**Module-1**

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

**Software:**

**Definition: Software refers to a set of instructions, programs, or data used to operate computers and execute specific tasks. It encompasses applications, operating systems, utilities, and other programs that enable the functioning of computer hardware.**

**Characteristics: Software is intangible, and it can be categorized into system software (e.g., operating systems) and application software (e.g., word processors, web browsers).**

**Software Engineering:**

**Definition: Software engineering is a systematic and disciplined approach to the design, development, testing, and maintenance of software. It involves applying engineering principles to the entire software development process to ensure the production of high-quality, reliable, and efficient software.**

**Objectives: The key goals of software engineering include managing the complexity of software systems, improving the quality of software products, and ensuring that software projects are completed on time and within budget.**

**Activities: Software engineering involves various activities such as requirements analysis, system design, coding, testing, deployment, and maintenance.**

**In summary, while software is the set of programs and data that enable computers to perform tasks, software engineering is the discipline that applies engineering principles to systematically develop and maintain high-quality software throughout its lifecycle.**

**2. Explain types of software**

**System Software:**

**Operating Systems (OS): Examples include Windows, macOS, Linux. They manage hardware resources and provide a platform for other software to run.**

**Device Drivers: Facilitate communication between hardware devices and the operating system.**

**Application Software:**

**Word Processors: Create and edit text documents (e.g., Microsoft Word, Google Docs).**

**Spreadsheets: Manage and analyze numerical data (e.g., Microsoft Excel, Google Sheets).**

**Web Browsers: Access and navigate the internet (e.g., Chrome, Firefox, Safari).**

**Graphics Software: Create and manipulate visual elements (e.g., Adobe Photoshop, GIMP).**

**Media Players: Play audio and video files (e.g., VLC, Windows Media Player).**

**Development Software:**

**Compilers and Interpreters: Translate programming code into machine-readable instructions.**

**Integrated Development Environments (IDEs): Provide tools for software development (e.g., Visual Studio, Eclipse).**

**Version Control Systems: Manage changes to source code (e.g., Git, SVN).**

**Utilities:**

**Antivirus Software: Protects against malware and other security threats (e.g., Norton, McAfee).**

**Backup Software: Safeguards data by creating copies for recovery (e.g., Time Machine, Acronis).**

**Compression Tools: Reduce file size for storage and transmission (e.g., WinZip, 7-Zip).**

**Middleware:**

**Database Management Systems (DBMS): Manage and organize data (e.g., MySQL, Oracle).**

**Web Servers: Facilitate communication between clients and servers (e.g., Apache, Nginx).**

**Enterprise Software:**

**Enterprise Resource Planning (ERP): Integrates business processes and data across an organization.**

**Customer Relationship Management (CRM): Manages customer interactions and relationships.**

**Embedded Software:**

**Firmware: Software embedded in hardware devices, such as routers, printers, and IoT devices.**

**Artificial Intelligence (AI) Software:**

**Machine Learning Frameworks: Tools for developing machine learning models (e.g., TensorFlow, PyTorch).**

**Natural Language Processing (NLP) Software: Processes and understands human language (e.g., SpaCy, NLTK).**

**These categories highlight the diversity of software types, each serving specific purposes within the broader realm of computing.**

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

**SDLC (Software Development Life Cycle): SDLC is a systematic process for planning, creating, testing, deploying, and maintaining software. It provides a structured approach to software development, ensuring that the end product meets user expectations and is delivered on time and within budget. The SDLC consists of several phases, each with its own set of activities and deliverables. Here are the common phases of the SDLC:**

**Requirements Gathering:**

**Objective: Identify and gather detailed requirements from stakeholders, including users, customers, and system architects.**

**Activities: Conduct interviews, surveys, and workshops to understand user needs. Document and analyze requirements to create a clear understanding of the project scope.**

**System Design:**

**Objective: Create a blueprint for the software based on the gathered requirements.**

**Activities: Design the system architecture, database structure, and user interfaces. Develop detailed technical specifications and create prototypes to visualize the system.**

**Implementation (Coding):**

**Objective: Translate the design into actual code.**

**Activities: Write code based on the specifications. Follow coding standards and best practices. This phase involves programming, unit testing, and code review to ensure quality.**

**Testing:**

**Objective: Verify that the software meets the specified requirements and is free of defects.**

**Activities: Perform various types of testing, including unit testing (testing individual components), integration testing (testing interactions between components), system testing (testing the entire system), and user acceptance testing (ensuring the software meets user expectations).**

**Deployment:**

**Objective: Release the software for use by end-users.**

**Activities: Plan and execute the deployment process. This includes installation, configuration, data migration, and any other activities necessary to make the software operational.**

**Maintenance:**

**Objective: Address issues, update features, and ensure the continued functionality of the software.**

**Activities: Regularly update the software to fix bugs, enhance features, and adapt to changes in the operating environment. Maintenance can be corrective (fixing bugs), adaptive (adjusting to changes), or perfective (adding new features).**

**These phases are often represented as a linear sequence, but in practice, they may overlap or iterate depending on the project's needs. Iterative models, such as Agile, emphasize flexibility and continuous feedback, allowing for adjustments throughout the development process.**

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

**DFD Components:**

**Processes:**

**Customer Registration/Login: This process involves customers creating accounts or logging into existing accounts.**

**Browse and Select Products: Customers browse the website, search for products, and select items to purchase.**

**Add to Cart and Checkout: Customers add selected items to their shopping cart and proceed to checkout.**

**Payment Processing: Handles the payment transaction, including verification and authorization.**

**Order Processing: Manages the processing of customer orders, updating inventory and notifying the customer.**

**Data Stores:**

**Product Database: Stores information about the products available for sale.**

**Customer Database: Contains customer information, including accounts, preferences, and order history.**

**Order Database: Stores details of customer orders, including items, quantities, and status.**

**Inventory Database: Manages the available stock of products.**

**Data Flow:**

**Customer Data Flow: Represents the flow of customer-related data between processes and the customer database.**

**Product Data Flow: Shows the movement of product-related data between processes and the product database.**

**Order Data Flow: Illustrates the flow of order-related data between processes and the order database.**

**Payment Data Flow: Represents the flow of payment-related data between processes.**

**External Entities:**

**Customer: The end-user interacting with the system by browsing products, adding items to the cart, and making purchases.**

**Payment Gateway: An external entity that handles payment processing.**

**Product Suppliers: External entities that provide information about product availability and updates to the inventory database.**

**Keep in mind that DFDs can vary in complexity, and this is a simplified representation. In practice, a comprehensive DFD for a platform like Flipkart would involve more details and potentially additional processes and data flows. You can use drawing tools or specialized software to create a more detailed and visually appealing DFD for a specific system.**

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

**A flowchart is a diagram that represents a process, illustrating the steps or activities involved and the order in which they occur. Here's a simple flowchart to depict the process of adding two numbers:**

**plaintext**

**Copy code**

**Start**

**|**

**V**

**[Enter Number A] --> (A)**

**|**

**V**

**[Enter Number B] --> (B)**

**|**

**V**

**(A) -----> [Add A and B] -----> (Result)**

**| |**

**V V**

**[Display Result] End**

**This flowchart breaks down the addition of two numbers into the following steps:**

**Start: The beginning of the process.**

**Enter Number A: User inputs the first number (A).**

**Enter Number B: User inputs the second number (B).**

**Add A and B: The two numbers (A and B) are added together.**

**Display Result: The result of the addition is displayed.**

**End: The end of the process.**

**Each rectangle represents a process or activity, and arrows indicate the flow of the process from one step to the next. The parallelogram represents input or output, and the rounded rectangle denotes the start or end of the process. In this case, the process is a simple arithmetic addition of two numbers.**

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

**Use Case Diagram:**

**A use case diagram is a visual representation that illustrates the interactions between actors (users or external systems) and a system. It identifies different use cases, which represent specific functionalities or features of the system. Here's a use case diagram for bill payment on Paytm:**

**plaintext**

**Copy code**

**---------------------------**

**| Paytm System |**

**---------------------------**

**|**

**| <<include>>**

**V**

**[Pay Bill] <---- [View Bill]**

**---------------------**

**| Actor |**

**---------------------**

**| |**

**[User] <---- [Bank]**

**Use Case Descriptions:**

**Pay Bill:**

**Description: Represents the use case where a user pays a bill using Paytm.**

**Actor: User**

**Includes: View Bill (the user needs to view the bill before making the payment).**

**View Bill:**

**Description: Represents the use case where a user views details of a bill before making a payment.**

**Actor: User**

**Bank:**

**Description: Represents an external entity (Bank) involved in the payment process.**

**Actor: Bank**

**Explanation:**

**The rectangle labeled "Paytm System" represents the entire Paytm system.**

**"Pay Bill" and "View Bill" are two use cases within the Paytm system.**

**The actor "User" interacts with both "Pay Bill" and "View Bill" use cases.**

**The "Bank" is an external entity involved in the payment process, interacting with the "Pay Bill" use case.**

**This diagram provides a high-level overview of the interactions between users and the Paytm system for bill payment. It shows the relationships between different actors and use cases, helping to understand the functionality and roles within the system.**