**ASSIGNMENT 1**

**Q.1. What is Software? What is Software Engineering?**

**Software:**

- is a collection of computer programs and related data that provide the instructions for telling a computer what to do and how to do it.

- Just like human language.

- Software refers to a collection of instructions, programs, and data that perform specific tasks on a computer or other devices. It can be categorized into three main types:

1) System software / operating system

2) Application s/w

3) Programming language

**1) System Software:** This type of software manages and controls the hardware components of a computer system, providing a platform for running application software.

- Examples include operating systems like Windows, macOS, and Linux, as well as device drivers and utility programs.

**2) Application Software:** Application software is designed to perform specific tasks or functions for end-users. Including different types of application software.

**•** Mobile app:

- Application that run on mobile

- Ex. Instagram, Facebook etc

**•** Desktop app:

- That run stand-alone in a desktop or laptop computer.

- Ex. Microsoft office suite which includes Word, Excel and PowerPoint.

- Ex. Outlook for email, and Firefox, Google Chrome, Mozilla are the web browser.

**•** Web app:

- That run on a web browser

- ex. google.com, facebook.com, etc

**3) Programming Languages:** Programming languages are formal languages used to write instructions or code that control the behavior of computers and other devices. They provide a way for programmers to communicate with computers and create software applications.

**Software Engineering:**

- Software: A Program or set of Programs containing instructions which provide desired functionality.

- Engineering: Process of designing and building something that ensure particular purpose.

- Software engineering is the art of developing quality software on time and within budget.

- Software engineering is a systematic and disciplined approach to the design, development, testing, and maintenance of software. It involves applying engineering principles and practices to software development processes to ensure that software is reliable, scalable, maintainable, and meets the needs of its users.

**Q.2. Explain types of Software**

- Certainly! Software can be categorized into several types based on its functionality, purpose, and usage. Here are some common types of software:

**1) System Software:**

- System software is essential for managing and controlling computer hardware and providing a platform for running application software. Examples include:

* **Operating Systems:** Such as Windows, macOS, Linux, iOS, and Android, which manage hardware resources, provide a user interface, and support the execution of applications.
* **Device Drivers:** Software that facilitates communication between hardware devices (like printers, graphics cards, and network adapters) and the operating system.
* **Utilities:** Tools and programs that perform maintenance tasks, manage system resources, and enhance system performance, disk management tools, and file compression.

**2) Application Software:**

- Application software is designed to perform specific tasks or functions for end-users. It includes a wide range of programs tailored to various purposes, such as:

* **Productivity Software:** Applications used for creating, editing, and managing documents, spreadsheets, presentations, and databases. Examples include Microsoft Office, Google Workspace, and Adobe Creative Suite.
* **Communication Software:** Tools for communicating and collaborating with others, including email clients, instant messaging apps, video conferencing software, and social media platforms.
* **Entertainment Software:** Software for entertainment and leisure activities, such as video games, multimedia players, streaming services, and digital art tools.
* **Educational Software:** Programs designed to facilitate learning and teaching, including interactive tutorials, educational games, simulation software, and virtual learning environments.
* **Business Software:** Applications used for managing business operations and processes, such as enterprise resource planning (ERP) systems, customer relationship management (CRM) software, and accounting software.
* **Utilities:** Specialized tools and applications that serve specific purposes, such as antivirus software, file management tools, backup software, and system optimization utilities.

**3) Programming Software:**

- Programming software provides tools and environments for writing, testing, debugging, and maintaining software code. Examples include:

* **Integrated Development Environments (IDEs):** Software suites that combine code editors, compilers, debuggers, and other development tools into a unified environment. Examples include Visual Studio, Eclipse, and IntelliJ IDEA.
* **Text Editors:** Lightweight tools for writing and editing code, with features like syntax highlighting, code completion, and version control integration. Examples include Sublime Text, Atom, and VS Code.
* **Compilers and Interpreters:** Software that translates high-level programming languages into machine code or intermediate code that can be executed by a computer. Examples include GCC (GNU Compiler Collection), Python interpreter, and Java Virtual Machine (JVM).
* **Version Control Systems (VCS):** Software for tracking changes to source code files, managing revisions, and coordinating collaboration among developers. Examples include Git, Subversion (SVN), and Mercurial.

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

- SDLC stands for Software Development Life Cycle. It's a structured process for planning, creating, testing, and deploying software applications. SDLC aims to produce high-quality software that meets customer expectations, is delivered on time, and stays within budget.

- A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the development and lifecycle management of an application or piece of software. There are several commonly recognized phases in the SDLC process:



**1) Planning:**

* **Purpose:**

- Define the project's scope, objectives, and requirements.

* **Activities:**

- Conduct feasibility studies to determine the project's viability.

- Gather requirements through interviews, surveys, and analysis of existing systems.

- Define project milestones, timelines, and deliverables.

* **Deliverables:**

- Project charter, feasibility study report, requirements specification document.

**2) Analysis:**

* **Purpose:**

- Understand the users' needs and define the system's functionality.

* **Activities:**

- Analyze gathered requirements to identify system features and constraints.

- Model the system's behavior and structure using techniques like use cases, data flow diagrams, and entity-relationship diagrams.

- Define the system's architecture and design principles.

* **Deliverables:**

- System requirements specification, functional specification, architectural design document.

**3) Design:**

* **Purpose:**

- Develop a detailed blueprint for implementing the system.

* **Activities:**

- Design the user interface, database schema, and software components.

- Specify algorithms, data structures, and programming languages.

- Create prototypes or mockups to visualize the system's appearance and behavior.

* **Deliverables:**

- Detailed design documents, interface prototypes, database schema.

**4) Implementation:**

* **Purpose:**

- Translate the design into actual code.

* **Activities:**

- Write, test, and debug the software modules according to the design specifications.

- Integrate individual components into a cohesive system.

- Conduct code reviews and refactor code for clarity, efficiency, and maintainability.

* **Deliverables:**
* Source code, executable files, unit test cases, integration test results.

**5) Testing:**

* **Purpose:**

- Verify that the software meets the specified requirements and functions correctly.

* **Activities:**

- Develop test plans and test cases based on requirements and design specifications.

- Execute tests to identify defects, errors, and deviations from expected behavior.

- Conduct functional testing, performance testing, security testing, and other types of testing as needed.

* **Deliverables:**

- Test plans, test cases, defect reports, test results.

**6) Maintenance:**

* **Purpose:**
* Ensure the long-term usability and reliability of the software.
* **Activities:**
* Monitor the software in production to identify and fix bugs, errors, and performance issues.
* Implement updates, patches, and enhancements to address changing requirements and technologies.
* Provide user support, troubleshooting assistance, and training as needed.
* **Deliverables:**

- Software updates, bug fixes, maintenance reports.

- These phases may overlap or be iterative, depending on the project's complexity, requirements, and development methodology (e.g., Waterfall, Agile, DevOps). The SDLC provides a structured framework for managing software projects from inception to retirement, ensuring that they are delivered efficiently and effectively.

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

- DFD stands for Data Flow Diagram, which is a graphical representation of the flow of data through a system.

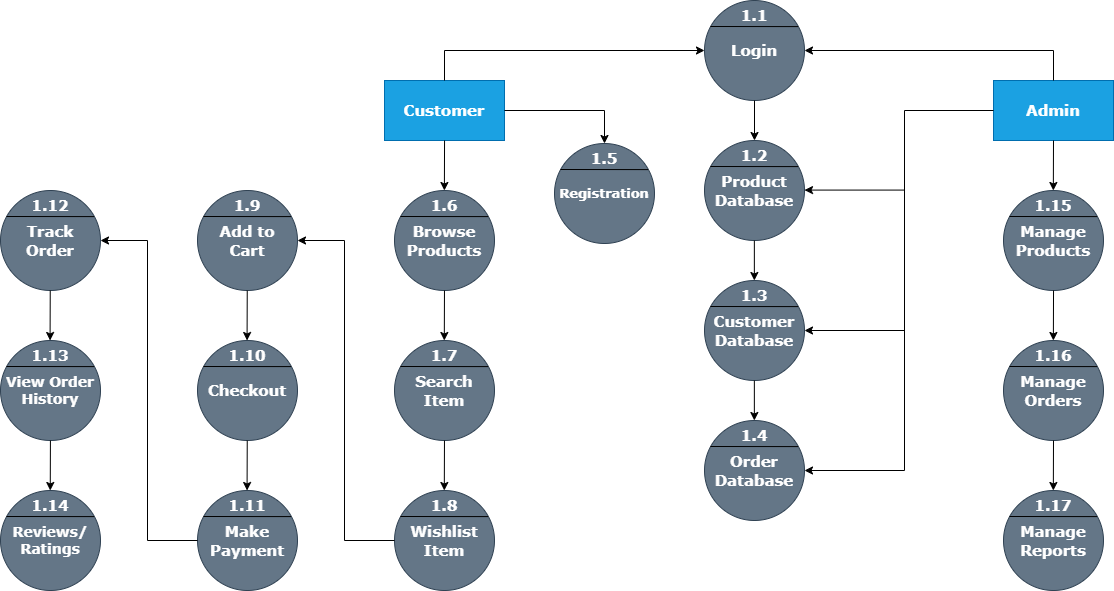
- It's commonly used in software engineering and systems analysis to model the flow of information within a system.

- DFDs are powerful tools for visualizing how data moves through a system and are particularly useful for understanding the processes involved in complex systems.

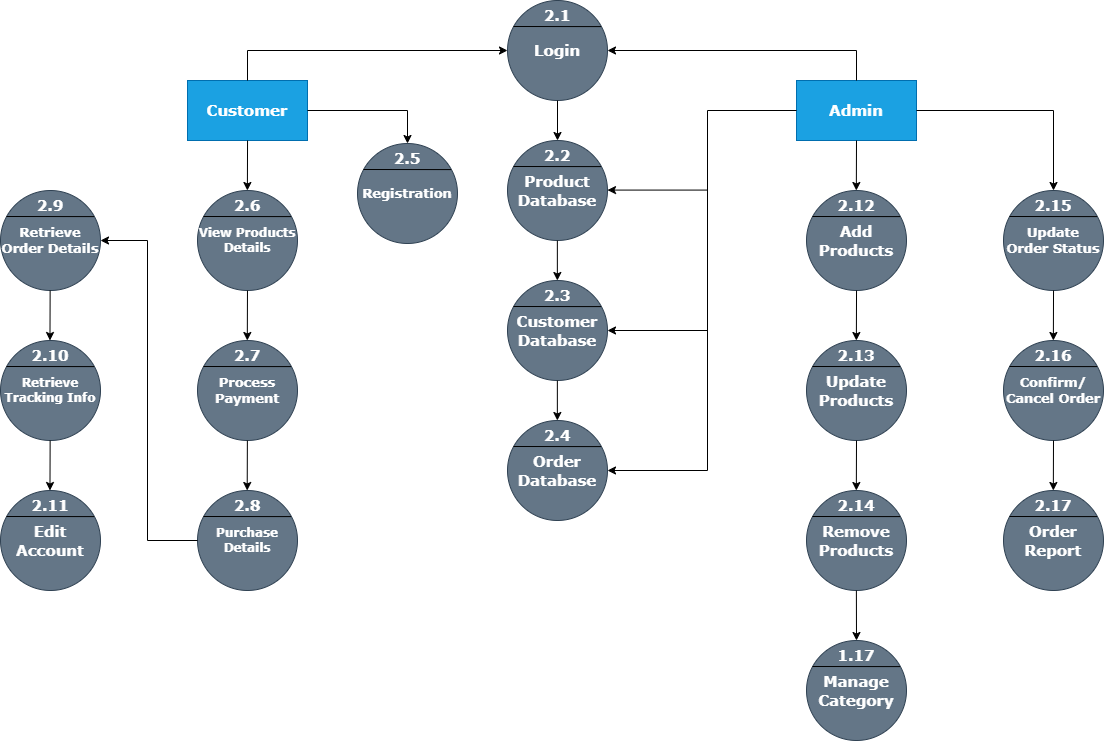
**0 Level DFD:**

****

**1 Level DFD:**

****

**2 Level DFD:**

****

**Detailed breakdown of DFDs:**

**Components of DFD:**

* **Processes:** Represent functions or transformations that are performed on data.
* **Data Flows:** Represent the flow of data between processes, data stores, and external entities.
* **Data Stores:** Represent where data is stored within the system.
* **External Entities:** Represent sources or destinations of data outside the system.

**Types of DFD:**

* **Context Diagram (Level 0 DFD):** Provides an overview of the system, showing the interactions between the system and external entities.
* **Level 1 DFD:** Breaks down the processes of the system into more detail, showing the main processes, data flows, and data stores.
* **Level 2 DFD:** Further decompose the processes into more detailed sub-processes, providing a deeper understanding of the system.

**Shapes in DFDs:**

* **Data Flow:** Represents the flow of data between processes, data stores, and external entities. Data flows are labeled with the name of the data being transferred.
* **Data Store:** Represents where data is stored within the system. Data stores are labeled with the name of the data they contain.
* **External Entity:** Represents a source or destination of data outside the system. External entities are labeled with the name of the entity.

**Uses of DFDs:**

* **System Analysis and Design:** DFDs are used during the early stages of system development to analyze and design the flow of data within a system.
* **Requirements Specification:** DFDs help in capturing and documenting system requirements by illustrating how data moves through the system.
* **Communication Tool:** DFDs serve as a communication tool between stakeholders, helping them understand the functionality and data flow of the system.
* **Identifying Opportunities for Improvement:** By visualizing the current data flow, DFDs can help identify inefficiencies or bottlenecks in the system, leading to opportunities for improvement.

- Overall, DFDs are valuable tools for understanding the flow of data within a system and are widely used in software engineering and systems analysis.

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

- A flowchart is a graphical representation of a process, showing the steps involved and the sequence in which they occur.

- It uses various symbols and arrows to illustrate the flow of control through the process.

- Flowcharts are commonly used in software development, business process analysis, and problem-solving to visually depict complex workflows and decision-making logic.

**Algorithm:**

Step 1: Start

Step 2: Declare variables num1, num2 and sum

Step 3: Read values for num1, num2

Step 4: Add num1 and num2 and assign the result to a variable sum

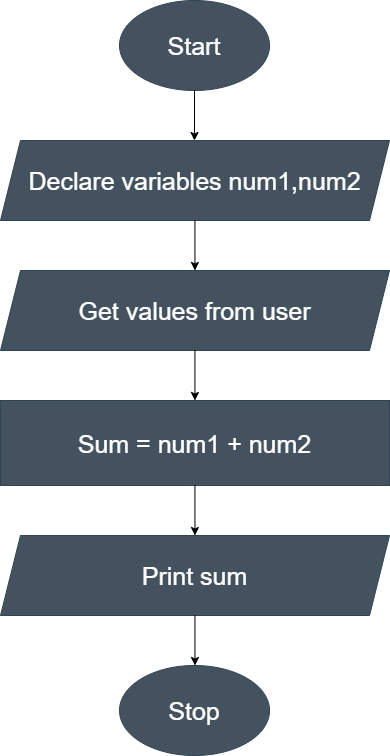
Step 5: Display sum

Step 6: Stop

**In this Flowchart:**

* Oval or Terminal symbols indicate the beginning and end of the process, respectively.
* **Parallelogram** symbols represent the input and output of data or information.
* Rectangle symbol denotes a process or operation step.
* Arrow indicates the flow between steps.

**Flowchart:**

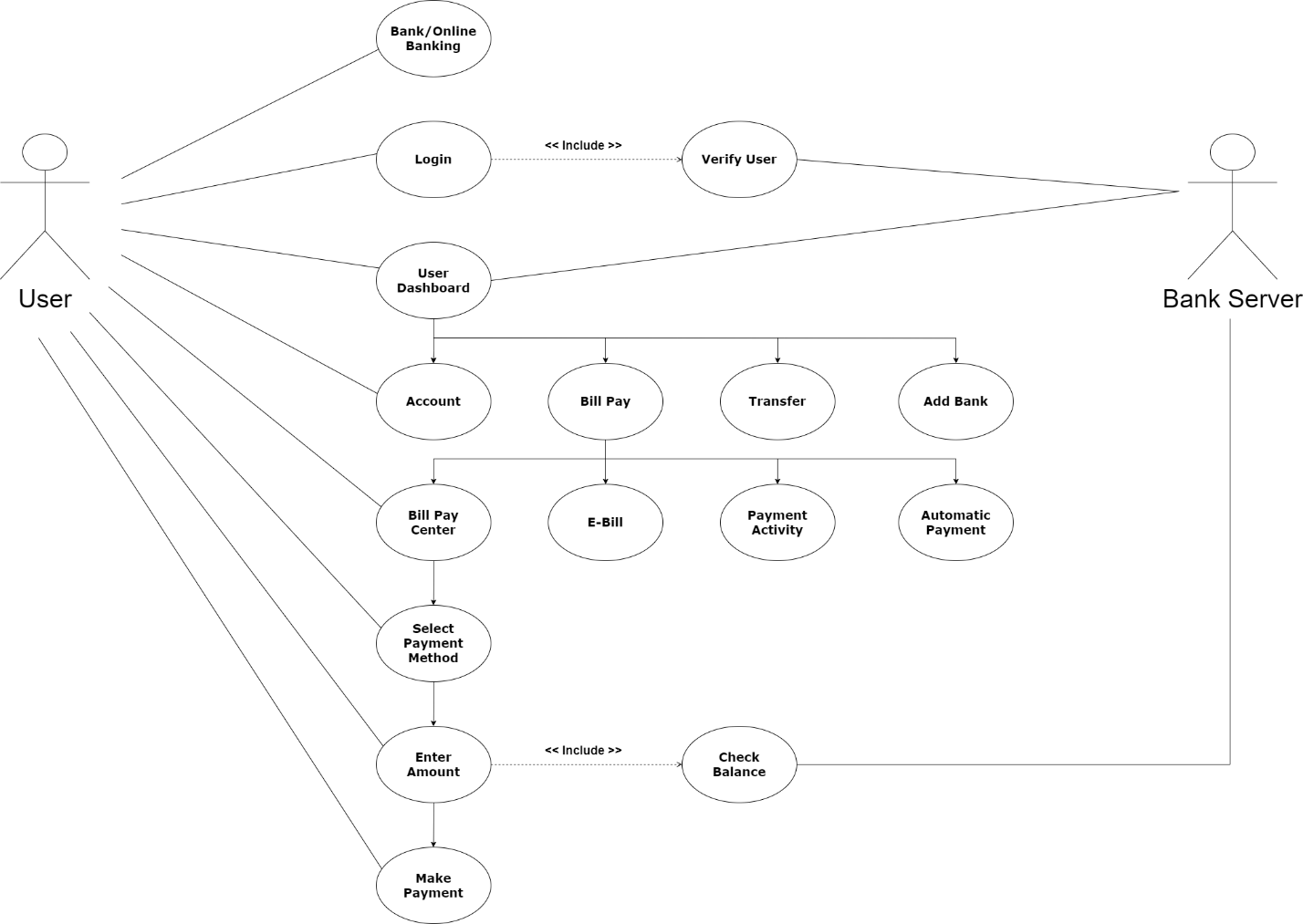


**Q.6.** **What is Use case Diagram? Create a use-case on bill payment on Paytm.**

- A Use Case Diagram is a graphical representation of the interactions between users (actors) and a system (or subsystem) to achieve specific goals.

- It helps to visualize the functional requirements of a system and the relationships between different actors and use cases.

**Use case Diagram:**



**In this diagram:**

**Actor:** An actor represents a user or an external system interacting with the system being modeled. In this case, there are two primary actors:

* **User:** The user who wants to pay a bill using Paytm.
* **Bank Server:** The entity or business that receives payments through Paytm.

**Use Cases:**

* **Login:** The user logs in to their Paytm account.
* **Select Bill Payment:** The user selects the option to pay a bill.
* **Enter Bill Details:** The user enters the details of the bill they want to pay, such as biller name, account number, and amount.
* **Verify Payment:** Paytm verifies the payment details and prompts the user for confirmation.
* **Process Payment:** Paytm processes the payment transaction and sends a confirmation to the user.

**Relationships:**

* **Association:** Represents a relationship between an actor and a use case. For example, the "User" actor interacts with various use cases such as "Login," "Select Bill Payment," etc.

- This Use Case Diagram provides an overview of the interactions between users and the Paytm system during the bill payment process, helping to understand the system's functionality from a high-level perspective.