**MODULE: 1**

**SE- Overview of IT Industry**

1. **What is software? What is software engineering?**

**Ans.** **Software** is a set of instructions, data or programs used to operate computers and execute specific tasks. It is the **opposite of hardware**, which describes the physical aspects of a computer, and **Software engineering** is a **branch of computer science** that involves the design, development, testing, and maintenance of software applications.

1. **Explain types of software.**

**Ans.** Among the various categories of software, the most common types include the following:

**Application software**: The most common type of software, application software is a computer software package that performs a specific function for a user, or in some cases, for another application. Examples of modern applications include office suites, graphics software, databases and database management programs, web browsers, word processors, software development tools, image editors and communication platforms.

**System software**: These software programs are designed to run a computer's application programs and hardware. In addition, it controls the operations of the computer hardware and provides an environment or platform for all the other types of software to work in. The OS is the best example of system software; it manages all the other computer programs. Other examples of system software include the firmware, computer language translators and system utilities.

**Driver software**: Also known as device drivers, this software is often considered a type of system software. Device drivers control the devices and peripherals connected to a computer, enabling them to perform their specific tasks. Every device that is connected to a computer needs at least one device driver to function. Examples include software that comes with any nonstandard hardware, including special game controllers, as well as the software that enables standard hardware, such as USB storage devices, keyboards, headphones and printers.

**Middleware**: The term middleware describes software that mediates between application and system software or between two different kinds of application software. For example, middleware enables Microsoft Windows to talk to Excel and Word. It is also used to send a remote work request from an application in a computer that has one kind of OS, to an application in a computer with a different OS. It also enables newer applications to work with legacy ones.

**Programming software**: Computer programmers use programming software to write code. Programming software and programming tools enable developers to develop, write, test and debug other software programs. Examples of programming software include assemblers, compilers, debuggers and interpreters.

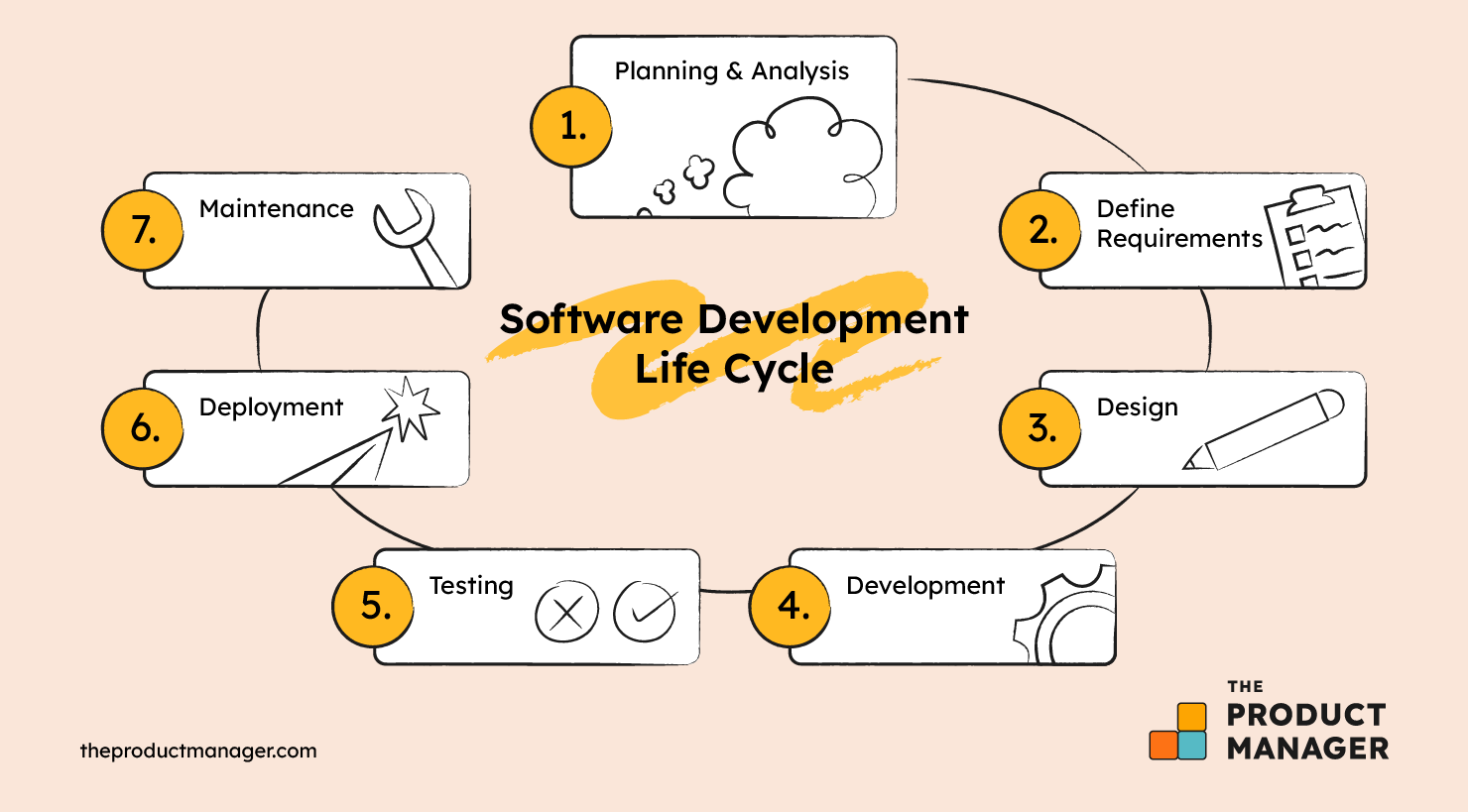
1. **What is SDLC? Explain each phase of SDLC.**

**Ans.** **SDLC** stands for **Software Development Life Cycle.**

A systematic approach that generates a structure for the developer to **design, create and deliver high-quality software** based on **customer requirements and needs**.

The primary goal of the SDLC process is to produce **cost-efficient and high-quality products**. The process comprises a detailed plan that describes how to develop, maintain, and replace the software.

Popular SDLC models include the **Waterfall model, Spiral model, Agile model, Iterative model, Spiral model and** **Big Bang model** .

The **7 Phases** of the **Software Development Life Cycle :**

1. **Planning & Analysis**

The first phase of the SDLC is the project planning stage where you are gathering business requirements from your client or stakeholders. To properly decide what to make, what not to make, and what to make first, you can use a feature prioritization framework that takes into account the value of the software/update, the cost, the time it takes to build, and other factors.

1. **Define Requirements**

This phase is critical for converting the information gathered during the planning and analysis phase into clear requirements for the development team. This process guides the development of several important documents: a software requirement specification (SRS) or product specification, a Use Case document, and a Requirement Traceability Matrix document.

1. **Design**

The design phase is where you put pen to paper—so to speak. This is also where you can flowchart how the software responds to user actions.

In most cases, the design phase will include the development of a prototype model. Creating a pre-production version of the product can give the team the opportunity to visualize what the product will look like and make changes without having to go through the hassle of rewriting code.

1. **Development**

The actual development phase is where the development team members divide the project into software modules and turn the software requirement into code that makes the product.

1. **Testing**

Before getting the software product out the door to the production environment, it’s important to have your quality assurance team perform validation testing to make sure it is functioning properly and does what it’s meant to do.

There are many types of testing. like, Performance testing, Functional testing, Security testing, etc.

1. **Deployment**

During the deployment phase, your final product is delivered to your intended user.

1. **Maintenance**

In the maintenance stage, users may find bugs and errors that were missed in the earlier testing phase. These bugs need to be fixed for better user experience and retention. In some cases, these can lead to going back to the first step of the software development life cycle.

The SDLC phases can also restart for any new features you may want to add in your next release/update.

1. **What is DFD? Create a DFD diagram on Flipkart.**

**Ans.** DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD.

It provides an overview of

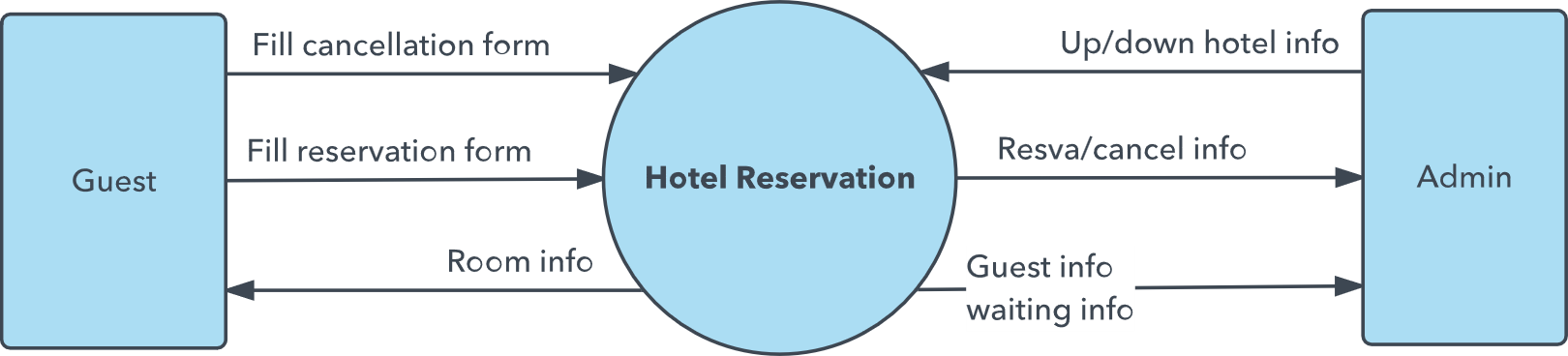
1. What data is system processes.
2. What transformation are performed.
3. What data are stored.
4. What results are produced, etc.

**DFD Symbols:**

1. **Processes:** Represented by circles, ovals, or rectangles, processes are used to transform incoming data flow into outgoing data flow.
2. **Data Flow:** Represented by arrows, these show the direction and route of data as it moves through the system. It signifies what kind of information will be input and output from the system.
3. **Data Stores:** Often represented by two horizontal lines, these indicate data repositories like databases or other storage mechanisms where data rests.
4. **Entities:** Represented by rectangles or squares, entities can be external actors or system units interacting with the system. They can be sources or destinations of data.

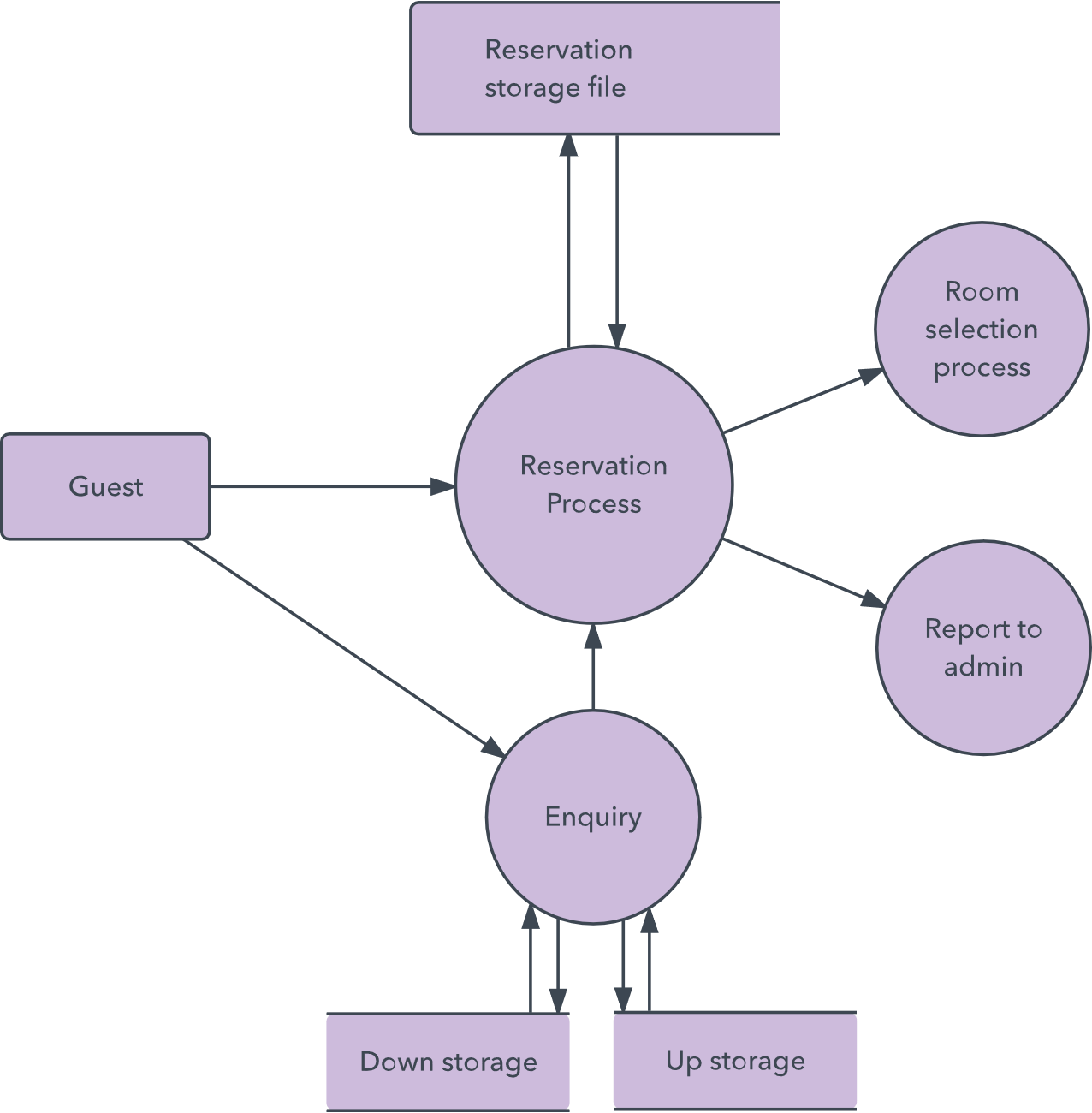
**DFD Levels and Layers:**

DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish.



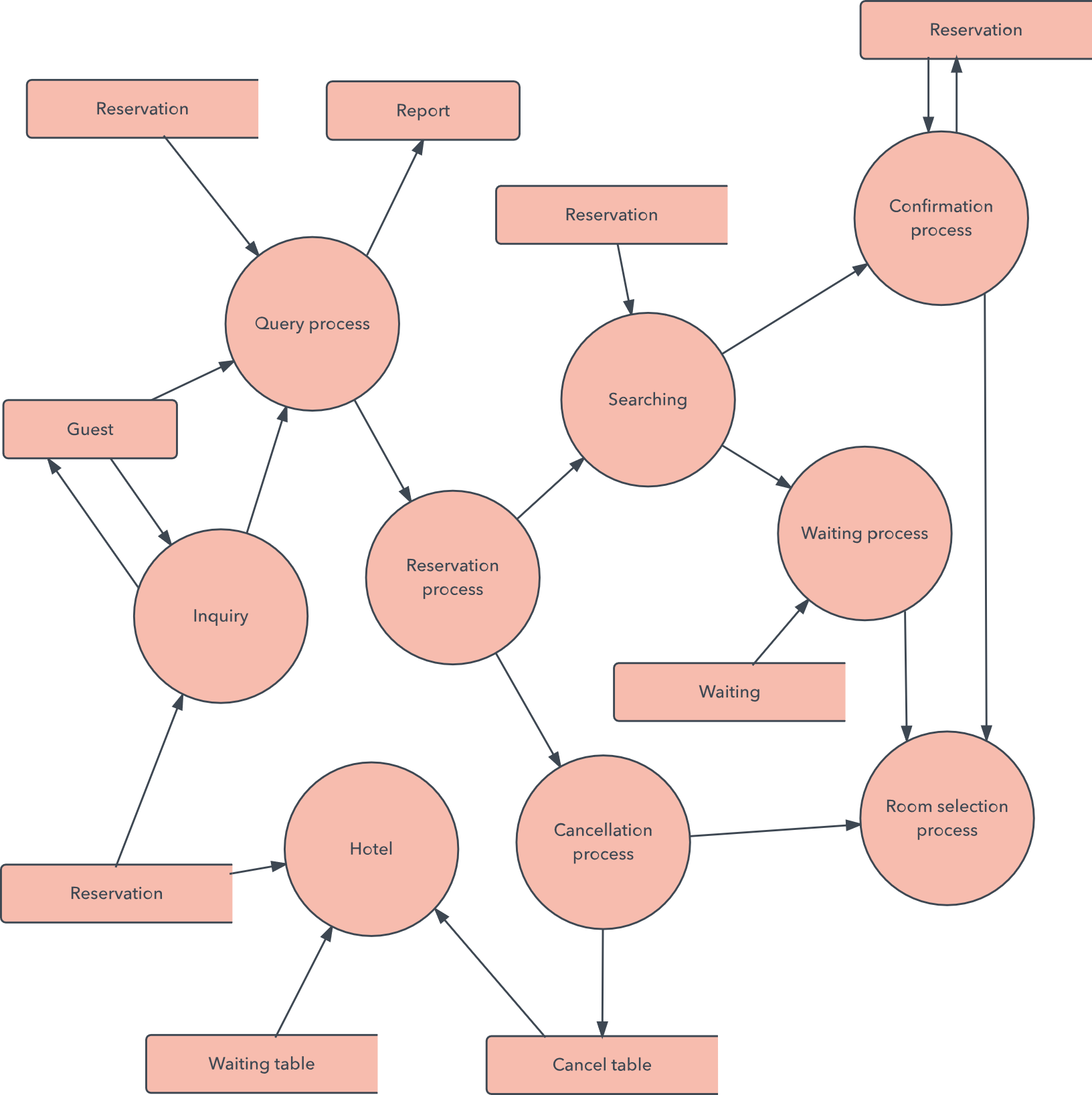
**Ex. DFD Level 0**

**DFD Level 0** is also called a **Context Diagram**. It’s a basic overview of the whole system or process being analyzed or modeled. It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.



**Ex. DFD Level 1**

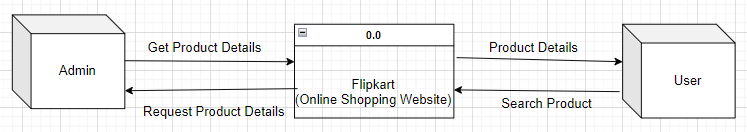
**DFD Level 1** provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its subprocesses.



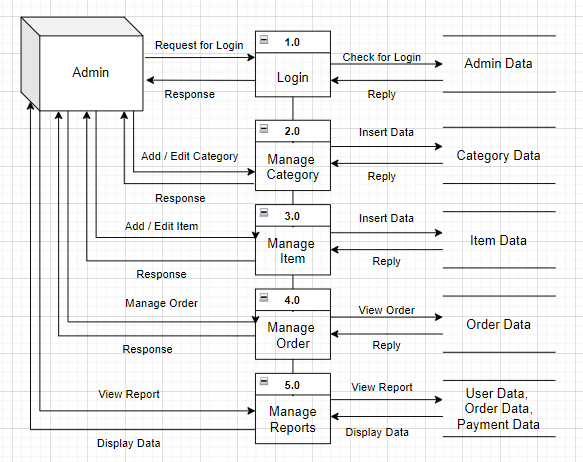
**Ex. DFD Level 2**

**DFD Level 2** then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the system’s functioning. Progression to Levels 3, 4 and beyond is possible, but create complexity that makes it difficult to communicate, compare or model effectively.

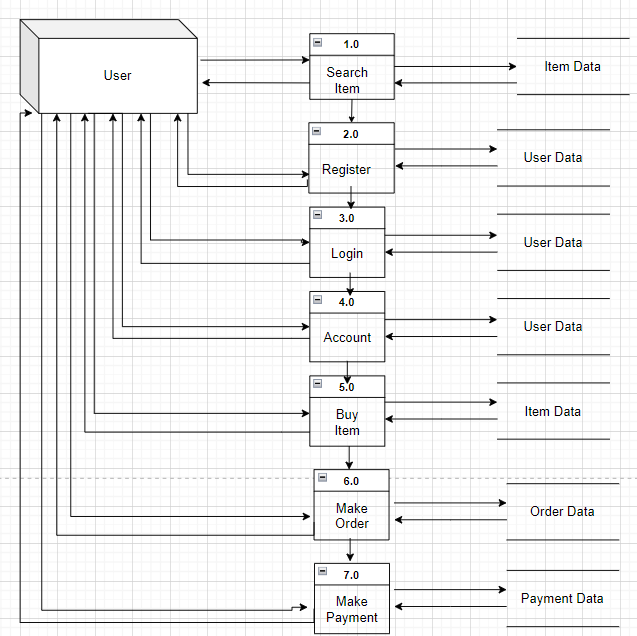
**DFD diagram on Flipkart:**

****

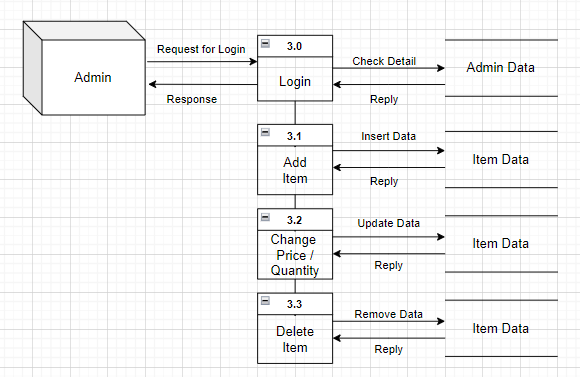
**DFD Level 0**

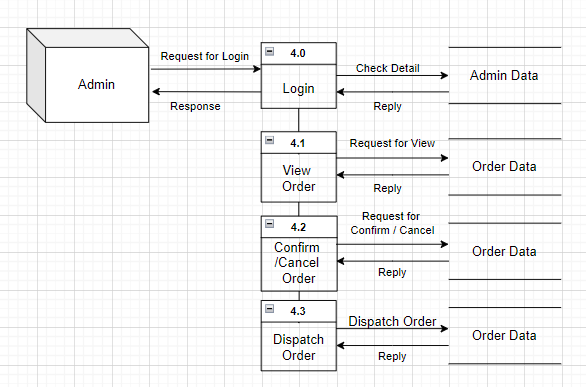
****

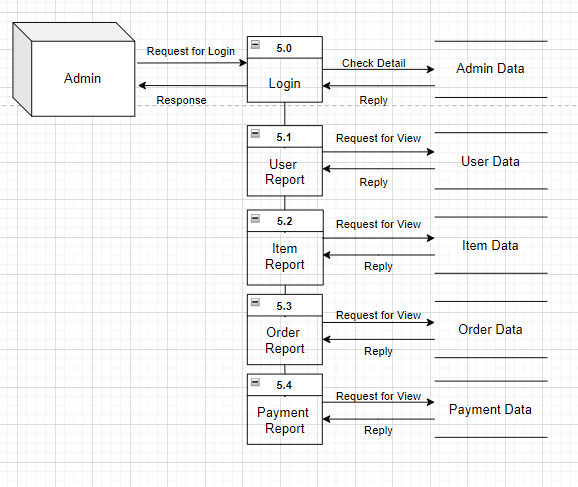
**DFD Level 1 (ADMIN)**

****

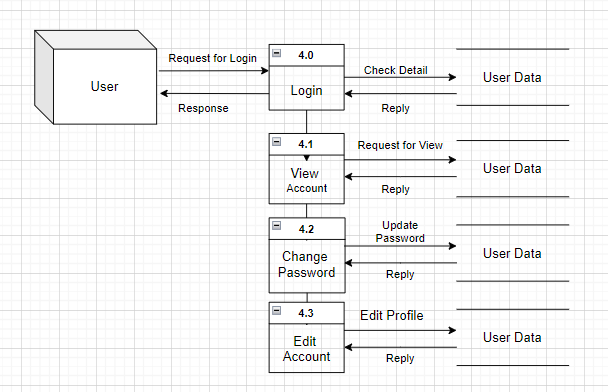
**DFD Level 1 (USER)**

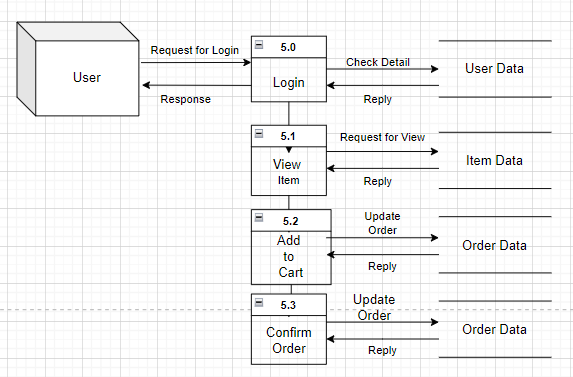
****

****

****

**DFD Level 2 (ADMIN)**

****

****

**DFD Level 2 (USER)**

1. **What is Flow chart? Create a** **flowchart to make addition of two numbers.**

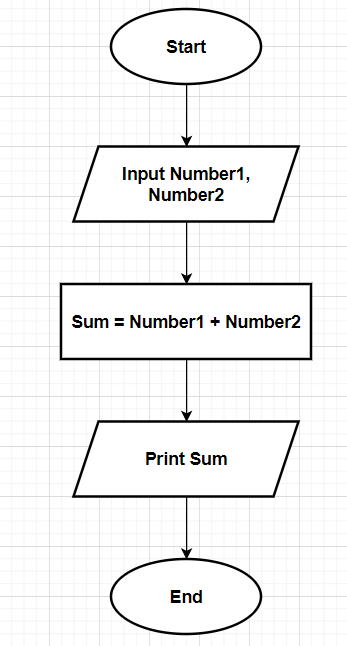
**Ans.** A **Flowchart** is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams.

Flowcharts, sometimes spelled as flow charts, use rectangles, ovals, diamonds and potentially numerous other shapes to define the type of step, along with connecting arrows to define flow and sequence.

**Flowchart symbols:**

Here are some of the common flowchart symbols.

|  |  |  |  |
| --- | --- | --- | --- |
| Terminal/Terminator |  | Data, or Input/Output |  |
| Process |  | Flow Arrow |  |
| Decision |  |  |  |

****

**Flowchart of Addition of Two Numbers:**

1. **What is Use case Diagram? Create a use-case on bill payment on Paytm.**

**Ans.** In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system.

**Use case diagram components:**

To answer the question, "What is a use case diagram?" you need to first understand its building blocks. Common components include:

**Actors:** The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.

**System:** A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.

**Goals:** The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

**Use Case Diagram for Bill payment on Paytm**

**Actors:**

1. User:The person who uses the Paytm application for bill payment.
2. Paytm System: The Paytm application/system itself.

**Use Cases:**

1. Login: The user logs into the Paytm application.
2. Select Bill Payment: The user selects the option for bill payment from the Paytm menu.
3. Choose Biller: The user selects the biller to whom they want to make the payment (e.g., electricity, water, gas).
4. Enter Bill Details: The user enters the details of the bill, such as bill amount, account number, etc.
5. Select Payment Method: The user selects the payment method (e.g., Paytm wallet, debit/credit card, net banking).
6. Authenticate Payment: The user authenticates the payment using the chosen method (e.g., entering OTP, fingerprint).
7. Process Payment: The Paytm system processes the payment request.
8. Update Payment Status: The Paytm system updates the payment status (success/failure) and notifies the user.
9. View Payment History: The user can view their payment history within the Paytm application.

**Relationships:**

**Includes:** The "Process Payment" use case includes "Authenticate Payment" and "Update Payment Status" use cases.

**Extends:** The "Select Payment Method" use case can extend to include additional steps or alternative payment methods if needed.

