**1. What is software ? What are the types of software ?**

Software is a collection of computer programs that perfumes specific tasks .

**2. Types of softwares?**

1. Computer softwares —> OS, utilities .

2. Programming softwares —> compilers, debuggers.

3. Application softwares —> mobile, web, iOS , desktop.

**3. What is software testing?**

Software testing is test the developed software before the deliver to the

client or customer it should be bugs free.

**4. What is software Quality?**

\* Bug free

\* delivers on time.

\*within a budget .

\* should be satisfy the client requirements and expectations.

\* should be maintainable in the client environment.

**5.project vs product?**

A **product** in software refers to an application or system that is **developed and maintained to** fulfils the needs of users or customers. **(Based on the multiple customer or Users).**

**Eg: oracle , google etc…**

A **project** in software refers to a **temporary endeavour** undertaken to create a unique product, service, or result. **(Based on the Specific customer requirements).**

**Eg: service based companies like Tcs , Infosys etc…**

**6. Why we need to testing?**

To deliver the **flawless product(**The word **flawless** means **without any flaws or defects; perfect and impeccable**)

to the customer or client…

**7. Errors , bugs and failures?**

**Error** is nothing but its a human mistakes …**(done by the Developer )**

**Bug** is a manifestation of one or more errors in the code that causes the software to behave unexpectedly or incorrectly.**(done by the tester)**

**Failure** occurs when the software does not perform as expected under specific conditions during operation.**.(End user action).**

**8. Why the software has Bugs ?**

**Miscommunication or no communication** with the developer and tester .

**Programming errors** done by the developer .

**Lack of customer expectations or requirements.** (In the middle of the project customer

will add a some additional requirements).

**Lack of unskilled Testers .**

**Lack of wrong tase cases.**

**9. What is SDLC?**

**SDLC—> Software Development LifeCycle .**

**SDLC (Software Development Life Cycle)** is a process used by software development teams to **design, develop, test, and deploy software systems.**

**—> Requirement Analysis:**

Business analysts and stakeholders collaborate to create a **Software Requirements Specification (SRS)** document.

**—> Design:**

Based on the requirements, system architecture and design are created, including **databases, user interfaces, and system workflows.**

**—> Developer/Code :**

Programmers translate the design into source code using **programming languages and frameworks.**

**——> Test :**

It involves various types of testing like **unit testing**, **integration testing**, **system testing**, and **user acceptance testing**.

**——> Display:**

Display the **Flawless product** to the customer.

**——> Maintenance:**

Once the software is **live**, it enters the maintenance phase where it is **regularly updated, and any issues or bugs reported by users are addressed.**

**PPP:**

**P—> People.**

**P —>Process.**

**P —>Project.**

**10. WaterFall Model:**

The **Waterfall Model** is one of the earliest and most traditional software development methodologies. It follows a **linear and sequential approach**, where the development process is divided into distinct phases, and each phase must be completed before the next one begins. The model is called **"Waterfall"** because the progress flows in one direction—like a waterfall—through various stages of development.

**Key Phases of the Waterfall Model:**

**1. Requirement Gathering and Analysis:**

**-** In this initial phase, all the requirements for the software product are gathered and documented.

- The goal is to understand what the system should do and identify the needs of the stakeholders.

- Once requirements are clearly understood, they are compiled into a **Software Requirements Specification (SRS)**document.

**2. System Design:**

- Based on the requirements, the overall system architecture and design are created.

- This phase is divided into:

- **High-level design (HLD)**: Defines the system’s structure, such as modules, components, and their relationships.

- **Low-level design (LLD)**: Focuses on detailed design specifications for each module or component.

- The design serves as a blueprint for the development phase.

**3. Implementation (Coding):**

- During this phase, the design is translated into actual source code.

- Developers build each component or module as per the design.

- Once completed, the modules are integrated to form the entire system.

**4. Integration and Testing:**

- After implementation, the software undergoes testing to identify and fix defects or bugs.

- Different types of testing, like **unit testing**, **integration testing,** and **system testing**, are conducted.

- The goal is to verify that the system works as expected and meets the original requirements.

**5. Deployment:**

**-** Once the testing is complete and the software is deemed stable, it is deployed to the production environment.

- The software becomes available to end users, and the product is delivered.

**6. Maintenance:**

- After deployment, the software enters the maintenance phase.

- Any issues or bugs reported by users are fixed, and updates or enhancements may be made as needed.

- This phase ensures the software continues to operate correctly over time.

**Key Characteristics of the Waterfall Model:**

- **Linear Process**: Each phase must be completed before the next begins; there is little to no overlap between phases.

- **No Going Back**: Once a phase is completed, you cannot go back and make changes easily.

- **Documentation-Drive**n: Detailed documentation **(like SRS, design documents, etc.)** is created at each phase to guide the next phase.

- **Minimal Client Interaction During Development**: After the requirement phase, client involvement is minimal until the product is delivered, making changes difficult once development starts.

**Advantages of the Waterfall Model:**

**1. Simple and Easy to Understand:** The structured approach makes it straightforward for teams to follow, especially when requirements are clear from the beginning.

**2. Well-Documented:** Each phase is well-documented, providing clear guidelines for teams.

**3. Ideal for Small Projects:** Works best for smaller projects with well-defined requirements where changes are unlikely during development.

**4. Easy to Manage**: Since each phase has defined deliverables, it is easy to track progress and manage the project.

**Disadvantages of the Waterfall Model:**

**1. Inflexibility**: Once a phase is completed, it is difficult to go back and make changes. This makes it unsuitable for projects where requirements may evolve over time.

**2. Late Testing:** Testing is done only after the implementation phase, which may lead to discovering major issues late in the process, requiring costly rework.

**3. Limited User Feedback:** Clients do not see the product until late in the process, which can result in mismatched expectations.

**4. High Risk for Large Projects**: For large or complex projects, the Waterfall Model can become inefficient, especially if new requirements emerge during development.

**When to Use the Waterfall Model:**

- The project requirements are clearly defined and unlikely to change.

- The project is short-term and straightforward.

- The client prefers a structured process with predefined timelines.

- Ample resources are available to complete each phase without overlaps.

In summary, the **Waterfall Model** is suitable for projects where the scope is well-defined and unlikely to change, and where the structure and documentation are more important than flexibility.