- By Akash prajapati

Software Engineering Assignment

MODULE: 1

SE – Overview of IT Industry

Q1. What is software ? What is software engineering ?

➡ **Ans.** Software is a set of instructions (input) to be executed (process) for specific task (output).

- **Software Engineering** is Process to develop any software/Product based on the engineering principles.

1. Software engineering includes a variety of techniques, tools, and methodologies, including requirements analysis, design, testing, and maintenance.

2. Software Engineering is mainly used for large projects based on software systems rather than single programs or applications.

3. The main goal of Software Engineering is to develop software applications for improving quality, budget, and time efficiency.

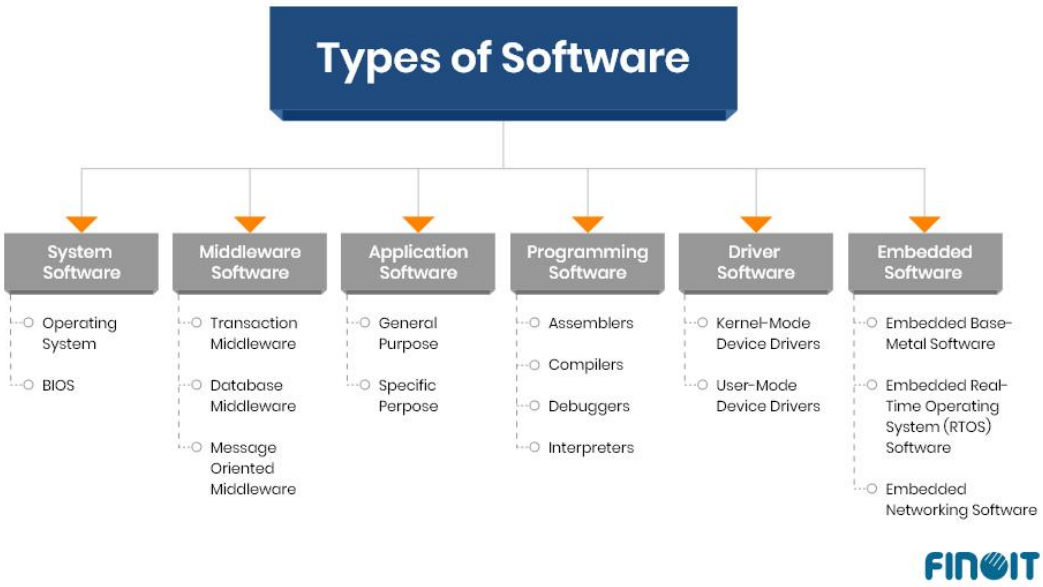
Q.2 Explain types of software.

- **Ans**. The Main Two Categories of Software :
  1. System Software
  2. Application Software.

1. **System Software :** System software is a type of computer software designed to provide a platform for running application software and manage hardware resources.

2. Application Software : It is provided by the developers and we need Download Or Install To use.
Ex. Instagram,linkdin

- And other software Categories are given in the Below chart :



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

- **Ans.** Software Development Life Cycle is a step by step approach to development any product/software with high quality, in shortest possible time and within the estimated budget.

- **SDLC** phases Are given in the Below :

1. Planning / Requirement Gathering
2. Analysis
3. Design
4. Coding
5. Testing
6. Maintenance

Here's an explanation of each phase in the SDLC:

1. **Requirement Gathering and Analysis :**

- **Objective**: To collect and define the software requirements from all stakeholders (end users, business owners, etc.). This phase involves understanding what the user needs and ensuring that these requirements are clear, complete, and feasible.
- **Activities**:
  - Conducting meetings, surveys, and interviews with stakeholders.
  - Documenting functional and non-functional requirements.
  - Analyzing the feasibility of the project (technical, operational, and financial feasibility).
- **Output**: A **Requirements Specification Document** that outlines what the software will do, how it will behave, and any constraints (budget, time, technology, etc.).

2. System Design

- **Objective**: To translate the requirements into a blueprint for building the software. This phase focuses on designing the system architecture, components, and data flows.
- **Activities**:
  - Creating high-level design (architecture, database schema, data flow diagrams, user interfaces).
  - Detailing the system components, modules, and interactions.
  - Defining the hardware and software specifications needed for implementation.
- **Output**: Design Documents that include architectural diagrams, data models, and design specifications.

3. Implementation (Coding)

- **Objective**: To translate the design into actual code that the computer can execute. This phase involves writing the code for the software modules as per the design specifications.
- **Activities**:
  - Writing code in the chosen programming languages (e.g., Java, Python, C++).
  - Following coding standards and guidelines.
  - Performing unit testing to verify that individual components work as expected.
- **Output**: The **software code** (source code) that implements the system's functionalities.

4. Testing

- **Objective**: To ensure the software works as intended and is free of bugs. Testing helps identify and fix any defects or issues that might affect the software's functionality, performance, or security.
- **Activities**:

  - Performing various types of testing such as unit testing, integration testing, system testing, acceptance testing, and performance testing.
  - Debugging errors or issues found during testing.
  - Verifying that the software meets the original requirements and business needs.
- **Output**: **Test Reports** and **Bug Fixes**, ensuring that the software is stable, functional, and secure.

5. Deployment

- **Objective**: To make the software available for use by the end users. This phase involves installing the software in the production environment and ensuring it functions as expected in a real-world setting.
- **Activities**:
  - Preparing the deployment environment (servers, cloud, etc.).
  - Installing the software and configuring it as per the requirements.
  - Training users on how to use the new system, if necessary.
- **Output**: The **Deployed Software** in the production environment, available to end-users.

6. Maintenance

- **Objective**: To provide ongoing support for the software after it has been deployed. This phase ensures the software continues to function well, receives updates, and any issues are addressed.
- **Activities**:
  - Monitoring software performance and fixing bugs or issues that arise.
  - Releasing updates to improve functionality, security, or performance.
  - Making modifications to adapt to new requirements or technologies.
- **Output**: **Patches** or **Updates** to the software as needed.

Q.4 What is DFD? Create a DFD diagram on Flip kart.

- **Ans.** A **Data Flow Diagram (DFD)** is a graphical representation that depicts the flow of data within a system. It shows how data moves between processes, data stores, external entities (e.g., users or other systems), and data flows between them. DFD are useful for understanding the functional aspects of a system without focusing on the technical details of implementation.

A DFD typically includes:

1. **Processes** : Actions or operations performed on the data (represented by circles or ovals).
2. **Data Flows** : Movement of data between processes, data stores, and external entities (represented by arrows).
3. **Data Stores** : Repositories where data is stored (represented by open-ended rectangles).
4. **External Entities** : Actors (like users or external systems) that interact with the system (represented by rectangles).

Levels of DFD:

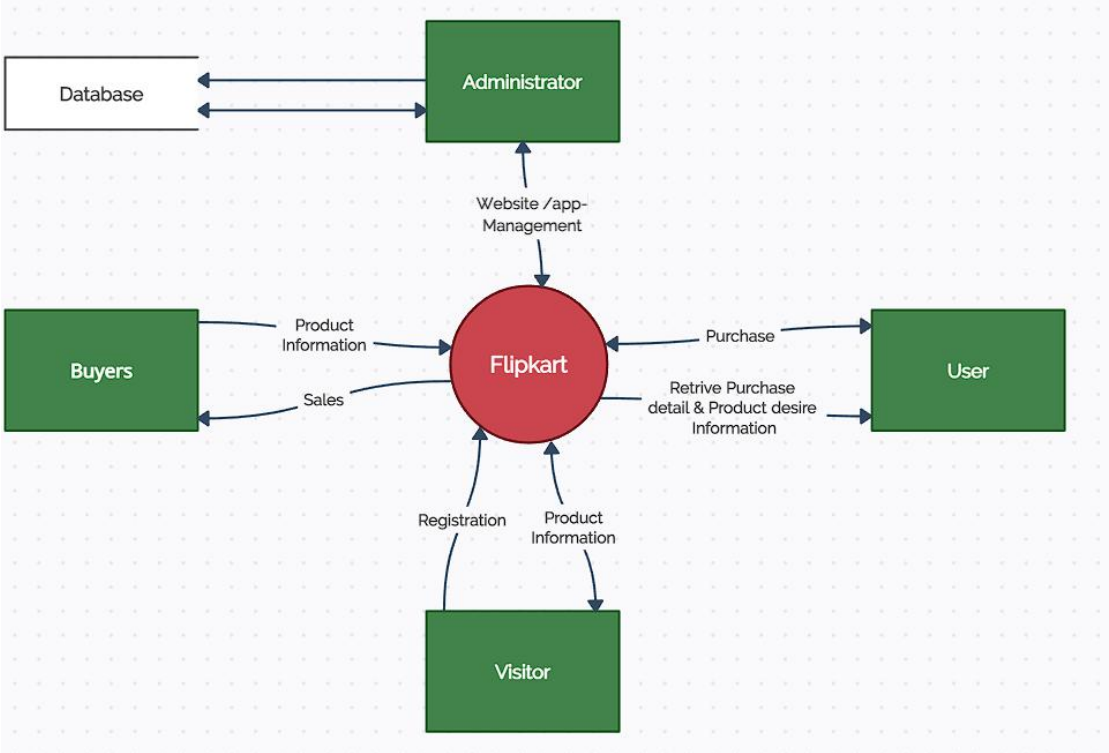
1. **Context Diagram (Level 0 DFD)**: Provides a high-level overview of the system, showing only the main process and its interactions with external entities.

2. **Level 1 DFD**: Breaks down the high-level process into sub-processes, showing more detailed interactions.

3. **Level 2 DFD and beyond**: Further decomposes the sub-processes into more detailed levels of data flow.

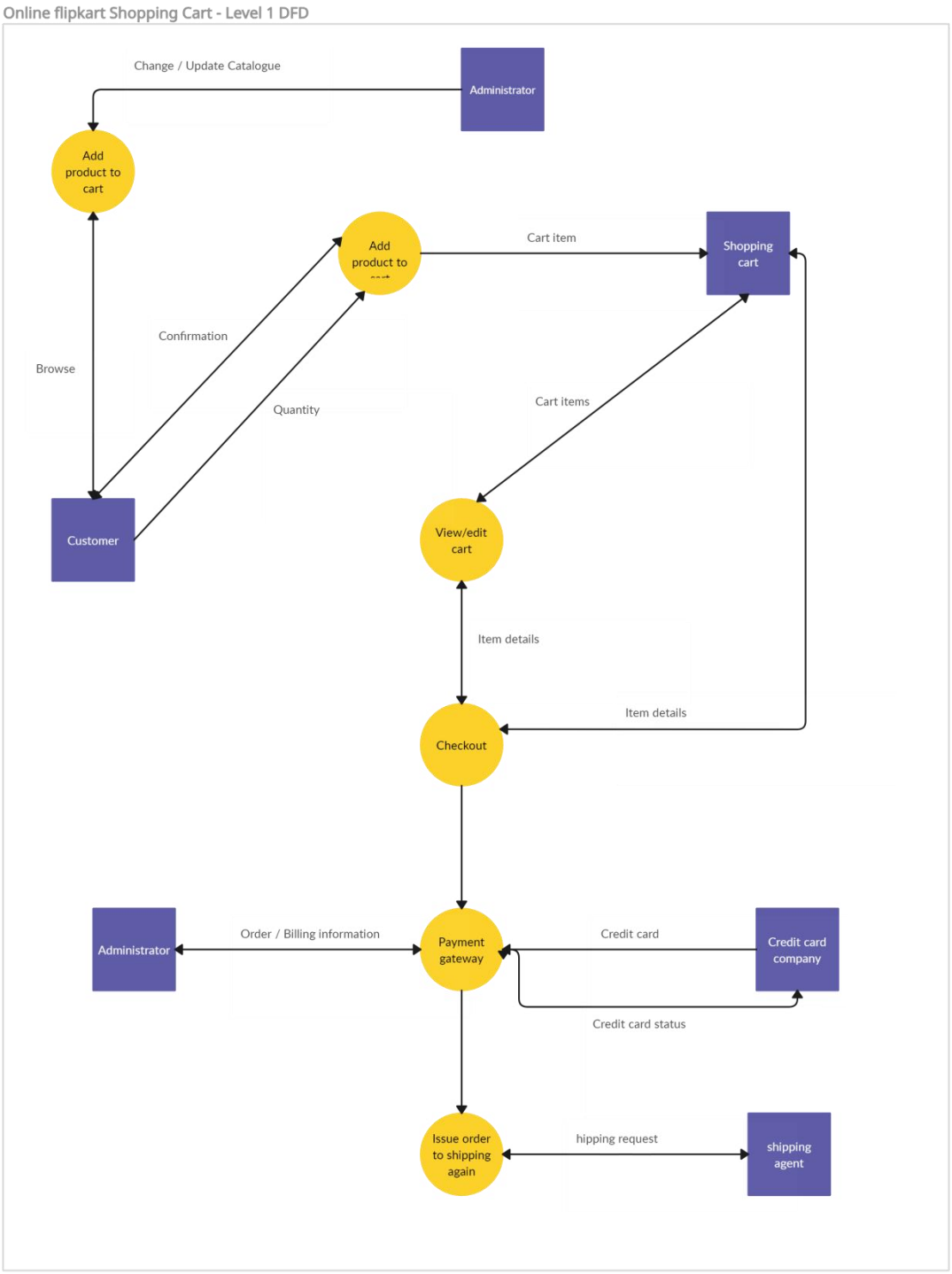
Level 0 DFD [Data Flow Diagram] of Flip cart

- By Akash prajapati



Level 1 DFD [Data Flow Diagram] of online Flipkart shopping cart

- By Akash prajapati



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

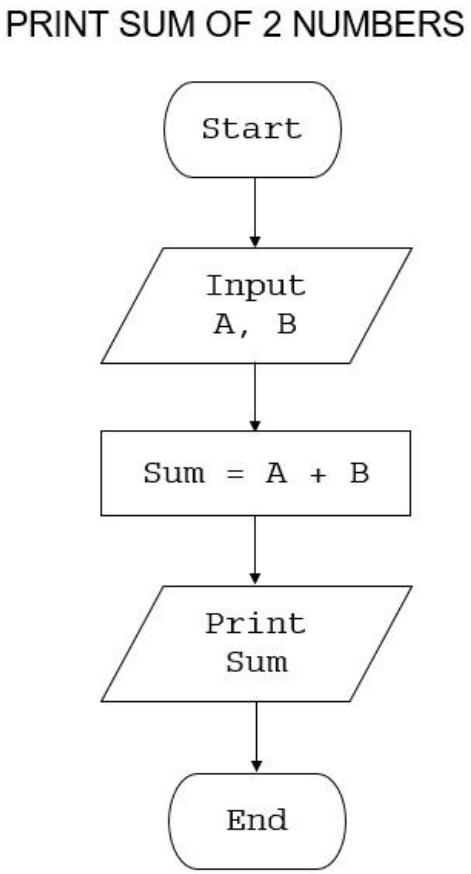
- **Ans.** A Flowchart is a type of diagram that represent a Workflow or process. A flowchart also can be a diagrammatic representation of an algorithm , a step by step approach to solving a task.

- A flowchart (or flow chart) is a diagram that shows the steps in a process.

- Flowcharts are often used for visualizing the sequence of actions or information needed for training,documenting, planning, and decision-making.

- They often use symbols, shapes, and arrows to illustrate how one step leads to another.

- **This straightforward flowchart shows how to add two numbers :**



In this flowchart:

- "Start" is represented by an oval or rounded rectangle.
- "Input" A and B is represented by a parallelogram.
- "Sum" A and B is represented by a rectangle.
- "print sum" is represented by a parallelogram.
- "End" is represented by an oval or rounded rectangle.

Arrows connecting each step show how the steps flow into one another.

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

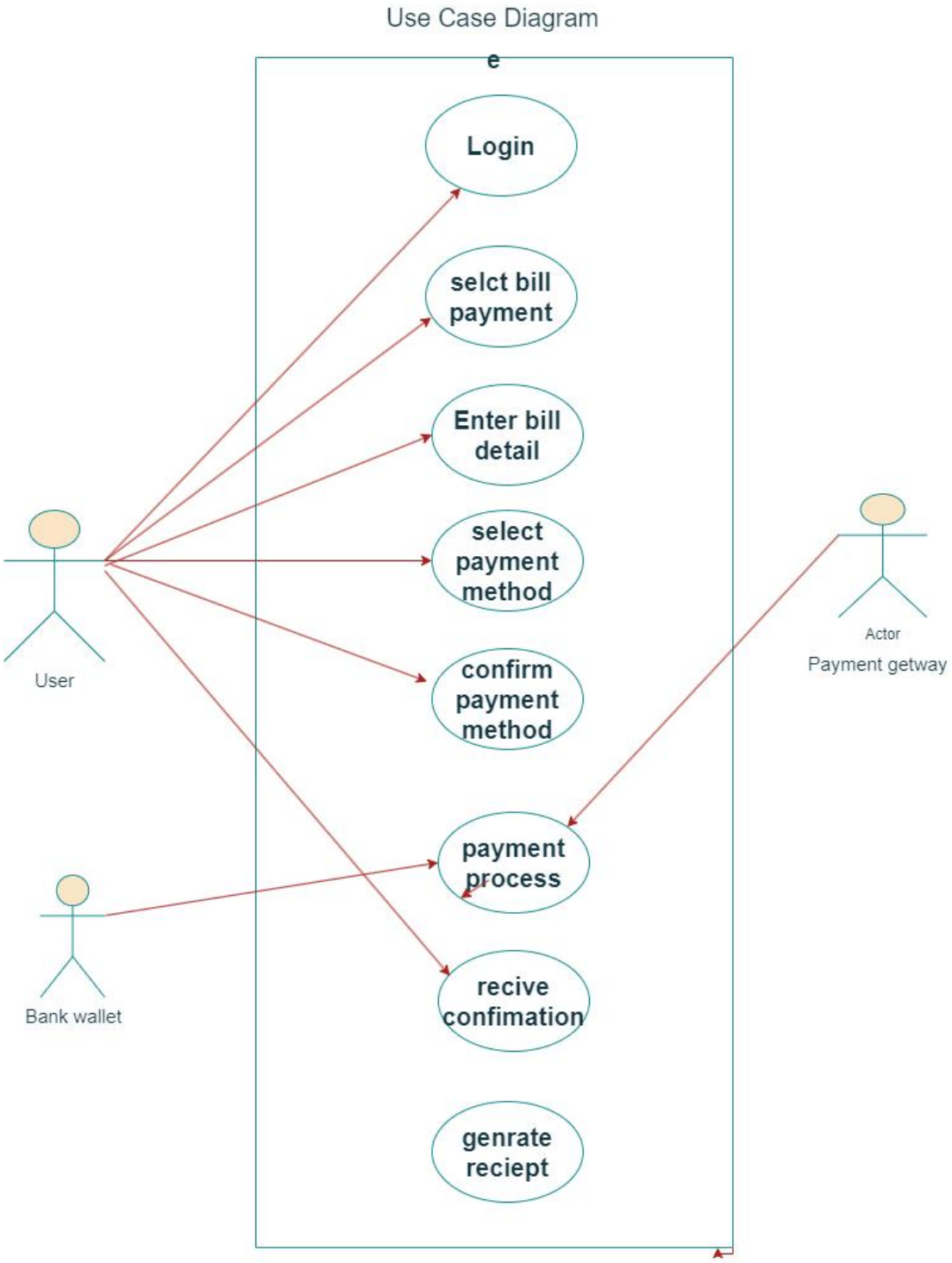
➢ **Ans.** A **Use Case Diagram** is a visual representation of the functional requirements of a system. It shows the interactions between "actors" (users or other systems) and the system itself, represented by "use cases" (functions or operations the system performs). Use case diagrams help clarify the scope of a system, highlighting what the system will do and who will interact with it.

- **Actors** are entities that interact with the system (e.g., users, external systems).
- **Use Cases** are the specific tasks or services the system provides to the actors.

Use Cases:

1. **Login** (User logs into Paytm)
2. **Select Bill Payment Option** (User chooses to pay a bill)
3. **Enter Bill Details** (User provides details like biler name, amount, etc.)
4. **Choose Payment Method** (User selects payment method—credit/debit card, UPI, Paytm Wallet)
5. **Confirm Payment** (User confirms payment details)
6. **Process Payment** (Paytm processes the payment through the payment gateway and communicates with the bank/wallet)
7. **Receive Payment Confirmation** (User receives a confirmation of the successful payment)
8. **Generate Receipt** (User can view or download a payment receipt)

- By Akash prajapati



Use case diagram for paytm process