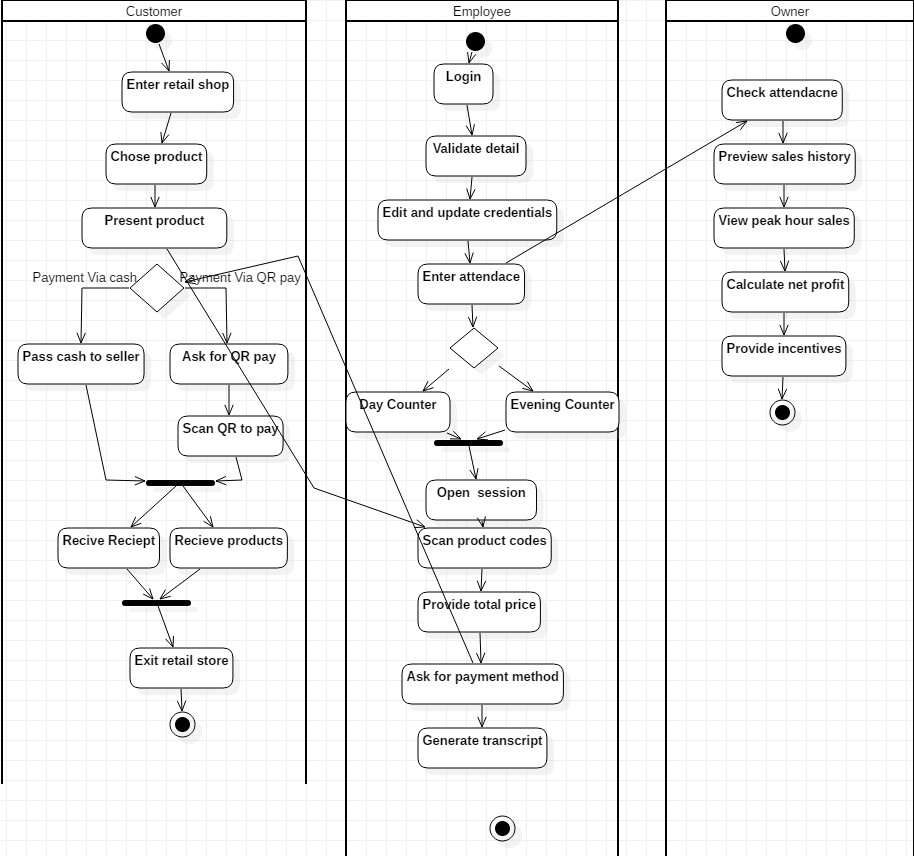
**EXPLAIN IMAGE**

# Behaviour Design

In software engineering, behavioural design patterns are design patterns that identify common communication patterns among objects and realize these patterns. By doing so, these patterns increase flexibility in carrying out this communication.

## 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 as fork,join,etc.



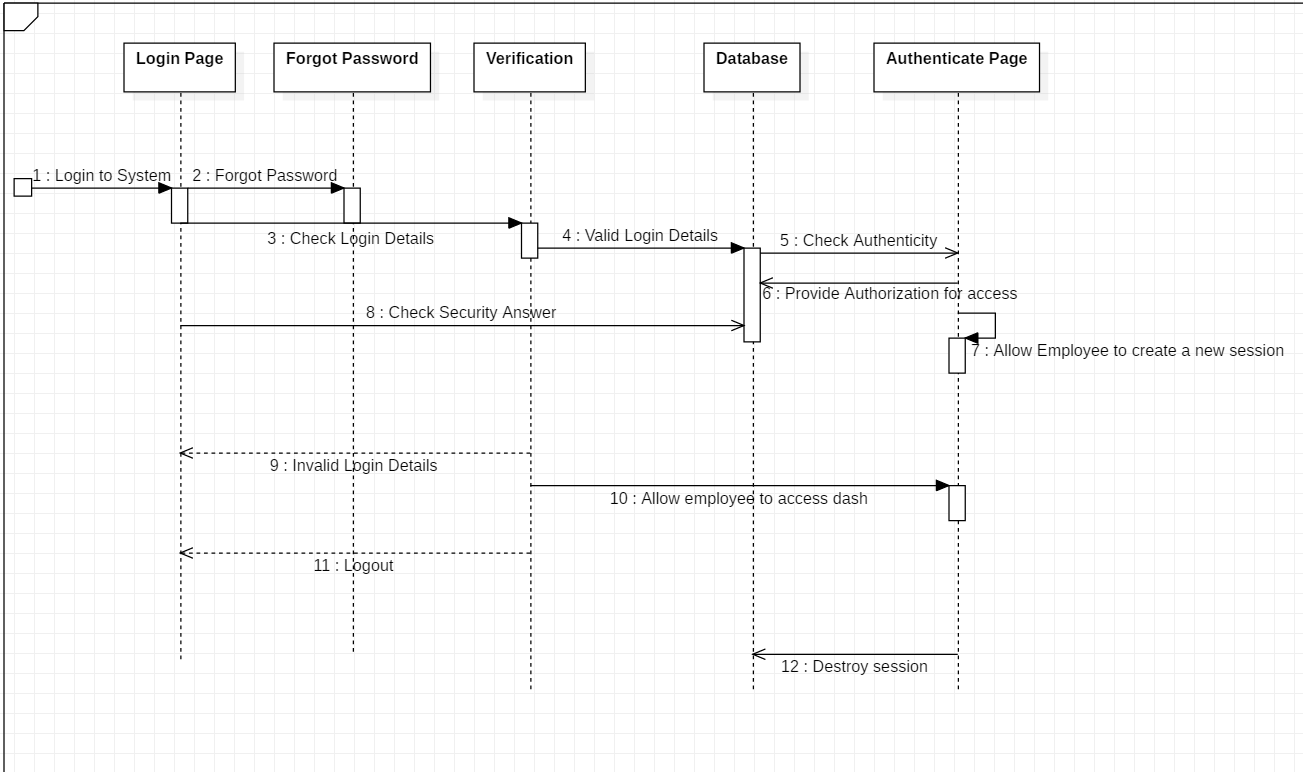
**IMAGE ACTIVITY**

**EXPALIN IMAGE**

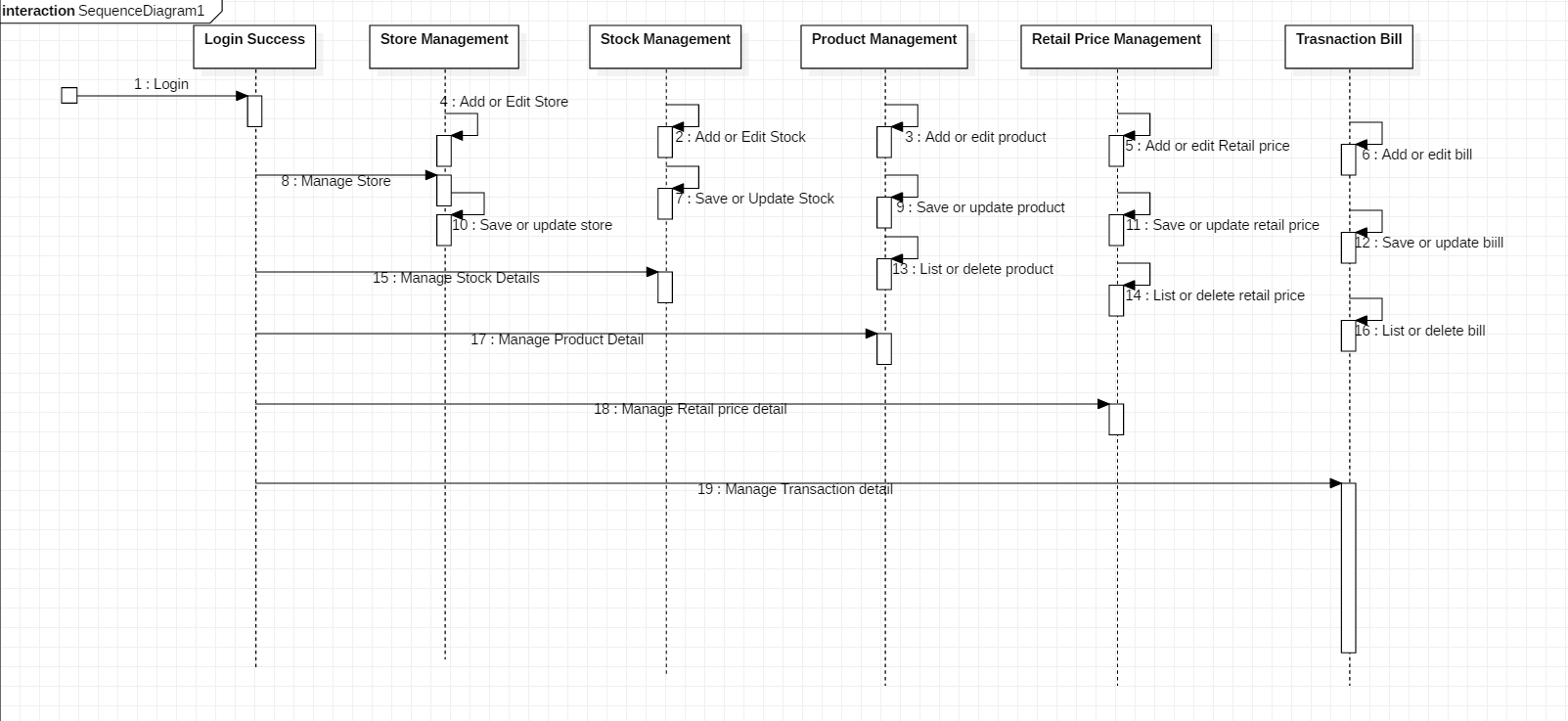
## Sequence diagram

Object interactions usually begin at the top of a diagram and end at the bottom. In a sequence diagram, object interaction occurs through messages on the vertical and horizontal dimensions and are designated by horizontal arrows and message names. The initial sequence diagram message begins at the top and is located on the diagram's left side. Subsequent messages are added just below previous messages. Sequence diagram messages may be subdivided by type, based on functionality.

**IMAGE Sequence**



**EXPLAIN IMAGE**



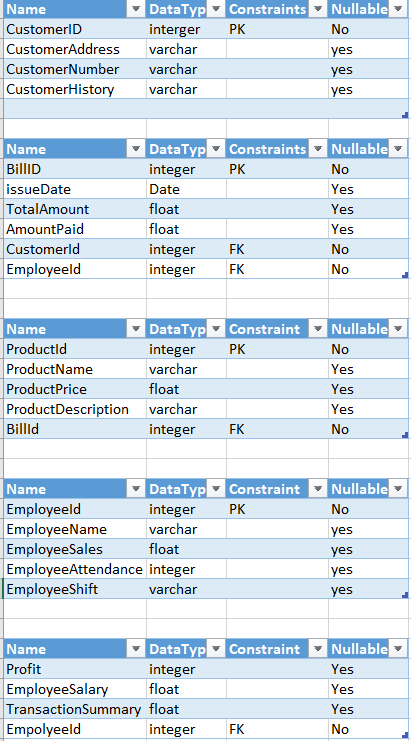
# Database design

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.

In a relational database, the metadata in the data dictionary includes the following:

* Names of all tables in the database and their owners
* Names of all indexes and the columns to which the tables in those indexes relate
* Constraints defined on tables, including primary keys, foreign-key relationships to other tables, and not-null constraints

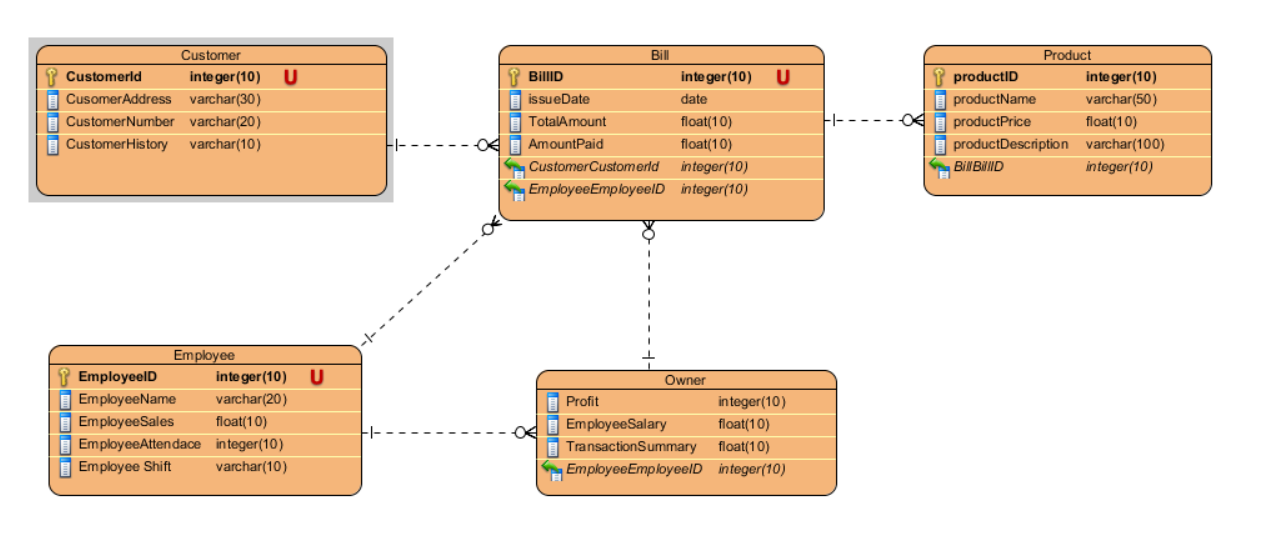


## ER Diagram

An entity-relationship diagram (ERD) is a data modelling technique that graphically illustrates an information system’s entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

The elements of an ERD are:

* Entities
* Relationships
* Attributes



# Architectural Design

Architecture serves as a blueprint for a system. It provides an abstraction to manage the system complexity and establish a communication and coordination mechanism among components.

It defines a structured solution to meet all the technical and operational requirements, while optimizing the common quality attributes like performance and security.Further, it involves a set of significant decisions about the organization related to software development and each of these decisions can have a considerable impact on quality, maintainability, performance, and the overall success of the final product. These decisions comprise of −

Selection of structural elements and their interfaces by which the system is composed.

* Behavior as specified in collaborations among those elements.
* Composition of these structural and behavioral elements into large subsystem.
* Architectural decisions align with business objectives.
* Architectural styles guide the organization.

## UI Protyping

User interface (UI) prototyping is an iterative development technique in which users are actively involved in the mocking-up of the UI for a system. UI prototypes have several purposes:

* As an analysis artefact that enables you to explore the problem space with your stakeholders.
* As a design artefact that enables you to explore the solution space of your system.
* A basis from which to explore the usability of your system.
* A vehicle for you to communicate the possible UI design(s) of your system.
* A potential foundation from which to continue developing the system (if you intend to throw the prototype away and start over from scratch then you don’t need to invest the time writing quality code for your prototype)

