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

* Software engineering defines software as computer programs,

procedures, rules and possibly associated documentation and

data planning to the operation of a computer system.

Software can also be defined as

1) Computer programs when executed provide desired results

and performance

2) Data structure that enable the programs to follow

manipulation

3) Documents that describe the operations and use of the

program

* Software delivers the most important product of our time –

information.

* **Software Engineering** is the systematic and disciplined process of designing, developing, testing, and maintaining software. Here are some key points:
  1. **Design and Development**: Software engineers create blueprints for software systems. They decide how different components will work together, plan the architecture, and write code to bring the software to life.
  2. **Testing**: Rigorous testing ensures that the software meets its requirements and is free of bugs. This step is crucial to ensure reliability and quality.
  3. **Maintenance**: Software needs regular updates, bug fixes, and improvements. Software engineers continuously maintain and enhance the software even after it’s deployed.
  4. **Engineering Principles**: Software engineering applies engineering principles to software development. Engineers use systematic methods to build robust, efficient, and reliable software solutions.
  5. **User-Centric Approach**: Software engineers focus on creating software that serves end users. They consider usability, performance, security, and scalability.
  6. **Tools and Techniques**: Software engineering employs various tools, methodologies, and best practices. These include modular design, abstraction, encapsulation, and design patterns.
  7. **Agile Methodologies**: Agile approaches emphasize flexibility, customer satisfaction, and iterative development. They allow teams to adapt to changing requirements.
  8. **Continuous Integration & Deployment**: Engineers integrate code changes frequently and deploy them into production environments.
  9. **Attributes**: Software engineering is systematic, disciplined, and quantifiable. It involves the entire lifecycle of a software system, from inception to maintenance.

Q-2. Explain types of software?

* APPLICATION SOFTWARE
* SYSTEM SOFTWARE
* DRIVER SOFTWARE
* MIDDLEWARE
* PROGRAMMING SOFTWARE

1. 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. An application can be self-contained, or it can be a group of programs that run the application for the user. Examples of [modern applications](https://www.techtarget.com/searchcio/feature/The-rise-of-modern-applications-Why-you-need-them) include office suites, graphics software, databases and database management programs, web browsers, word processors, software development tools, image editors and communication platforms.

1. SYSTEM SOFTWARE

These software programs are designed to run a computer's application programs and hardware. System software coordinates the activities and functions of the hardware and software. 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](https://www.techtarget.com/whatis/definition/firmware), computer language translators and system [utilities](https://www.techtarget.com/whatis/definition/utility).

1. 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.

1. 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.

1. PROGRAMMING SOFTWARE

Computer programmers use programming software to write code. Programming software and programming tools enable developers to develop, write, test and [debug](https://www.techtarget.com/searchsoftwarequality/definition/debugging) other software programs. Examples of programming software include assemblers, compilers, debuggers and interpreters.

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



**Software Development Life Cycle (SDLC)** is a systematic process used by software development organizations to plan, design, develop, test, deploy, and maintain software applications. Let’s delve into the details of each phase:

1. **Requirements Gathering and Analysis**:
   * In this initial phase, information about the software requirements is collected from stakeholders, including customers, end-users, and business analysts.
   * The goal is to understand what the software needs to achieve, its functionalities, and the constraints.
   * Activities include interviews, surveys, and documentation to create a comprehensive understanding of the project.
2. **Design**:
   * The design phase involves creating the blueprint for the software.
   * It includes two steps:
     + **High-Level Design (HLD)**: This step defines the overall architecture of the software, including its components, modules, and interactions.
     + **Low-Level Design (LLD)**: Here, detailed specifications are created for each feature, describing how they should work and how components interact.
   * The design phase ensures that the software will meet the requirements and be scalable, maintainable, and efficient.
3. **Implementation/Coding**:
   * In this phase, developers write the actual code based on the design specifications.
   * It involves translating the design into executable code using programming languages and tools.
   * Proper coding practices, adherence to coding standards, and documentation are essential.
4. **Testing**:
   * The testing phase ensures that the software functions correctly and meets quality standards.
   * Different types of testing (unit testing, integration testing, system testing, etc.) are performed to identify and fix defects.
   * Quality assurance activities help ensure the software is free of bugs and performs as expected.
5. **Deployment**:
   * Once the software passes testing, it is deployed to the production environment.
   * Deployment involves installing the software on servers, configuring databases, and making it accessible to users.
   * Proper deployment procedures are crucial to avoid disruptions and ensure a smooth transition.
6. **Maintenance**:
   * The maintenance phase involves ongoing support, bug fixes, and updates.
   * Regular monitoring, addressing user feedback, and making necessary enhancements fall under maintenance.
   * It ensures the software remains reliable, secure, and up-to-date.

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

**DFD** stands for **Data Flow Diagram**. It is a graphical tool used to represent the flow of data within a system or process. Here are some key points about DFDs:

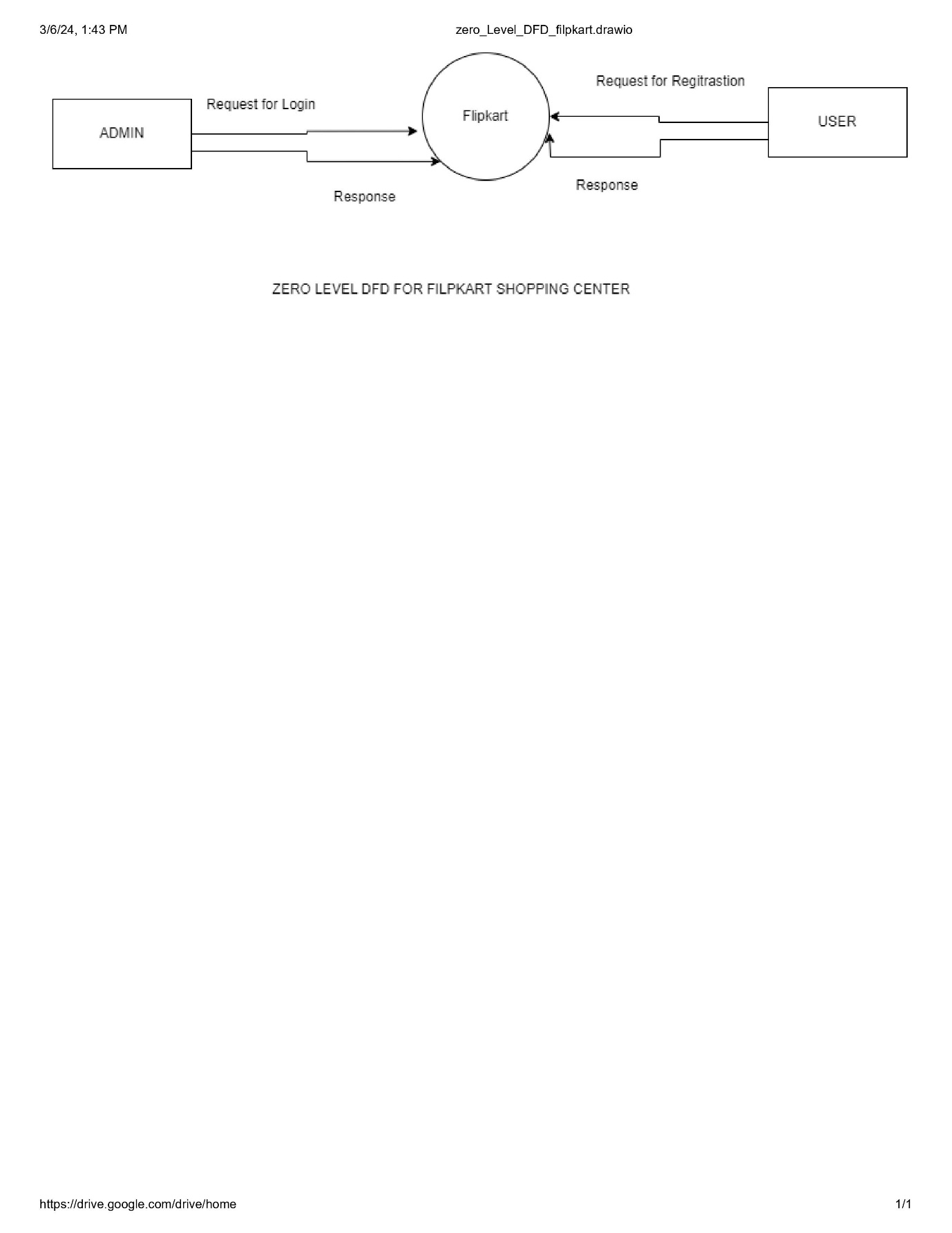
1. **Purpose and Representation**:

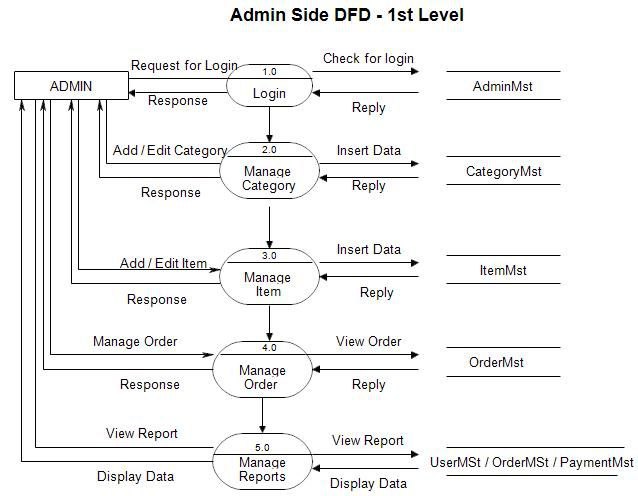
* A DFD illustrates how data moves through a system or process, providing insights into the inputs, outputs, and transformations at each stage.
* Unlike flowcharts, DFDs do not include control flow, loops, or decision rules.
* They are useful for communicating with users, managers, and other personnel during system analysis.
* DFDs help analyse both existing and proposed systems.

1. **Components of DFD**:
   * **Process**: Represents the input-to-output transformation within the system. Processes are depicted as rectangles with rounded corners, ovals, rectangles, or circles.
   * **Data Flow**: Describes the movement of information between different parts of the system. Arrows represent data flows, and each flow should transfer a single type of information.
   * **Warehouse**: Stores data for later use.
2. **Characteristics of DFD**:
   * DFDs are commonly used during problem analysis and understanding a system.
   * They view a system as a function that transforms inputs into desired outputs.
   * Processes are shown as named circles, and data flows are represented by named arrows.
   * Source/sink rectangles represent data origins or consumers outside the main system.
3. **Advantages and Usage**:
   * DFDs help discover potential problems, improve efficiency, and develop better processes.
   * They are widely used in software engineering to visualize major steps and data involved in software-system processes.

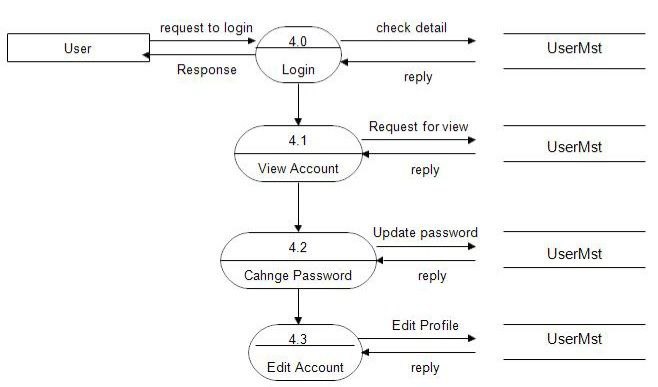


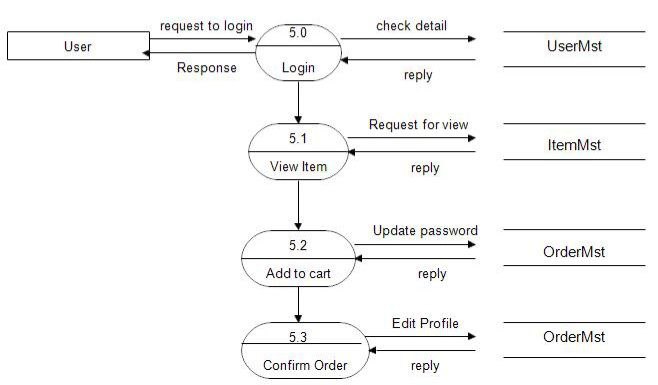
ZERO LEVEL DFD ON FILPKART





**User Side DFD – 2 Level**

****

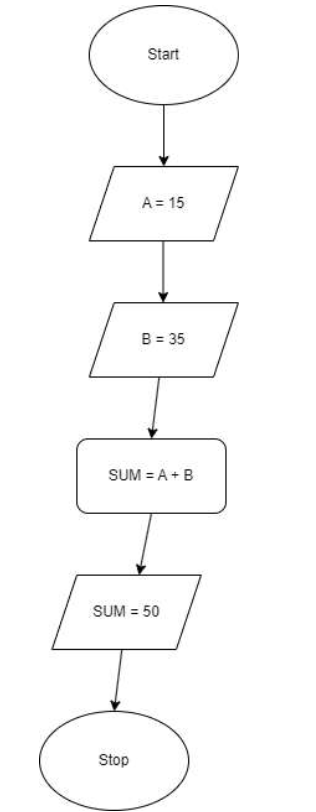


Q-5. What is Flow chart? Create a flowchart to make addition of two numbers

A flowchart is a graphical representation of steps or processes. It visually depicts the sequence of actions in a systematic manner. Flowcharts are widely used in various fields, including software development, business processes, and problem-solving. Let’s explore how to create a flowchart for adding two numbers:

**Flowchart:**

* A flowchart consists of different shapes that represent specific actions or decisions. Here are some common flowchart symbols:
* Terminator: Represents the starting or ending point of the system.
* Process Box: Indicates a specific operation or task.
* Decision Diamond: Represents a decision or branching point.
* Data: Represents information entering or leaving the system.
* Flow Lines: Connect the shapes to show the flow of the sequence.



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

A Use Case Diagram is a type of Unified Modeling Language (UML) diagram that represents the interaction between actors (users or external systems) and a system under consideration to accomplish specific goals. It provides a high-level view of the system’s functionality by illustrating the various ways users can interact with it.

**Use Case Diagram components:**

Actors:

* Actors are external entities that interact with the system. These can include users, other systems, or hardware devices.
* In the context of a Use Case Diagram, actors initiate use cases and receive the outcomes.

Use Cases:

* Use cases represent specific things your system can do. They are like scenes in a play.
* Examples of use cases for an online shopping system could be “Place Order,” “Track Delivery,” or “Update Product Information.” Use cases are represented by ovals in the diagram.

System Boundary:

* The system boundary defines what is inside the system and what is outside. It outlines the scope of the system.
* It helps establish a clear distinction between the elements that are part of the system and those that are external to it.
* The system boundary is typically represented by a rectangular box that surrounds all the use cases of the system.

**UseCase Diagram of PAYTM:**

