Module 1

1. What is a Program?

LAB EXERCISE: Write a simple "Hello World" program in two different languages.

- **Python:** print("Hello World")
- **JavaScript:** console.log("Hello World");
- **Comparison:** Python is concise and doesn't require semicolons, while JavaScript is structured for web consoles.

THEORY EXERCISE: Explain in your own words what a program is and how it functions.

A program is a sequence of instructions written to perform a specific task with a computer. It functions by taking input, processing it through the CPU, and giving an output.

2. What is Programming?

THEORY EXERCISE: Key steps in the programming process:

1. Problem Definition/Requirement Analysis.
2. Planning and Logic Design.
3. Coding (Writing Source Code).
4. Testing and Debugging.
5. Deployment and Maintenance.

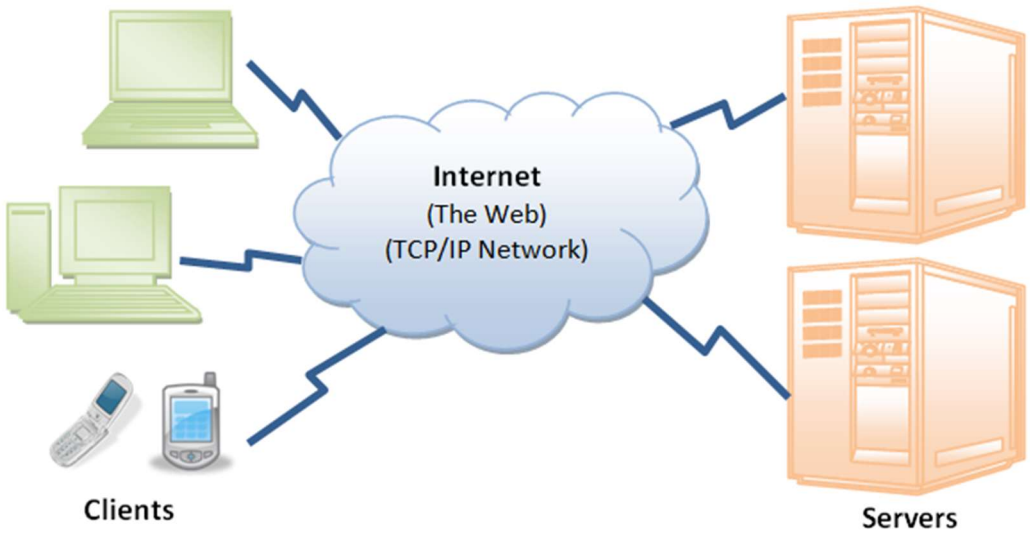
3. Types of Programming Languages

Differences between High-level and Low-level languages:

High-Level Programming Languages	Low-Level Programming Languages
Easy to understand and use by humans	Closer to machine and hardware
High, English-like syntax	Low, uses symbols and binary
Machine-independent	Machine-dependent
Highly portable	Not portable
Java, Python, C++	Machine language, Assembly language

4. World Wide Web & How Internet Works

LAB EXERCISE: Diagram of data transmission.



THEORY EXERCISE: Roles in web communication:

- **Client:** The device/browser that sends a request for information.
- **Server:** The computer that stores the data and responds to the client's request.

5. Network Layers on Client and Server

LAB EXERCISE:
Simple HTTP Client–Server Communication

Client (Browser / HTTP Client):

Sends an HTTP request (GET/POST) to the server.

Request contains URL, headers, and optional data.

Server (Web Server):

Receives the request.

Processes it.

Sends an HTTP response (status code + data).

Flow:
Client → HTTP Request → Server → HTTP Response → Client

(Example: When a browser requests a webpage, the server sends HTML as a response.)

THEORY EXERCISE:

TCP/IP Model and Its Layer
The TCP/IP model is a standard communication model used for data transmission over the internet.

Layers of TCP/IP Model

Application Layer

Provides services to user applications.

Examples: HTTP, FTP, SMTP

Transport Layer

Ensures reliable data transfer.

Examples: TCP, UDP

Internet Layer

Handles logical addressing and routing.

Example: IP

Network Access Layer

Transmits data over physical network.

Examples: Ethernet, Wi-Fi

Function:
The TCP/IP model enables reliable, structured, and efficient communication between client and server systems.

6. Client and Servers

THEORY EXERCISE: **Explain Client–Server Communication**

Client–server communication is a network model where a client requests services or data from a server.
The client sends a request through the internet using protocols like HTTP or HTTPS.
The server receives the request, processes it, accesses the required data, and sends a response back to the client.
This model allows multiple clients to access services from a centralized server efficiently.
It is widely used in web applications, email services, and online databases.

7. **Types of Internet Connections**

LABEXERCISE: Pros and Cons:

- **Fiber:** (Pro) Fastest speed, (Con) High cost.
- **Broadband:** (Pro) Widely available, (Con) Slower than fiber.
- **Satellite:** (Pro) Accessible in remote areas, (Con) High latency.

THEORY EXERCISE: How does broadband differ from fiber-optic internet?

Broadband often uses copper/cable, whereas fiber-optic uses pulses of light over glass threads

For much higher speeds.

8. Protocols

LAB EXERCISE:

HTTP Request Simulation

curl is a command line tool used to send HTTP requests to a web server.

It sends an HTTP GET request to the specified URL.

Example command: curl http://example.com

The server processes the request and returns a response.The response is displayed directly on the command line.

FTP Request Simulation

FTP is used to transfer files between a client and a server.

curl can be used to connect to an FTP server.

Example command: curl ftp://ftp.example.com

It allows access to files or directories stored on the FTP server.

The server sends the requested data to the client.

HTTP	**HTTPS**
Stands for HyperText Transfer Protocol	Stands for HyperText Transfer Protocol Secure
Data is sent in plain text	Data is encrypted
Less secure	More secure
Uses port number 80	Uses port number 443
Does not use SSL/TLS	Uses SSL/TLS encryption

9. Application Security

LAB EXERCISE:
Common Application Security Vulnerabilities and Solutions
1. SQL Injection

Vulnerability: Attackers insert malicious SQL queries through input fields.

Risk: Unauthorized access to database data.

Solution: Use prepared statements and input validation.

2. Cross-Site Scripting (XSS)

Vulnerability: Malicious scripts are injected into web pages.

Risk: Steals user data like cookies and session details.

Solution: Validate and sanitize user input and use secure coding practices
3. Weak Authentication

Vulnerability: Use of weak passwords or poor login mechanisms.

Risk: Unauthorized user access.

Solution: Use strong passwords, multi-factor authentication, and secure login policies.

THEORY EXERCISE: Role of Encryption in Securing Applications

Encryption converts data into unreadable form to protect it from unauthorized access.

It secures data during transmission and storage.

Encryption helps maintain data confidentiality and privacy.

It protects sensitive information like passwords and financial data.

It ensures secure communication between client and server..

10. Software Applications and Types

LAB EXERCISE: identify and classify 5 applications you use daily as either system software or application software.

Five applications I use daily can be classified as follows:

Google Chrome – This is an application software because it is used to browse the internet.

Microsoft Word – This is an application software because it helps in creating and editing documents.

VLC Media Player – This is an application software as it is used to play audio and video files.

Windows 10/11 – This is system software because it is the operating system that manages computer hardware and provides a platform for other software to run.

Antivirus (e.g., Windows Defender) – This is system software because it protects the computer

from viruses and malware, running in the background to maintain system security.

So, system software examples are Windows OS and antivirus, while application software examples are Chrome, Word, and VLC.

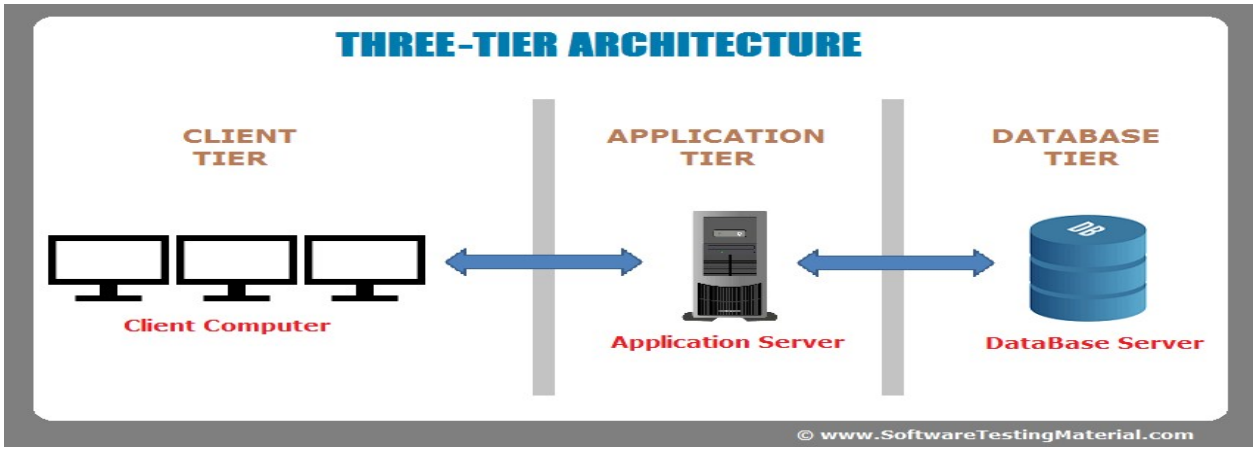
THEORY EXERCISE

Difference between System Software and Application Software:

System Software	Application Software
Software designed to manage and control computer hardware and provide a platform for running applications.	Software designed to perform specific tasks for users.
To run the computer and its hardware efficiently.	To help users complete tasks like writing, browsing, or entertainment.
Operating systems (Windows, Linux), Utility software (antivirus, disk cleanup)	MS Word, Google Chrome, Photoshop, VLC Media Player
Works independently and provides a base for applications.	Depends on system software to run.

11. Software Architecture

LAB EXERCISE: Three-tier architecture.



THEORY EXERCISE: What is the significance of modularity in software architecture?

Modularity is the design principle of dividing a software system into independent, self-contained modules. It helps in easier development and maintenance, as changes in one module do not affect others. Modules can be tested and debugged separately, improving reliability. It promotes code reusability, allows parallel development by different teams, and makes the system scalable and flexible for future enhancements. Overall, modularity improves readability, manageability, and efficiency of software systems.

12. Layers in Software Architecture

LAB EXERCISE: Create a case study on the functionality of the presentation, business logic, and data

Case Study: Library Management System (LMS)

Objective:

To understand the functionality of the Presentation Layer, Business Logic Layer, and Data Access Layer in a software system.

1. Presentation Layer (UI Layer):

Function: Provides an interface for users to interact with the system.

Example in LMS:

Screens where librarians can add books, issue books, or check member details.

Members can search books, view due dates, or reserve books.

Key Role:

Captures user inputs and displays information returned from the system.

Ensures a user-friendly experience without exposing system internals.

Logic Layer (Application Layer):

Function: Implements the core functionality and rules of the system.

Example in LMS:

Issue Book Rule: A member cannot borrow more than 5 books at a time.

Return Book Rule: Late returns incur fines.

Search Logic: Efficiently finds books based on title, author, or category.

Key Role:

Processes input from the presentation layer, applies business rules, and decides what data

operations are needed.

Acts as a bridge between the UI and data storage.

3. Data Access Layer (DAL / Database Layer):

Function: Handles storage, retrieval, and update of data from databases.

Example in LMS:

Adding a new book record to the database.

Fetching member details for verification.

Updating issued/returned book status in the database.

Key Role:

Encapsulates database operations so that other layers don't interact directly with the database.

Ensures data integrity, security, and efficient data handling.

Flow of Operation (Example):

A librarian enters a new book in the UI (Presentation Layer).

The system checks rules (e.g., duplicate ISBN) in the Business Logic Layer.

The book information is stored in the Database (Data Access Layer).

Confirmation is sent back through BLL to UI, and the librarian sees a success message.

Conclusion:

The layered architecture ensures separation of concerns, making the system maintainable, scalable, and easier to debug. Each layer has a clear responsibility: UI for interaction, BLL for rules, and DAL for data management.

Theory Exercise:

Why layers are important in software architecture:

Layers in software architecture are important because they help organize the system into separate levels, where each layer has a specific role. This separation makes the software easier to develop, maintain, and understand. For example, one layer can handle user interface, another layer can handle business logic, and another can manage data storage.

Layers also improve modularity, meaning changes in one layer (like changing the database) usually do not affect other layers. They enhance reusability, scalability, and security because each layer can be designed independently with clear responsibilities.

13. Software Environments

- **LAB EXERCISE:**

  Types of Software Environments

  Development Environment: Where developers write and test code using IDEs.

  Testing Environment: Where software is tested for bugs and performance before release.

  Production Environment: The live environment used by end-users; must be stable and secure.

  Flow: Development → Testing → Production ensures software works correctly.

  Setting Up a Basic Environment in a Virtual Machine

  Install VirtualBox or VMware.

  Download an OS ISO (e.g., Ubuntu/Windows).

  Create a new VM, allocate CPU, RAM, and storage, and install the OS.

  Install development tools (IDEs, compilers) or testing tools inside the VM.

  Take a snapshot to save the environment.

  This lets you develop and test safely without affecting your main system.

●THEORY EXERCISE:

Importance of a Development Environment in Software Production:

A development environment is crucial because it provides a controlled space where developers can write, test, and debug code safely. It ensures that software is built efficiently without affecting the live system. Using a proper development environment helps in identifying errors early, maintaining code quality, and making collaboration easier among developers. It also allows testing new features and configurations before they go to production, reducing the risk of failures in the live application.

14. Source Code

LAB EXERCISE: Uploading Source Code to GitHub

Step	Description
1	Write source code: Create a file like hello.py with print("Hello, World!").

2 Create GitHub repository: Go to GitHub → New Repository → Give a name → Create.

3 Upload source code:

• Web interface: Upload file → Commit changes.

• Git commands: Initialize repo, add file, commit, set remote, push.

4 Verify: Check your repository on GitHub to see the uploaded code.

THEORY EXERCISE: Source Code vs Machine Code

Source Code	Machine Code
Code written by programmers in a high-level language.	Binary code (0s and 1s) that the computer executes directly.
Human-readable	Not human-readable
print("Hello, World!")	10101000 11000011 ...
To create software in a readable format	**To execute instructions on a computer**

15. Github and Introductions

LAB EXERCISE: Create a GitHub Repository & Commit/Push Code

Steps to Create Repository and Push Code
Create GitHub Account
Go to github.com
Sign up using email → verify → login
Create New Repository
Click "+" → New Repository
Enter repository name
Choose Public / Private
Click Create Repository
Initialize Git in Project Folder
git init
Add Files to Staging Area

git add .
Commit Changes
git commit -m "Initial commit"
Connect Local Repo to GitHub
git remote add origin repository_URL
Push Code to GitHub
git push -u origin main

THEORY EXERCISE: Why is Version Control Important in Software Development?

Version control is important because:

- It tracks changes made to source code over time
- Allows multiple developers to work together safely
- Helps in restoring previous versions if errors occur
- Maintains a history of modifications
- Prevents code loss and conflicts
- Improves project management and collaboration

16.Student account in github

LAB EXERCISE: Create a Student Account on GitHub & Collaborate on a Project.

Steps to Create a Student Account

1. Visit github.com and sign up.

2. Login → Go to Settings.

3. Click Education → Student Developer Pack.

4. Verify student status using:

College email or Student ID card

5. After verification, student benefits are activated.

Steps to Collaborate with a Classmate

1. Create a Repository

Click New Repository

Select Public

Add README file

2. Add Classmate as Collaborator

Go to Repository → Settings → Collaborators

Enter classmate's GitHub username

Send invitation

3. Classmate Accepts Invitation

Classmate logs in and accepts invite

4. Both Students Work on Project

One student pushes code

Other student pulls, edits, and commits changes

Changes are shared using GitHub

THEORY EXERCISE: Benefits of Using GitHub for Students

- Free access to GitHub Student Developer Pack
- Helps learn version control and Git
- Encourages team collaboration
- Builds real project portfolio
- Useful for internships and placements
- Improves coding and project management skills
- Supports open-source learning

17.Types of software

LAB EXERCISE: List of Software Used Regularly and Their Classification

- System Software
    1. Windows 10
    2. Linux
- Application Software
    1. MS Word
    2. Google Chrome
    3. VLC Media Player
- Utility software
    1. Antivirus software
    2. Disk Cleanup

THEORY EXERCISE: **Differences Between Open-Source and Proprietary Software**

Open-Source Software	Proprietary Software
source code is freely available	Source code is not available
Users can modify the software	Users cannot modify the software
Mostly free of cost	Generally paid
Example: Linux	Example: Windows

18. GIT and GITHUB Training

LAB EXERCISE: Practice Cloning, Branching, and Merging Using GIT

Cloning:

A repository is copied from GitHub to the local system using git clone.

Branching:

A new branch is created to work on a feature separately from the main code.

Changes can be made without affecting the main branch.

Merging:

After work is completed, the branch is merged into the main branch.

This combines changes from different branches into one codebase.

THEORY EXERCISE: How Does GIT Improve Collaboration in a Software Development Team?

- Allows multiple developers to work on the same project simultaneously
- Maintains complete history of code changes
- Helps manage and resolve code conflicts
- Enables safe experimentation using branches
- Improves coordination and teamwork

19.Application Software

LAB EXERCISE: **Report on Types of Application Software and How They Improve Productivity**

1. Application software refers to programs designed to help users perform specific tasks such as writing documents, managing data, communication, designing, or business operations. These software tools make work faster, easier, and more accurate, thereby improving productivity.
2. One common type is Word Processing Software like MS Word or Google Docs. It helps users create, edit, and format documents quickly, reducing paperwork and saving time.
3. Spreadsheet Software such as MS Excel is used for calculations, data analysis, and record keeping. It increases productivity by automating calculations and organizing large amounts of data efficiently.
4. Presentation Software like PowerPoint helps in creating visual presentations. It improves productivity by allowing users to communicate ideas clearly and professionally in less time.
5. Database Management Software such as MySQL or MS Access is used to store and manage large volumes of data. It improves productivity by providing fast data retrieval, accuracy, and security.
6. Communication Software like email clients, Zoom, or messaging apps enables quick communication and collaboration, saving time and improving teamwork.
7. Graphics and Design Software like Photoshop or Canva helps in designing images and marketing materials efficiently, reducing manual effort.

Overall, application software automates tasks, minimizes errors, saves time, and helps users work more efficiently.

THEORY EXERCISE: Role of Application Software in Businesses

Application software plays a crucial role in businesses by supporting daily operations and decision-making. It helps businesses manage data, communicate effectively, and perform tasks efficiently.

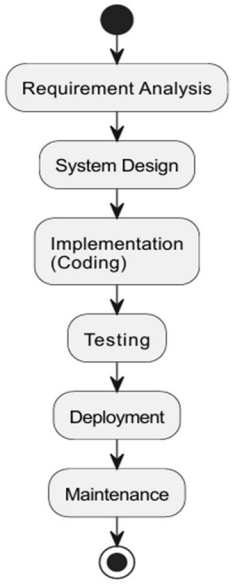
Businesses use application software for accounting, payroll, inventory management, customer relationship management, and reporting. This reduces manual work and improves accuracy.

Application software also helps in improving productivity, reducing costs, enhancing communication, and providing better customer service. It enables faster decision-making through data analysis and reporting tools.

In short, application software is essential for smooth business operations, growth, and competitiveness.

20.Software development process

LAB EXERCISE: Create a flowchart representing the Software Development Life Cycle (SDLC).



THEORY EXERCISE: What are the main stages of the software development process?

1. Requirement Analysis:

This is the first stage of the software development process. In this stage, user needs are collected and analyzed to understand what the software should do.

2. System Design:

In this stage, the overall structure of the software is planned. It includes system architecture, database design, and user interface design.

3. Implementation (Coding):

Here, developers write the actual program code based on the system design. The software is developed using suitable programming languages.

4. Testing:

This stage is used to identify and remove errors in the software. It ensures that the software works correctly and meets the requirements.

5. Deployment:

After successful testing, the software is released and installed for real-world use by users.

6. Maintenance:

This is the final stage where updates, bug fixes, and improvements are done to keep the software running smoothly.

21.Software Requirement

LAB EXERCISE: Requirement Specification for a Simple Library Management System

Introduction

The Library Management System (LMS) is designed to automate and manage library activities such as book records, member information, issuing and returning books, and fine calculation. The system aims to reduce manual work and improve efficiency.

Objectives

The main objective of the system is to maintain accurate records of books and members, provide easy access to information, and support smooth library operations.

Functional Requirements

- The system should allow the librarian to add, update, search, and delete book records.
- The system should store details of library members such as name, ID, and contact information.The system should allow issuing and returning of books.
- The system should automatically calculate fines for late returns.
- The system should generate simple reports such as available books and issued books.
- The system should provide login access for the librarian.

Non-Functional Requirements

- The system should be easy to use and user-friendly.
- The system should provide fast response time for searching records.
- The system should ensure data security and authorized access.
- The system should be reliable and available during library working hours.

Constraints

The system will be used only within the library.

It will be developed using basic software tools and technologies.

Only the librarian will have full access to the system.

THEORY EXERCISE: Why is the Requirement Analysis Phase Critical in Software Development?

- The requirement analysis phase is critical because it clearly defines what the software should do. It helps developers understand user needs and expectations before development

begins.

- Proper requirement analysis reduces misunderstandings between users and developers. It helps in identifying system scope, functionalities, and constraints at an early stage.
- This phase also helps in reducing errors, rework, and development cost. A well-defined requirement analysis ensures that the final software meets user requirements and achieves its intended purpose.

22. Software Analysis

LAB EXERCISE: Functional Analysis of an Online Shopping System

Functional analysis identifies what functions the system must perform to meet user needs. An online shopping system provides facilities for customers to browse products, place orders, and make payments, while allowing administrators to manage products and orders.

User Functions

- The system allows users to register and log in securely.
- Users can browse products by category and search for specific items.
- Users can view product details such as price, description, and availability.
- Users can add products to the shopping cart and modify cart items.
- Users can place orders and make online payments.
- Users can view order history and track order status.

Admin Functions

- The system allows the administrator to add, update, and delete product details.
- The administrator can manage product categories and pricing.
- The administrator can view and process customer orders.
- The administrator can manage user accounts and handle order issues.

System Functions

- The system validates user input and login credentials.
- The system calculates the total cost including taxes and discounts.
- The system updates product stock after order placement.
- The system sends order confirmation notifications to users.

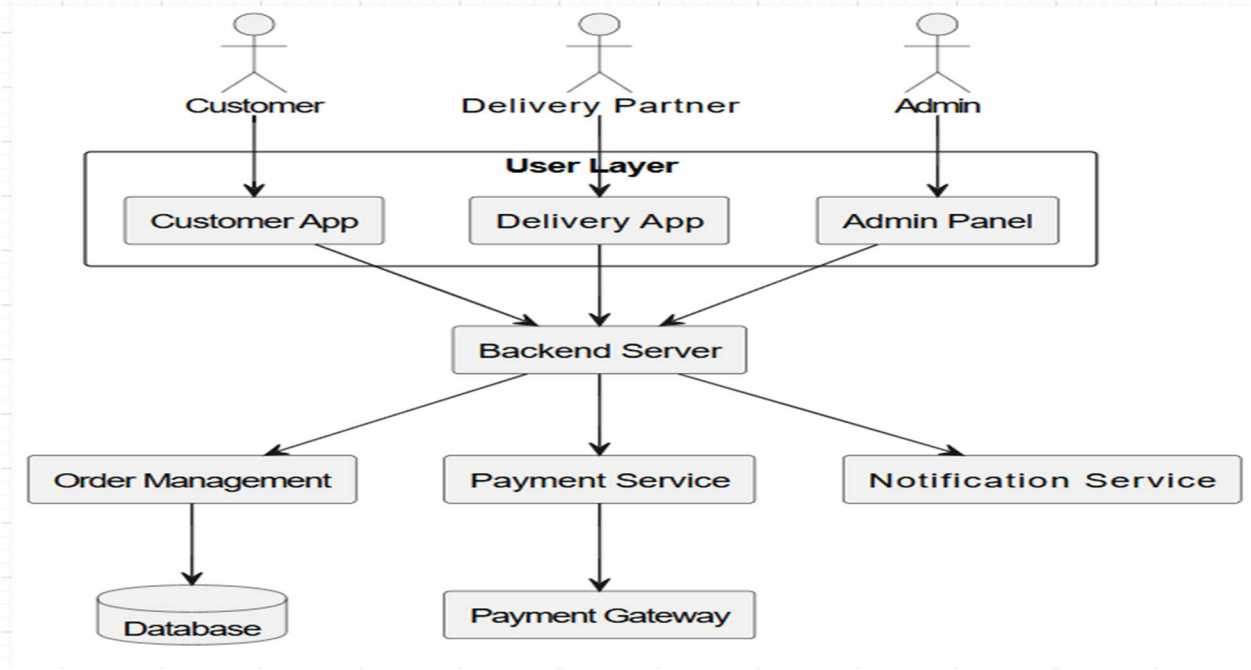
THEORY EXERCISE: Role of Software Analysis in the Development Process

- Software analysis plays an important role in the development process by clearly defining system requirements and functionalities before design and coding begin.
- It helps in understanding user needs, identifying system scope, and analyzing feasibility. Software analysis ensures that all requirements are documented clearly and correctly.

- This phase reduces risks, prevents errors, and avoids unnecessary changes during later stages of development. It also helps developers create efficient and reliable software that meets user expectations.

23.System Design

LAB EXERCISE: Design a basic system architecture for a food delivery app.



THEORY EXERCISE: What are the key elements of system design?

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. The key elements are:

- **Architecture Design:**
  Defines the overall structure of the system, including major components, their relationships, and interaction.

- **Data Design:**
  Focuses on designing the database, data structures, and how data flows within the system.

- **Interface Design:**
  Specifies how different modules and components communicate with each other, and how users interact with the system (UI/UX).

- **Component/Module Design:**
  Breaks the system into smaller modules or components and defines their functionalities.

- **Security and Performance Considerations:**
  Ensures the system is secure, reliable, efficient, and scalable.

- **Documentation:**
  Provides clear specifications for developers, testers, and users to understand the system design.

24. Software Testing

LAB EXERCISE: Develop test cases for a simple calculator program.

Test caseno	Input1	Input 2	Operation	Expected Output	Result
1	5	3	+	8	
2	10	4	-	6	
3	6	7	*	42	
4	20	5	/	4	
5	10	0	/	Error	
6	-3	7	+	4	
7	0	0	+	0	

THEORY EXERCISE: Why is Software Testing Important?

- Software testing is important because it ensures that the software works correctly and meets user requirements.
- It helps detect errors and bugs before deployment.
- Improves software quality, reliability, and performance.
- Reduces cost of fixing problems after release.
- Ensures the software is safe, secure, and user-friendly.

- Builds confidence for developers and users that the system works as expected**.**

25. Maintenance

LAB EXERCISE: Real-World Case of Critical Software Maintenance

Case Study: Windows 10 Security Update (2017)

Software: Microsoft Windows 10

Problem: In 2017, a critical security vulnerability named "EternalBlue" was discovered, which allowed malware like WannaCry ransomware to spread rapidly through networks.

Maintenance Required: Microsoft released an emergency security patch to fix the vulnerability.

Impact of Maintenance:

- Prevented further attacks and data loss
- Ensured system stability and security
- Maintained user trust and business continuity

Conclusion: Critical maintenance is necessary to fix serious bugs, security issues, or system failures that can affect millions of users.

THEORY EXERCISE: Types of Software Maintenance

Software maintenance refers to modifying and updating software after delivery to fix defects, improve performance, or adapt to changes.

1. Corrective Maintenance

Fixes bugs, errors, or defects found after software deployment.

Example: Fixing a login issue or crash bug.

2. Adaptive Maintenance

Modifies software to adapt to new environments or OS updates.

Example: Updating an app to run on the latest version of Android.

3. Perfective Maintenance

Improves performance, efficiency, or usability without changing functionality.

Example: Enhancing the UI of a website or optimizing database queries.

4. Preventive Maintenance

Makes proactive changes to prevent future problems.

Example: Refactoring code, updating libraries, or improving security measures.

26.Development

THEORY EXERCISE: Key Differences Between Web and Desktop Applications

Web Application	**Desktop Application**
Accessed via web browser over the internet or intranet	Installed and run directly on a computer
No installation required; runs on any device with a browser	Must be installed on each device individually
Generally platform-independent	Often platform-dependent (Windows, macOS, Linux)
Updates are done on the server; users automatically get latest version	Users need to manually install updates or patches
Gmail, Facebook, Google Docs	MS Word, Photoshop, VLC Media Player

27.Web Application

THEORY EXERCISE: What are the advantages of using web applications over desktop applications?

- Accessibility Anywhere

Can be accessed from any device with a web browser and internet connection.

- No Installation Needed

Users don't need to install software on their computers.

- Automatic Updates

Updates are applied on the server, so all users get the latest version automatically.

- Platform Independence

Works on multiple operating systems (Windows, macOS, Linux, Android, iOS) without extra Effort.

- Cost-Effective

Lower maintenance cost because only the server is updated, not each individual computer.

- Easy Collaboration

Multiple users can work together in real-time (e.g., Google Docs).

- Reduced Storage Requirement

Most of the processing and storage is done on the server, saving local disk space.

28. Designing

THEORY EXERCISE: What role does UI/UX design play in application development?

UI (User Interface) / UX (User Experience) design plays a crucial role in making software usable, efficient, and enjoyable for users.

Enhances Usability

A good UI/UX ensures that users can easily navigate the application and perform tasks without confusion.

Improves User Satisfaction

Intuitive and attractive design keeps users engaged and satisfied.

Reduces Errors

Clear layouts, instructions, and feedback help prevent mistakes while using the application.

Supports Efficiency

Well-designed workflows and interactions save time and improve productivity.

Increases Adoption

Applications with excellent UI/UX are more likely to be used and recommended by users.

Competitive Advantage

Good UI/UX design makes an application stand out from competitors in the market.

29. Mobile Application

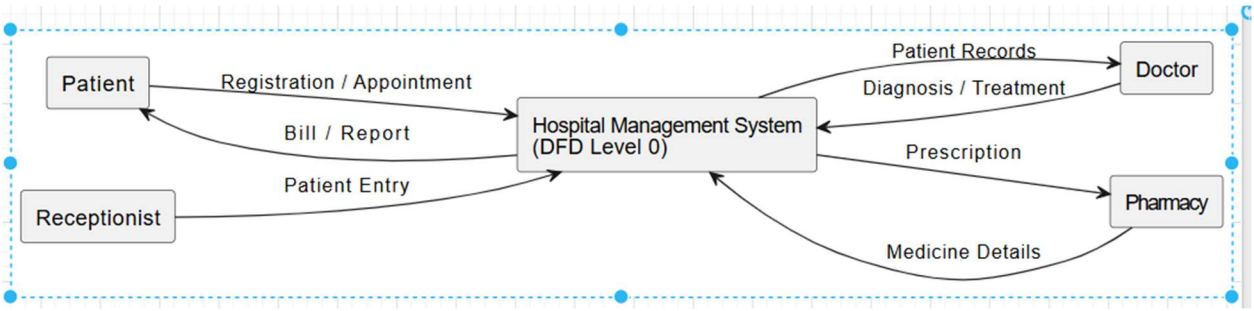
THEORY EXERCISE: Differences Between Native and Hybrid Mobile Apps

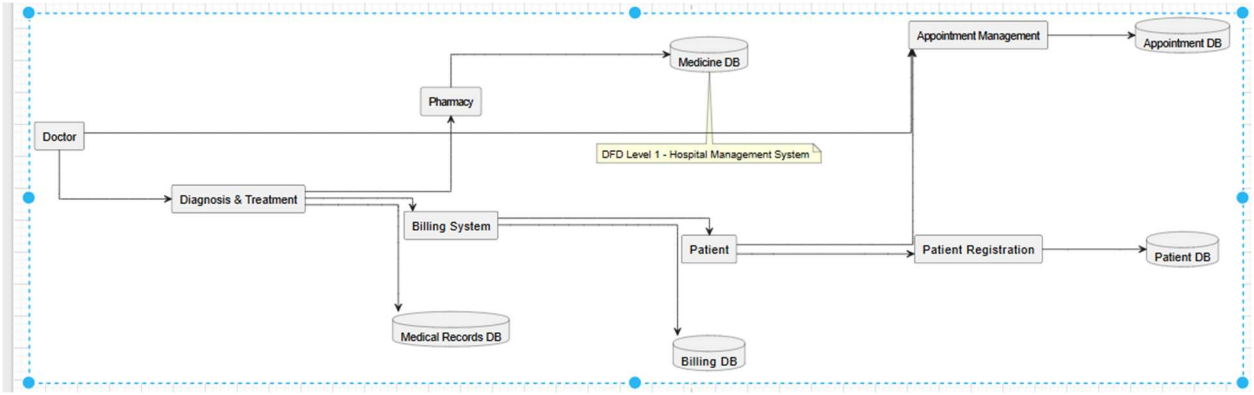
Native Mobile Apps	**Hybrid Mobile Apps**
Developed for a specific platform (iOS or Android) using platform-specific languages (Swift, Kotlin/Java)	Developed using web technologies (HTML, CSS, JavaScript) and wrapped inside a native container
High performance, faster, smooth animations	Slightly slower, depends on WebView performance
Full access to device hardware and APIs (camera, GPS, sensors)	Limited access; some device features may require plugins

Separate updates for each platform	Easier maintenance, one codebase for all platforms
WhatsApp, Snapchat, Instagram	Twitter Lite, Gmail (web wrapped), Uber Lite

30.DFD(Data Flow Diagram)

LAB EXERCISE: Create a DFD for a hospital management system.





THEORY EXERCISE: What is the significance of DFDs in system analysis?

Significance of DFDs (Data Flow Diagrams) in System Analysis:

Visual Representation: DFDs provide a graphical view of how data moves within a system, making it easier to understand complex processes.

Clarifies System Functions: They help identify what processes the system performs, what data is used, and where it flows.

Improves Communication: DFDs act as a common language between analysts, developers, and users, reducing misunderstandings.

Identifies Redundancies and Errors: By visualizing data flow, analysts can spot inefficiencies,

missing processes, or unnecessary steps early.

Basis for System Design: DFDs serve as a foundation for creating detailed system models, helping in designing databases, interfaces, and program logic.

Supports Documentation: They provide clear documentation of the system's operation for future maintenance or upgrades.

31. Desktop Application

LAB EXERCISE: Build a simple desktop calculator application using a GUI library.

Title: Simple Desktop Calculator using GUI

Objective:

To create a calculator application with a graphical interface that performs basic arithmetic operations: addition, subtraction, multiplication, and division.

Requirements:

Programming language: Python / Java / C#

GUI library: Tkinter (Python) / Swing (Java) / Windows Forms (C#)

Steps:

- The objective of this lab exercise is to design and develop a simple desktop calculator application using a graphical user interface.
- The calculator is developed using C programming language with a GUI library suitable for desktop applications.
- The application provides a graphical window containing a display area and buttons.
- Number buttons are used to enter numeric values.
- Operator buttons such as addition, subtraction, multiplication, and division are provided to perform calculations.
- The equal button is used to compute and display the result of the operation.
- A clear button is included to reset the calculator display.
- Event handling is used to capture button clicks and perform the corresponding operations.
- The application is tested for different inputs to ensure accurate calculations.
- The final output is a functional desktop calculator with a user-friendly graphical interface.

THEORY EXERCISE: What are the pros and cons of desktop applications compared to web applications?

Pros of Desktop Applications:

- Desktop applications generally provide better performance because they run directly on the local system.
- They can work without an internet connection once installed.
- Desktop applications have full access to system resources such as hardware, memory, and files.
- They are more secure for sensitive data since data can be stored locally.
- They offer a more responsive and stable user experience.

Cons of Desktop Applications:

- Desktop applications require installation on every device.
- Updates and maintenance must be done individually on each system.
- They are usually platform dependent and may not run on all operating systems.
- Sharing data and collaboration is more difficult compared to web applications.
- Deployment and distribution are more time-consuming.

Pros of Web Applications:

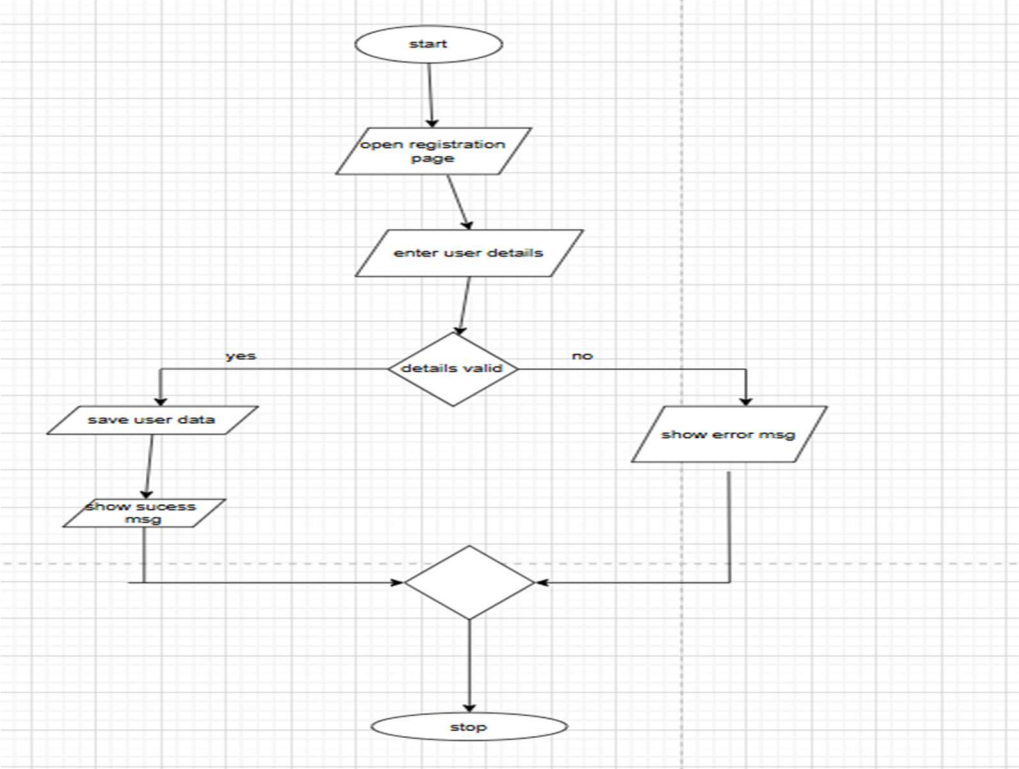
- Web applications can be accessed from any device with a web browser.
- No installation is required on the user's system.
- Updates are handled centrally and are immediately available to all users.
- They support easy data sharing and real-time collaboration.
- Web applications are platform independent.

Cons of Web Applications:

- Web applications usually require a stable internet connection.
- Performance depends on network speed and browser efficiency.
- They have limited access to system hardware and local resources.
- Security risks may be higher if proper protection is not implemented.
- Browser compatibility issues may affect user experience.

32. Flow Chart

LAB EXERCISE: Draw a flowchart representing the logic of a basic online registration system.



THEORY EXERCISE: How do flowcharts help in programming and system design?

- Flowcharts provide a clear visual representation of program logic and system processes.
- They help programmers understand the sequence of steps before writing code.
- Flowcharts make complex problems easier to analyze by breaking them into simple steps.
- They help in identifying logical errors and missing steps at an early stage.
- Flowcharts improve communication among programmers, designers, and users.
- They serve as a guide during coding and testing of programs.
- Flowcharts help in proper planning and structuring of systems.
- They act as useful documentation for future maintenance and modifications.