MODULE:-1 (SDLC)

Q-1 What is Software?What is software engineering?

➔ Software is a set of instructions, data, or programs used to operate computers and execute specific tasks. It is the intangible components of a computer system that tells the hardware what to do. Software can be broadly categorized into two main types:

➔ **1. System software:** This includes operating systems like Windows, macOS, and Linux. Device drivers and utilities that manage computer resources and provide a platform for running application software. System software acts as interface between the hardware the hardware and the user.

➔ **2. Application Software:** These are programs designed to perform specific tasks for the user, such as word processors, web browsers, games, and business application like accounting software.

➔ Software is typically created using programming languages, such as python, java, or C++, and is stored in various like source code, executable and scripts.

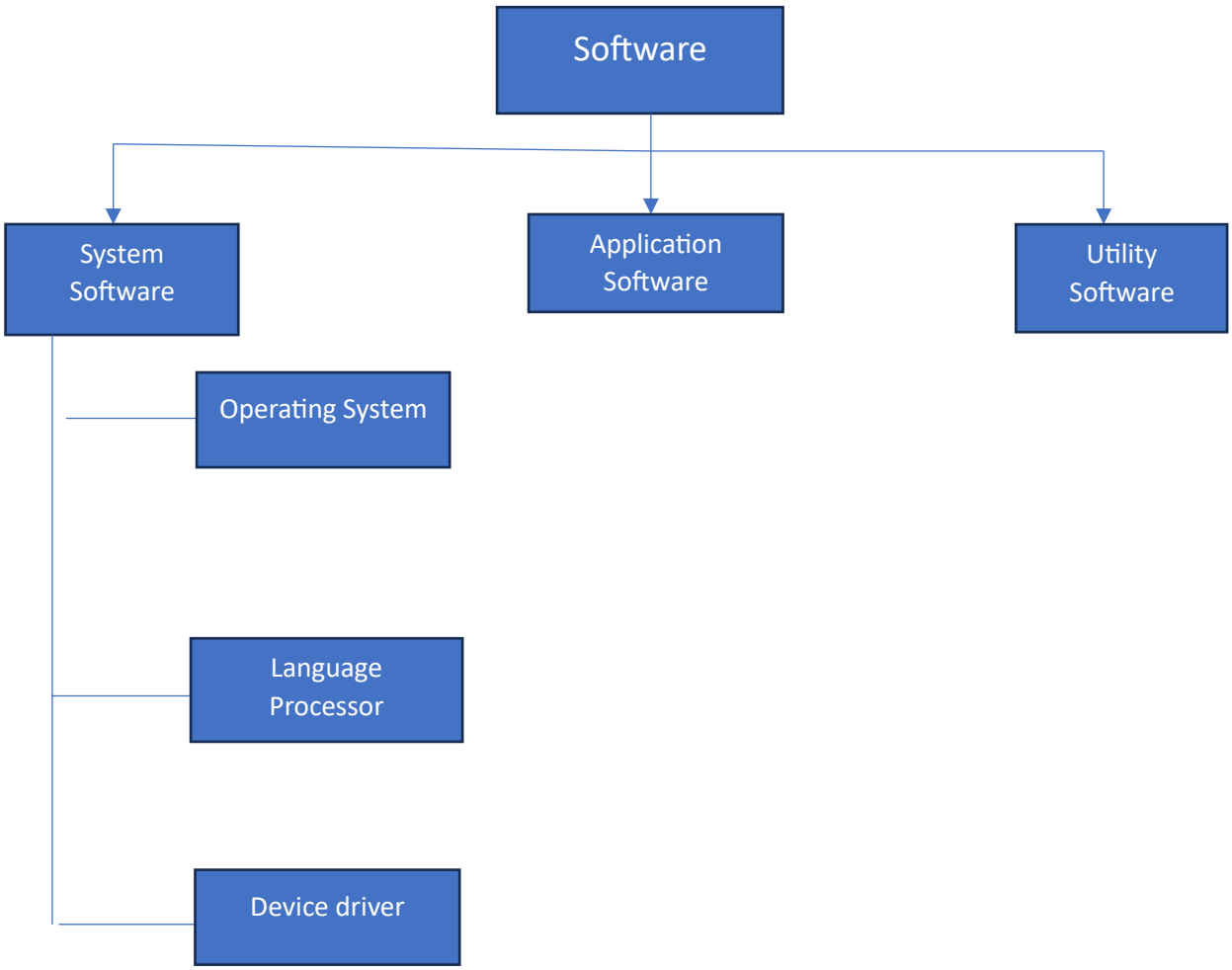
➔ Software engineering is the discipline of designing, developing, testing, and maintaining software systems in a systematic, efficient, and reliable manner. It combines principles from computer science, engineering, project management, and other fields to create software that meets user needs and performs effectively in a wide range of environment.

➔ Key aspects of software engineering include:

➔ **1. Requirement Analysis**
➔ **2. Designing**
➔ **3. Development**
➔ **4. Testing**
➔ **5. Maintenance**
➔ **6. Project Management**
➔ **7. Quality Assurance**

➔ Software engineering emphasizes the use of best practices, methodologies, and tools to create software that is scalable, maintainable, and adaptable to changing requirements. The goal is to produce high-quality software that meets needs of user and can be efficiently developed and maintained over its lifecycle.

Q-2 Explain types of software.



1. **System Software:**

➔ **Operating System:** These are the most fundamental type of system software that manage hardware resources and provide a platform for application software to run. Examples include Windows, macOS, Linux, and Android.

➔ **Language Processor:** Language processor software is also known as a language translator, is a type of system software that translates high level programming languages into machine code, which is executable by a computer's hardware. This process is essential because computers only understand machine code, which consists of binary instructions. Language processors play a critical role in the development and execution of software.

There are three main types of language processor:

1. **Compiler**
2. **Interpreter**
3. **Assembler**

➔ **Device driver:** These are specialized programs that allow the operating system to communicate with hardware devices like printers, graphics cards, and network adapters.

2. **Application software:** Application software is a type of computer program designed to help users perform specific tasks. these task can range from writing documents and browsing the internet to playing games and managing finances. Examples of application software include Microsoft Word for word processing, Google chrome for web processing, and adobe photoshop for editing images. essentially, application software is what you use on your computer or phone to get things done.

3. **Utility Software:** These programs perform specific tasks that help manage, maintain, and control computer resources. Examples include antivirus software, disk management tools, and backup software**.**

Q-3. What is SDLC? Explain each phase of SDLC.

→ The software development life cycle is a process used to plan, develop, test, and maintain software. It provides a structured approach to building software, ensuring that it meets user needs, is of high quality, and is delivered on time. The SDLC typically involves the following phases:

1. **Planning:** Defining the scope, objectives, and resources needed for the project.
2. **Requirement Analysis:** Gathering and documenting what the users need the software to do.
3. **Designing:** Creating the architecture and detailed design of the software.
4. **Development:** Writing the actual code based on the design.
5. **Testing:** Checking the software for errors and ensuring it works as expected.
6. **Deployment:** Releasing the software to users.
7. **Maintenance:** Updating and fixing the software after it is in use.

→ Each phase has specific goals and deliverables, helping to ensure the project stays on track and meets its objectives. The SDLC can follow different models, such as waterfall, or spiral, depending on the project needs.

Q-4. What is DFD? Create a DFD diagram on Flipkart.

➔ DFD stands of "data flow diagram". It is also known as "bubble char" through which we can represent the flow of data graphically in an information system.

➔ By using DFD we can easily understand the overall functionality of system because diagram represent the incoming data flow, outgoing data flow and stored data in a graphical form.

➔ describes how data is processed in a system in term of input & output.

➔ A DFD model uses a number of notations or symbol to represent flow data.

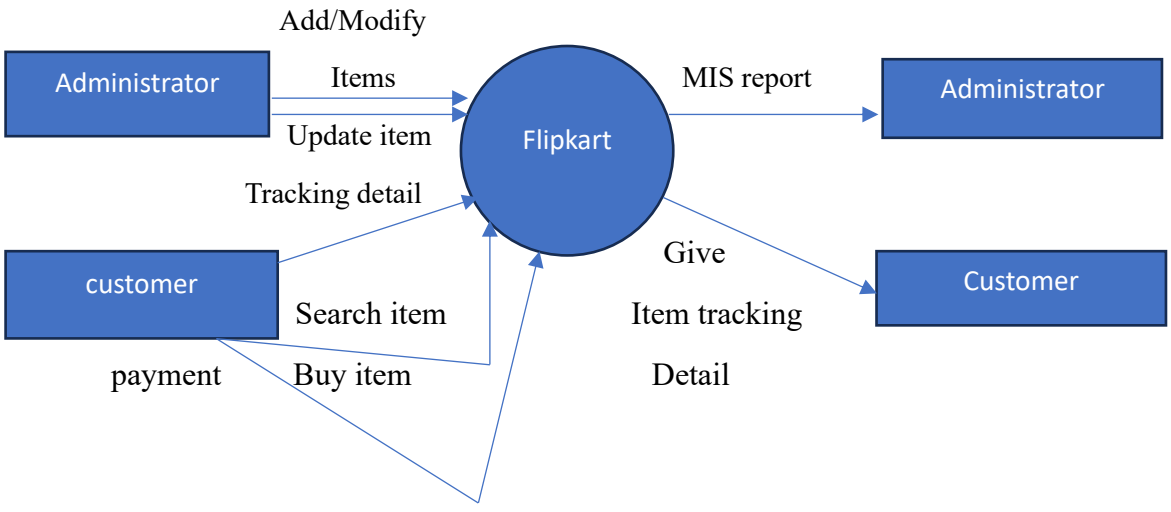
1. External entity:

2. Data flow:

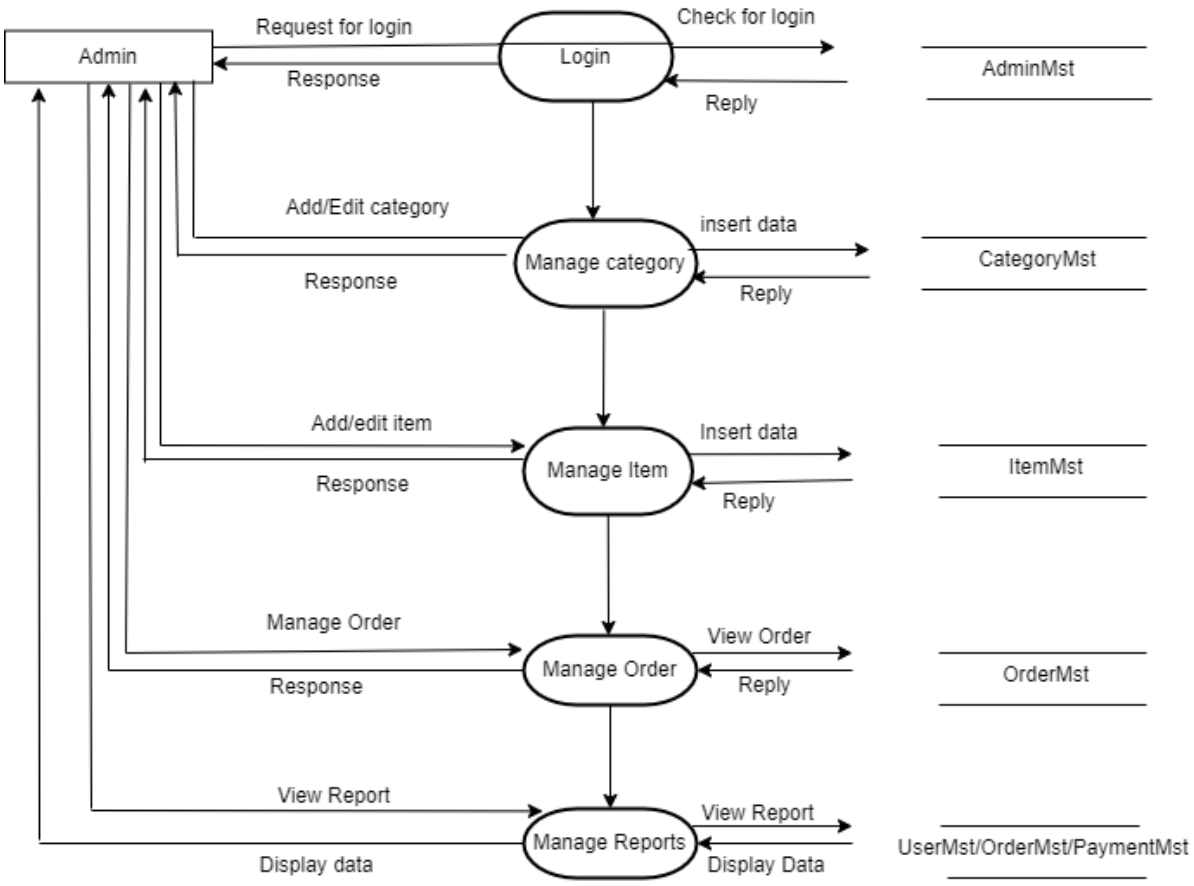
3. Process:

4.Data store:

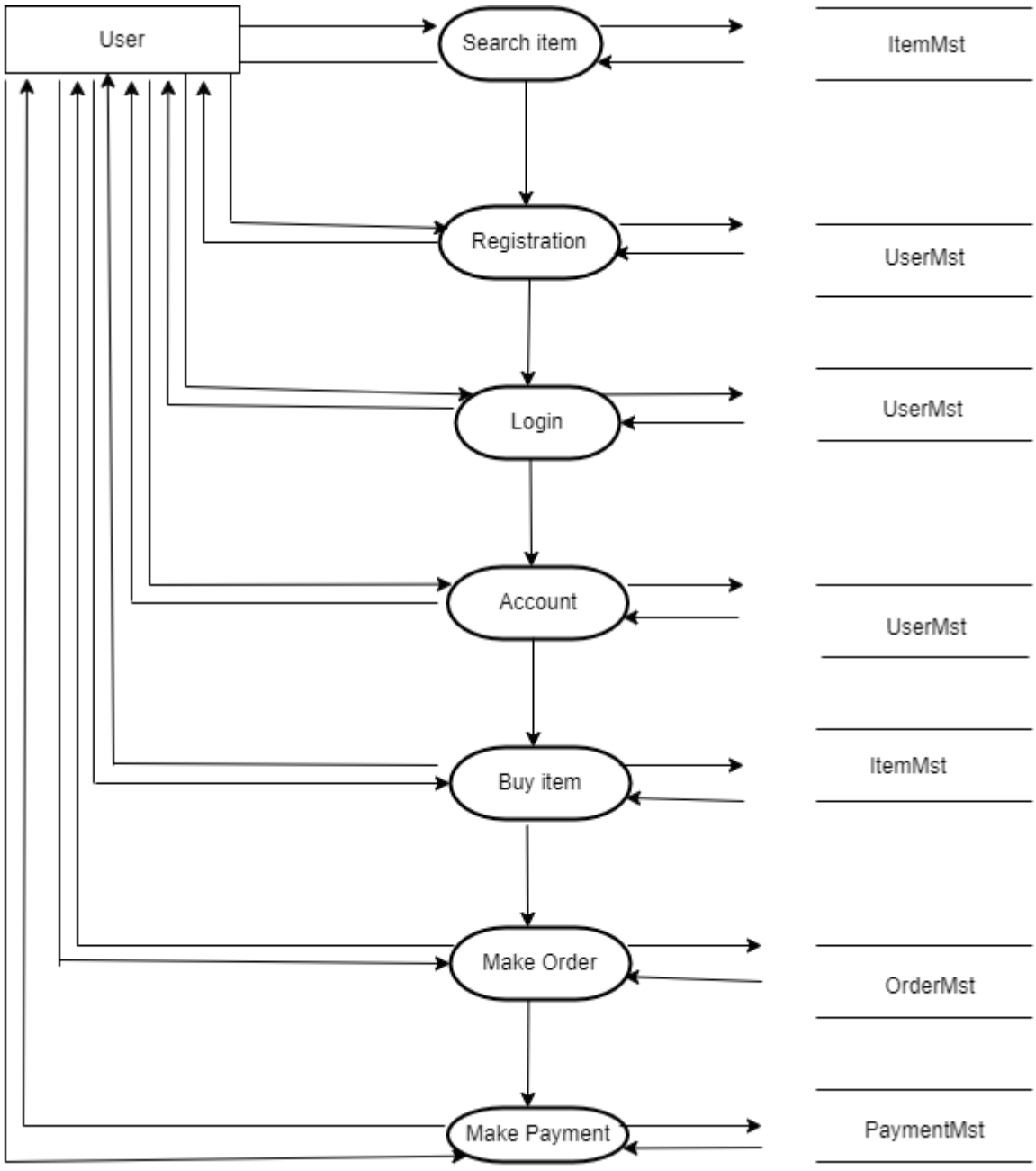
0 – Level DFD:



Admin Side DFD – 1st Level

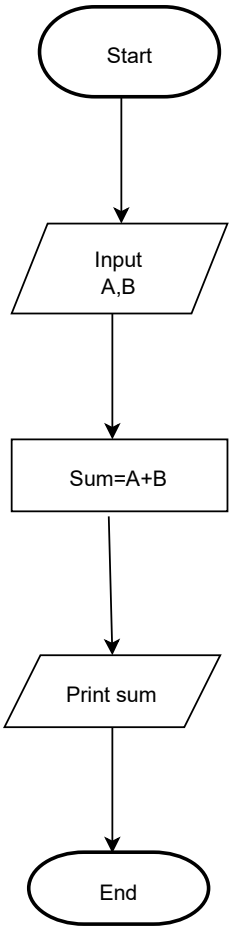


User Side DFD – 1nd Level



Q-5. What is Flowchart? Create a flowchart to make addition of two numbers.

➔ A flowchart is a simple diagram that shows the steps or decisions you take to complete a task. Think of it like a map that helps you figure out what to do next by following arrows from one step to another. Each step is usually shown in a box, and arrows show the flow or direction to the next step.



Q-6. What is Use-case Diagram? Create a use-case on bill payment on paytm.

➔ A use case diagram is a simple picture that shows how a system like website or app interacts with the people or things that use it. It helps you see what the system does from the user's point of view. In the diagram: **Stick figures** represent the users or other systems that interacts with your system. These are called "actors". **Ovals** represent the actions or tasks that the system can perform. these are called "use case". **Lines** connect the actors to the use cases they interact with, showing what each user can do.

➔ For example, in a use case diagram for an online shopping site, a stick figure labeled "customer" might connect to ovals labeled "browse products", "add to cart", and "checkout". This shows that a customer can perform these actions on the site.