Module 1 – Overview of IT Industry

1. What is a Program?

Ans: A program, in the context of computing, refers to a set of instructions that tell a computer how to perform specific tasks. These instructions are written in a programming language, such as Python, Java, or C++.

2. What is Programming?

Ans: Programming, also known as coding or software development, is the process of creating instructions that a computer can understand and execute.

3. Types of Programming Languages ?

1. Ans: Low-Level Languages
2. High-Level Languages
    - Procedural Programming Languages

    - Object-Oriented Programming Languages (OOP)
    - Functional Programming Languages
    - Scripting Languages
3. Markup and Query Languages
    - Markup Languages
    - Query Languages
4. Domain-Specific Languages (DSLs)
5. Logic Programming Languages
6. Concurrent and Parallel Programming Languages
7. Visual Programming Languages
8. Systems Programming Languages
9. Event-Driven Programming Languages

4. What are the differences between HTTP and HTTPS protocols?

- Ans: **HTTP**: It is an unsecured protocol. Data transmitted between the client (e.g., a web browser) and the server is sent in plaintext, meaning that it can be intercepted or tampered with by attackers (e.g., during a man-in-the-middle attack).
- **HTTPS**: It is a secure version of HTTP. It uses SSL/TLS encryption to protect data during transmission, ensuring that data exchanged between the client and server is encrypted and secure from interception or tampering.

- **HTTP**: No encryption; the data is sent in plaintext.

- **HTTPS**: Uses encryption protocols (SSL/TLS) to secure the data, ensuring confidentiality and integrity.

5. Explain the importance of a development environment in software production. Source Code?

Ans: A development environment plays a crucial role in software production. It provides the tools, resources, and setup necessary for software developers to write, test, and debug code efficiently. Here's why a development environment is so important:

1. Code Writing and Editing:

- A development environment typically includes **code editors** or **Integrated Development Environments (IDEs)**, which help developers write clean, error-free code.

2. Testing and Debugging:

- It provides tools to **test** and **debug** the software as it's being developed. The environment often includes built-in testing frameworks, simulators, or debuggers to identify and fix bugs early in the development cycle.

3. Version Control Integration:

- A development environment integrates with **version control systems** (e.g., Git), allowing developers to track changes to the source code, collaborate with others, and manage different versions of the project.

4. Build Automation:

- It includes tools for **automating build processes**, such as compiling code, generating executable files, and packaging software for distribution. This improves efficiency, reduces human error, and ensures consistency in the build process.

5. Code Quality and Standards:

- Many development environments incorporate tools for **static code analysis** or **linting**, which automatically check the code for style violations or potential issues (e.g., security vulnerabilities, performance bottlenecks).

6. Why are layers important in software architecture?

Ans: Layers in software architecture help organize a system by separating different concerns, making it easier to maintain, scale, and test. Each layer focuses on a specific responsibility, such as data access or business logic, allowing for modularity, flexibility, and better code management. This structure reduces complexity and makes it easier to update or replace components without affecting the entire system.

7. What are the main stages of the software development process?

Ans: The main stages of the software development process typically include:

1. **Planning**: Defining the project goals, requirements, timeline, and resources needed. This stage often involves gathering feedback from stakeholders.

2. **Analysis**: Understanding and documenting detailed system requirements, identifying potential risks, and creating a roadmap for development.
3. **Design**: Creating architectural and detailed designs for the system, including database structures, user interfaces, and component layouts.
4. **Implementation (Coding)**: Writing the actual code based on the design specifications, including development of features and integration with other components.
5. **Testing**: Verifying that the software functions correctly through various types of testing (unit, integration, system, and user acceptance tests) to ensure it meets the requirements.
6. **Deployment**: Releasing the software to the production environment for use by end users, often in stages.
7. **Maintenance**: Addressing bugs, updating features, and making improvements as needed after the software is in use.

8: How does GIT improve collaboration in a software development team?

Ans: Git improves collaboration in software development by allowing multiple developers to work independently on separate branches, then merge their changes smoothly. It tracks code changes, making it easy to review and resolve conflicts. Git's version control ensures a clear history of edits, promotes code ownership, and enhances accountability. It also integrates

with CI/CD tools to automate testing and deployment, improving workflow efficiency and code quality across the team.

9. Explain different github commands.

1. **git clone <repository-url>**: Copies a remote repository to your local machine so you can work on it.
2. **git status**: Shows the current state of your working directory, telling you which files have been modified or are staged for commit.
3. **git add <file-name>**: Stages changes to be committed. You can add specific files or all files using git add ..
4. **git commit -m "message"**: Commits staged changes to the local repository with a descriptive message.

5. **git push origin <branch>**: Uploads your local commits to the remote repository (e.g., GitHub), syncing changes.
6. **git pull origin <branch>**: Fetches and merges changes from the remote repository to your local machine, keeping it up-to-date.