Modul - 1

1 - A program is a set of instructions written in a programming language that tells the computer what to do.
The computer follows these instructions step by step to perform a specific task, like calculations, displaying output, or running games.

2 - Understand the problem

Plan the solution

Write the code

Compile/Interpret the code

Test the program

Debug errors

Document the code

Maintain the program

3 - (1) High-Level Language:

A high-level programming language is easy for humans to read and write. It uses English-like words and is designed to be simple and portable.
Example: Java, Python, C++

(2) Low-Level Language:

A low-level programming language is closer to machine code and harder for humans to read. It gives more control over hardware.
Example: Assembly language, Machine code

4 - (i) Client:

Sends request to the server.

Gets response back.

Displays the result to the user.
(Example: Your browser or mobile app.)

(ii) Server:

Receives the client's request.

Processes it (maybe uses a database).

Sends back the response.
(Example: Website server like google.com)

5 - TCP/IP model is a set of communication protocols used to connect devices on the internet. It has 4 layers that manage how data is sent and received.

(i) Application Layer – Handles user-level communication like websites and email.

(ii) Transport Layer – Ensures data is correctly sent and received (uses TCP/

6 - Client-server communication is a model where the client sends a request to the server, and the server responds with the required data or service.

7 - (i) Broadband Internet – Definition:

Broadband is a high-speed internet connection that uses copper wires like DSL or coaxial cables to transmit data over a wide bandwidth.

(ii) Fiber-Optic Internet – Definition:

Fiber-optic internet uses thin glass or plastic fibers to transmit data as light signals, providing very high speed, low latency, and better reliability.

(iii) A protocol is a set of rules that define how data is transmitted and communicated between computers over a network.

8 - (i) HTTP – Definition:

HTTP (HyperText Transfer Protocol) is a protocol used to transfer data over the web without encryption.

(ii) HTTPS – Definition:

HTTPS (HyperText Transfer Protocol Secure) is an advanced version of HTTP that uses encryption (SSL/TLS) to secure data transfer between client and server.

(iii) Application Security – Definition:

Application Security refers to protecting software and apps from threats by using security features like authentication, encryption, firewalls, and secure coding practices.

9 - Encryption – Role in Securing Applications:

Encryption is the process of converting data into a coded format so that only authorized users can read it. It protects sensitive information during storage and transmission.

Data Protection: Keeps data safe from hackers during transfer (e.g., passwords, credit card info).

Privacy: Ensures only the intended user can access the information.

Integrity: Prevents data from being changed or tampered with.

Authentication Support: Works with digital certificates (SSL/TLS) to confirm identity.

Compliance: Helps meet legal/security standards (like GDPR, HIPAA).

10 - System Software – Definition:

System software is the software that runs the computer hardware and provides a platform for other software to run.
Example: Operating System (Windows, Linux), Device Drivers.

Application Software – Definition:

Application software is the software that is designed to perform specific tasks for the user.
Example: MS Word, Chrome, Photoshop, VLC Player.

Software Architecture – Definition (Short):

Software architecture is the overall structure/design of a software system that defines how its parts work together, including components, their relationships, and communication methods.

11 - Modularity – Definition & Significance:

Modularity means breaking software into independent, smaller parts (modules) that each perform a specific function.

Significance:

Easy to Maintain – Fix one module without touching others.

Reusability – Same module can be reused in other projects.

Team Collaboration – Different teams can work on different modules.

Better Testing – Each module can be tested separately.

Improved Scalability – Easy to upgrade or add features.

12 - Importance of Layers in Software Architecture – Definition:

Layers in software architecture separate the system into different parts, each with a specific role. This improves clarity, manageability, and scalability of the application.

Why Layers Are Important:

Separation of Concerns – Each layer has its own job (UI, logic, data).

Easy Maintenance – You can change one layer without affecting others.

Code Reusability – Business logic and data access can be reused.

Better Testing – Each layer can be tested separately.

Scalability – Easier to scale or upgrade specific layers.

Team Efficiency – Teams can work in parallel on different layers.

13 - Development Environment – Definition:

A development environment is a set of tools and software (like IDEs, compilers, debuggers) that helps developers write, test, and debug code efficiently.

Importance in Software Production:

Code Writing & Editing – IDEs (like VS Code, Eclipse) help write clean code faster.

Testing & Debugging – Helps catch and fix errors before deployment.

Version Control – Works with Git to manage code changes.

Consistency – Ensures all developers work in the same setup.

Build & Compile – Converts source code to working software.

Source Code – Definition:

Source code is the human-readable code written by developers using programming languages (like Java, Python, C#), which tells the computer what to do.

14 - (i) Source Code – Definition:

Source code is the human-readable code written by a programmer in languages like C, Java, Python.

(ii) Machine Code – Definition:

Machine code is the binary (0s and 1s) version of the code that the computer's CPU can understand and execute.

(iii) GitHub – Introduction:

GitHub is an online platform used to store, manage, and collaborate on source code using Git version control.

15 - Version Control – Definition:

Version control is a system that tracks changes in source code over time, so developers can collaborate, undo mistakes, and manage multiple versions.

(i) Importance in Software Development:

Track Changes – Know who changed what and when.

Undo Mistakes – Go back to a previous working version.

Collaboration – Multiple developers can work on the same project.

Branching – Try new features without breaking the main code.

Backup – Acts as a backup of your entire codebase.

16 - GitHub Student Account – Introduction:

GitHub Student Developer Pack gives students free access to premium tools for learning and building projects.

17 - Open-Source Software: The source code is freely available. Anyone can view, use, modify, and share it.
Example: Linux, VLC Media Player

Proprietary Software: The source code is private. Only the owner or company can modify or share it.
Example: Microsoft Windows, Adobe Photoshop

18 - Git helps teams work together by tracking changes in code, so everyone knows who did what and when.
Each team member can work on their own part without messing up others' work. Git allows merging code, resolving conflicts, and reverting mistakes easily, making teamwork faster, safer, and more organized

19 - Application software helps businesses perform specific tasks like managing finances, communicating, or creating documents. It improves productivity by automating work, organizing data, and making processes faster and easier.

Role of Application Software in Businesses – Definition:

Application software helps businesses perform specific tasks like managing sales, accounting, communication, customer service, etc.

Software Development Process – Definition:

The software development process is a step-by-step method used to design, build, test, and deliver software.

20 - Main Stages of Software Development Process:

Requirement Analysis
(1)Collect and understand what the software should do.

Design
(2)Plan the structure, UI, and flow of the software.

Implementation (Coding)
(3) Developers write the actual source code.

Testing

(4)Find and fix bugs, ensure software works correctly.

Deployment

(5) Launch the software for end users.

Maintenance

(6) Fix issues, update features after release.

Software Requirement – Definition:

Software requirements are the detailed needs and expectations that the software must fulfill for users and the business.

21 - Why Requirement Analysis is Critical:

Better Planning – Time, budget, and resources allocate.

Customer Satisfaction – End-product exactly customer.

Foundation for Design & Coding – Design and development

Software Analysis – Definition:

Software analysis is the process of studying and breaking down the system requirements to understand what the software must do.

22 - Software Analysis – Definition:

Software analysis is the process of studying user requirements to clearly understand what the software must do before development starts.

Role of Software Analysis in Development:

Understand Needs – Helps developers know exactly what the user wants.

Problem Identification – Detects issues or gaps early.

Creates Base for Design – Gives a clear guide for system design.

Avoids Rework – Reduces chances of rewriting code later.

Improves Quality – Leads to better planning and cleaner development.

System Design – Definition:

System design is the process of planning the structure and components of the software system, including UI, data flow, and architecture.

23 - System Design – Definition:

System design is the process of defining the structure, components, modules, and flow of a software system

24 - Software Testing – Definition:

(i) Software testing is the process of checking a software system to ensure it is working correctly, free from bugs, and meets user requirements.

(ii) Why is Software Testing Important?

Software testing is important because it:

Detects bugs early before software is released.

Ensures the software works correctly as per requirements.

Improves performance and system stability.

Increases user confidence and satisfaction.

Saves cost and time by reducing major issues later.

(iii) Maintenance – Definition:

Maintenance is the process of updating, fixing, and improving software after it is deployed.

25 - Software Maintenance – Definition:

Software maintenance is the process of modifying and updating software after delivery to fix bugs, improve performance, or adapt to new environments.

Types of Software Maintenance:

Corrective Maintenance
? Fixing bugs or errors found after the software is released.

Adaptive Maintenance

? Updating the software to work with new hardware, OS, or tools.

Perfective Maintenance
? Enhancing existing features or adding new ones to improve performance.

Preventive Maintenance
? Making changes to prevent future problems, like optimizing code.

Development – Definition:

Software development is the process of designing, coding, testing, and deploying software to fulfill specific user or business needs.

26 - Web Application

- Runs directly in your browser — no need to install anything.
- Requires internet connection to function (most of the time).
- Accessible from any device that has a browser — mobile, tablet, PC, etc.
- Stores data online (cloud-based storage).
- Examples: Gmail, Google Docs, Canva.

Desktop Application

- Needs to be downloaded and installed on your computer.
- Often works offline, no internet required after installation.
- Works only on the device where it's installed.
- Saves files locally on your hard drive or SSD.
- Examples: Microsoft Word, Adobe Photoshop, VLC Media Player.

27 - Advantages of Web Applications over Desktop Applications:

Accessibility – Can be accessed from anywhere using a browser.

No Installation Needed – Runs directly on the web; no setup required.

Easy Updates – Changes made on the server reflect instantly for all users.

Cross-Platform – Works on any OS (Windows, Mac, Linux) with a browser.

Cost-Effective – Easier to maintain and scale compared to desktop apps.

Collaboration – Supports multi-user access and real-time sharing.

28 - UI/UX Design – Definition:

UI (User Interface) design focuses on the look and feel of the application.

UX (User Experience) design focuses on the overall user journey and interaction with the app.

Role of UI/UX in Application Development:

Enhances Usability – Makes the app easy and intuitive to use.

Improves User Satisfaction – Users enjoy the experience, so they return.

Boosts Engagement – Attractive design keeps users involved.

Reduces Errors – Clear layout helps users avoid mistakes.

Strengthens Branding – A polished design gives a professional image.

Increases Conversions – Better UX leads to more signups, purchases, etc.

29 - Native App – Definition:

(i) A native app is built specifically for one platform (like Android or iOS) using that platform's official language and tools.

Android ? Java/Kotlin

iOS ? Swift/Objective-C

(ii) Hybrid App – Definition:

A hybrid app is built using web technologies (like HTML, CSS, JavaScript) and runs inside a native container, making it work on multiple platforms.

30 - DFD (Data Flow Diagram) – Definition:

A Data Flow Diagram (DFD) is a graphical tool used to show how data moves through a system — from input to output.

Significance of DFDs in System Analysis:

Visual Representation – Makes it easy to understand the system flow.

Clarifies Requirements – Shows how processes and data interact.

Detects Errors Early – Helps find issues in data handling.

Improves Communication – Clear for developers, analysts, and clients.

Supports System Design – Acts as a foundation for building the actual system.

31 - (1)Desktop Applications

(i) Pros:

High Performance – Runs faster using system resources directly.

Works Offline – No internet needed after installation.

Full Hardware Access – Better access to system features (printer, USB, etc.).

(ii) Cons:

Platform Dependent – Needs separate versions for Windows, Mac, etc.

Manual Updates – User must download/install updates.

Limited Accessibility – Only usable on the installed device.

(2)Web Applications

(i) Pros:

Accessible Anywhere – Just need a browser & internet.

No Installation – Runs without setup.

Easy to Update – Updates are made on the server for all users.

(ii) Cons:

Needs Internet – Mostly can’t work offline.

Lower Speed – Slower than desktop apps for heavy tasks.

Limited Device Access – Cannot fully use local hardware features.

32- Flowchart – Definition:

A flowchart is a visual diagram that shows the steps of a process or program using symbols like arrows, boxes, and decisions.

(i) How Flowcharts Help in Programming & System Design:

Visual Clarity – Shows the logic of a program clearly.

Easy to Understand – Non-technical people can follow the flow.

Helps in Planning – Before writing code, you can map out the logic.

Identifies Errors – Easy to spot missing steps or wrong logic.

Improves Communication – Useful for teams to understand the system flow.

Saves Time – Better planning leads to fewer bugs and rewrites.