Tops Assignment: Basics of Computer Programming.

Module 1 – Overview of IT Industry

Q.1. What is a program? Explain in your own words what a program is and how it functions.

Ans.: A program is a collection of instructions that tells a computer what to do.

How it functions:

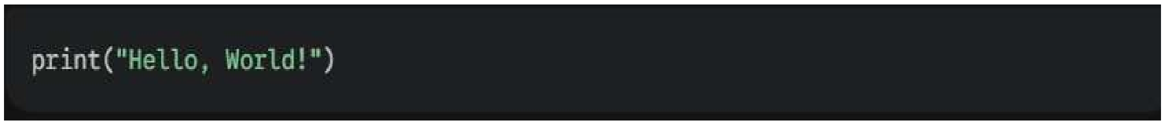
1. Input: The program takes data from the user.
2. Processing: The CPU follows the instructions to change the data.
3. Output: The computer shows the result (on a screen or file)

LAB EXERCISE:

Write a simple "Hello World" program in two different programming languages of your choice. Compare the structure and syntax.

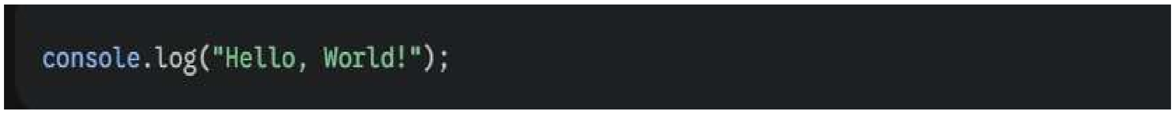
Ans.

Language 1: Python



```python
print("Hello, World!")
```

Language 2: JavaScript



```javascript
console.log("Hello, World!");
```

Comparison:

- **Python:** Very simple. It does not require semicolons or strict brackets. It reads like English.
- **JavaScript:** Requires console.log to print and typically uses semicolons; at the end of lines.

Q.2. What is programming? What are the key steps involved in the programming process?

Ans.: To create a good program, developers follow these steps:

1. Problem Definition: Understanding what needs to be solved.
2. Design: Planning the logic (using flowcharts or algorithms).

3. Coding: Writing the actual code in a programming language.
4. Testing: Checking for errors or "bugs."
5. Maintenance: Updating and fixing the program over time.

Q.3. Types of Programming Languages. What are the main differences between high-level and low-level programming languages?

Ans.: There are mainly four types of programming languages.

1. Procedural Programming Language, e.g., C
2. Object-Oriented Programming Language, e.g., C++
3. Logical Programming Language` e.g., Prolog
4. Functional Programming Language, e.g., Python

differences between high-level and low-level programming languages

High-Level Language

- For humans.
- Looks like English (e.g., print "Hello").
- Easy for us to write and read.
- Hides the complicated hardware details.
- Examples: Python, JavaScript, and Java.

Low-Level Language

- For machines.
- Looks like cryptic codes (e.g., MOV AX, 5).
- Hard for us to read, but very fast for the computer.
- Directly controls the computer's hardware.
- Examples: Assembly Language, Machine Code.

Q.4. Describe the roles of the client and server in web communication.

Ans.: Client

The client is usually the browser or app running on a user's device.

Its role is to:

- **Send requests** to the server (like "give me this webpage").
- **Display the response** to the user.
- **Handle user interactions**, such as clicks, typing, navigation.
- **Often run frontend code** (HTML, CSS, JavaScript).

Example:
When you open *google.com* in Chrome, your browser (client) sends a request to Google's server.

Server

The server is a powerful computer that stores data, processes logic, and sends responses back to clients.

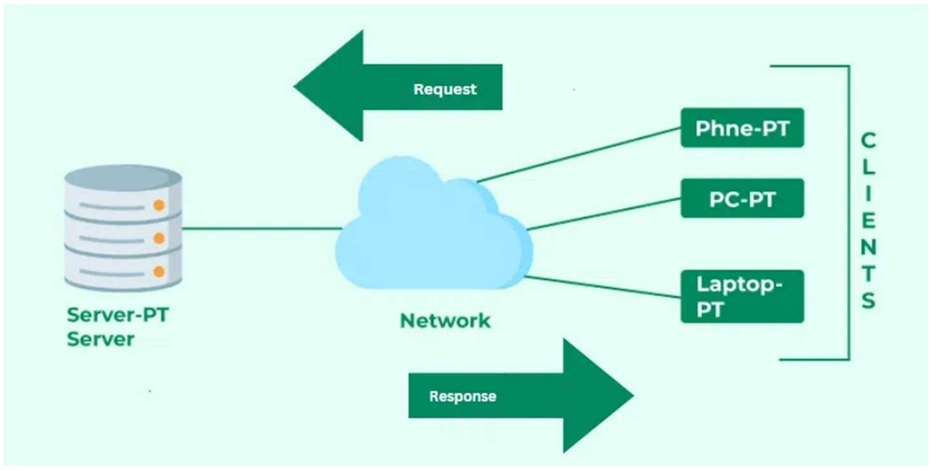
Its role is to:

- **Receive and process requests** from clients.
- **Run backend logic** (authentication, database queries, calculations).
- **Send responses** back (HTML, JSON, images, files).
- **Store and protect data** using databases.

Example:
When Chrome asks for google.com's homepage, the server prepares the page and sends it back.

LAB EXERCISE:

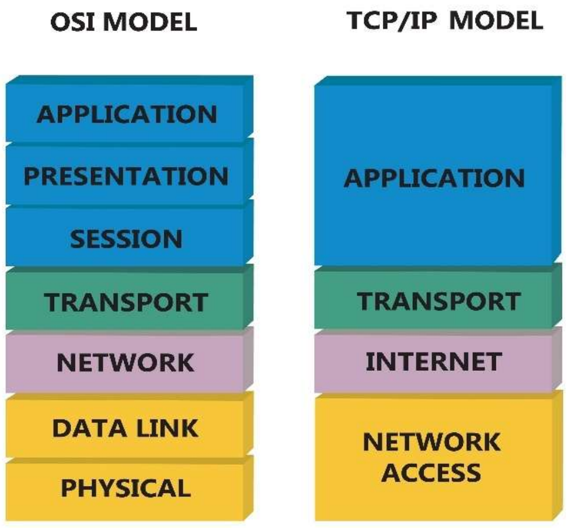
Research and create a diagram of how data is transmitted from a client to a server over the internet.



Q.5. Explain the function of the TCP/IP model and its layers.

Ans.: The TCP/IP Model

The TCP/IP model (Transmission Control Protocol/Internet Protocol) is the conceptual framework that describes how data is transmitted across a network. Unlike the 7-layer OSI model, TCP/IP is condensed into **4 layers**.



1. Application Layer (Top)

- **Function:** This is the layer where the user interacts with the software (web browsers, email clients). It formats data so the receiving application can understand it.
- **Protocols:** HTTP (Web), SMTP (Email), FTP (File Transfer), DNS.
- **Data Unit:** Data / Message.

2. Transport Layer

- **Function:** Ensures that data arrives reliably and in the correct order. It establishes a logical connection between two devices. It breaks data into smaller chunks and handles error checking.
- **Protocols:**
    - **TCP:** Reliable (checks if data arrived). Used for web browsing.
    - **UDP:** Fast but unreliable (doesn't check). Used for video streaming/gaming.
- **Data Unit:** Segment (TCP) or Datagram (UDP).

3. Internet Layer (Network Layer)

- **Function:** Handles the addressing and routing of data. It determines the best path for the data to travel across different networks to reach the destination.
- **Protocols:** IP (Internet Protocol), ICMP, ARP.
- **Data Unit:** Packet.

4. Network Interface Layer (Link Layer)

- **Function:** Handles the physical transmission of data via hardware. It interacts with the wire, fiber optic cable, or Wi-Fi signal.
- **Protocols:** Ethernet, Wi-Fi (802.11).
- **Data Unit:** Frame.

How Data Moves (Encapsulation)

When you send data, it goes *down* the stack. Each layer adds a "Header" (meta-data) to the original data. When you receive data, it goes *up* the stack, and layers strip the headers off.

1. **App Layer:** Creates the HTTP request.
2. **Transport Layer:** Adds the Source and Destination Ports.
3. **Internet Layer:** Adds Source and Destination IP addresses.
4. **Link Layer:** Adds MAC addresses (hardware IDs).

Here is a complete breakdown of the TCP/IP model alongside a practical coding exercise to see it in action.

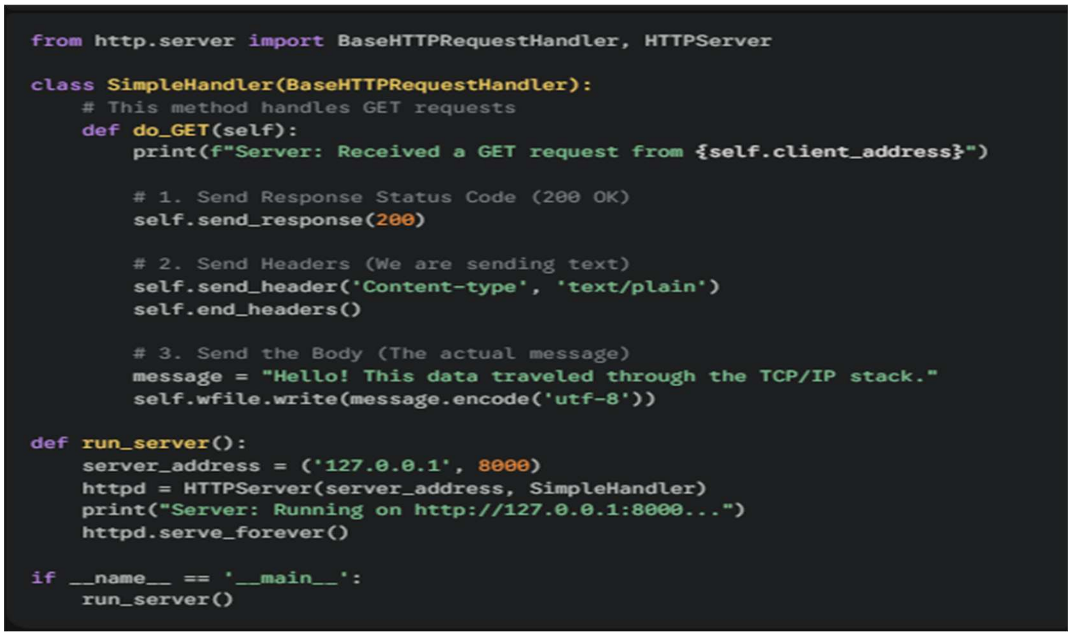
Lab Exercise: Design a simple HTTP client-server communication in any language.

Simple HTTP Client-Server in Python

Step 1: The Server Code (server.py)

This script acts as the web server. It listens on localhost (your own computer) at port 8000.

Python:

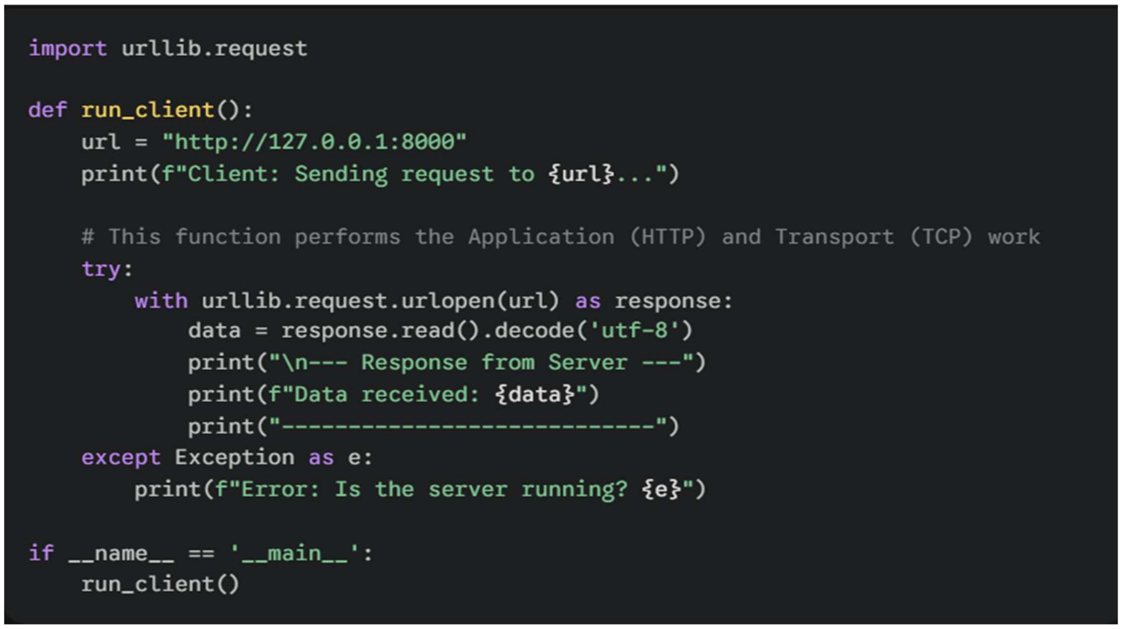


```python
from http.server import BaseHTTPRequestHandler, HTTPServer

class SimpleHandler(BaseHTTPRequestHandler):
    # This method handles GET requests
    def do_GET(self):
        print(f"Server: Received a GET request from {self.client_address}")

        # 1. Send Response Status Code (200 OK)
        self.send_response(200)

        # 2. Send Headers (We are sending text)
        self.send_header('Content-type', 'text/plain')
        self.end_headers()

        # 3. Send the Body (The actual message)
        message = "Hello! This data traveled through the TCP/IP stack."
        self.wfile.write(message.encode('utf-8'))

def run_server():
    server_address = ('127.0.0.1', 8000)
    httpd = HTTPServer(server_address, SimpleHandler)
    print("Server: Running on http://127.0.0.1:8000...")
    httpd.serve_forever()

if __name__ == '__main__':
    run_server()
```

Step 2: The Client Code (client.py)

This script acts as the browser. It initiates the TCP connection.

Python:



```python
import urllib.request

def run_client():
    url = "http://127.0.0.1:8000"
    print(f"Client: Sending request to {url}...")

    # This function performs the Application (HTTP) and Transport (TCP) work
    try:
        with urllib.request.urlopen(url) as response:
            data = response.read().decode('utf-8')
            print("\n--- Response from Server ---")
            print(f"Data received: {data}")
            print("----------------------------")
    except Exception as e:
        print(f"Error: Is the server running? {e}")

if __name__ == '__main__':
    run_client()
```

Step 3: Execution

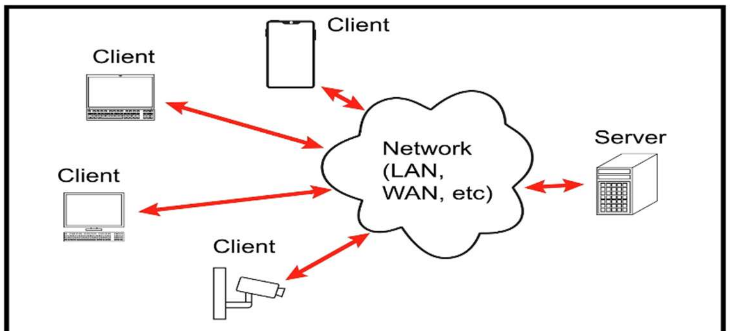
1. Open a terminal and run the server: python server.py
2. Open a **second** terminal and run the client: python client.py

Q.6. Explain Client Server Communication

Ans.: **The Client (Front-End):** This is the device or software interacting with the user. It could be a web browser (Chrome, Firefox), a mobile app (Instagram), or a desktop application. The client handles the **user interface** and presentation.

The Server (Back-End): This is a powerful computer (or cluster of computers) located remotely. It runs the application logic, communicates with databases, and stores files. It is "serving" data to the client.

The Network (The Road): The communication channel (Internet/Intranet) that connects the client and server, usually via cables or Wi-Fi.



Client–server model (architecture)

How the Communication Loop Work:

- **User Action: You click a link, type a URL, or refresh a feed on your phone.**
- **The Request: The Client packages this action into a message (usually an HTTP request) and sends it across the internet to the Server. This request asks: *"Please give me the data for the home page."***
- **Processing: The Server receives the request. It checks its database, runs necessary calculations (logic), and locates the requested information.**
- **The Response: The Server packages the result into a response message (usually containing HTML, JSON, or an image) and sends it back.**
- **Rendering: The Client receives the data and assembles it into the visual page or app screen you see.**

Q.7. How does broadband differ from fibre-optic internet?

Ans.: Broadband uses electrical signals through traditional copper cables (like telephone lines or coaxial cables). Because copper has limitations, the speed and stability are lower and long distances can weaken the connection.

Fiber-optic internet uses thin glass or plastic fibers that transmit data as light. Light travels faster than electricity, so fiber provides much higher speed, lower latency, and more stable performance. It can carry data over long distances without losing quality.

LAB EXERCISE:

Research different types of internet connections (e.g., broadband, fiber, satellite)and list their pros and cons.

Ans.: Types of Internet Connections

1. Broadband (DSL / Cable)

Pros

- Widely available in most cities and towns
- Cheaper compared to fiber
- Easy installation
- Stable connection for normal browsing and streaming

Cons

- Slower than fiber
- Speed decreases during peak hours
- Cable broadband often shares bandwidth with neighbors
- Susceptible to noise and interference

---

2. Fiber-Optic Internet

Pros

- Very high speed (up to 1 Gbps or more)
- Very stable and reliable
- Supports heavy usage: 4K streaming, cloud gaming, large uploads
- Low latency

Cons

- Not available in all areas
- More expensive than broadband
- Installation takes time

---

3. Satellite Internet

Pros

- Works in remote and rural areas
- No need for wired infrastructure
- Useful where broadband/fiber is not available

Cons

- High latency
- Weather affects performance
- Expensive plans
- Slower compared to fiber and high-speed broadband

---

4. Mobile Data (4G / 5G)

Pros

- Portable and wireless
- High speeds with 4G/5G
- Good option for mobile devices and hotspots

Cons

- Data limits on most plans
- Speed varies with location and network congestion
- Battery drain on mobile hotspots

---

5. Wi-Fi / Wireless Broadband

(Uses radio signals from a local ISP tower)
Pros

- Good for rural/urban areas
- No cables required
- Affordable

Cons

- Speed depends on distance from the tower
- Interference from buildings, weather
- Less stable than fiber

Q.8. What are the differences between HTTP and HTTPS protocols?

Ans.: HTTP (HyperText Transfer Protocol):

- Transfers data between browser and server without encryption
- Anyone on the same network can intercept or read the data
- Not safe for logins, payments, or private information
- URL starts with: http://

HTTPS (HyperText Transfer Protocol Secure):

- Encrypted using SSL/TLS
- Protects data such as passwords, card details, and personal info
- Trusted by browsers, required for secure websites
- Shows a padlock symbol
- URL starts with: https://

LAB EXERCISE:

Simulate HTTP and FTP requests using command line tools (e.g., curl)

Ans:-

Below are basic commands can be run in a terminal (Windows PowerShell, Mac, or Linux).

1. Simulate an HTTP Request using curl

GET request:

```
curl http://example.com
```

View headers:

```
curl -I http://example.com
```

POST request:

```
curl -X POST -d "name=Mithlesh&msg=Hello" http://example.com/form
```

2. Simulate an FTP Request using curl

Download a file:

```
curl ftp://speedtest.tele2.net/1MB.zip -o testfile.zip
```

Access FTP with username and password:

```
curl -u username:password ftp://yourftpserver.com/file.txt
```

Upload a file:

```
curl -T myfile.txt ftp://yourftpserver.com/
```

Q.9. What is the role of encryption in securing applications?

Ans.: Encryption protects data by converting readable information into an unreadable form so that only authorized users can access it. Even if someone steals the data, encryption makes it useless to them because they can't read it without the key.

Role of Encryption

- Protects sensitive data such as passwords, card details, and personal information
- Ensures privacy when data travels over the internet
- Prevents attackers from understanding stolen data
- Helps secure stored data (at rest) and data in transit
- Builds trust and prevents unauthorized access

LAB EXERCISE:

Identify and Explain Three Common Application Security Vulnerabilities and Suggest Solutions

Ans:

1. SQL Injection:
When an attacker inserts malicious SQL code into an input field to access or modify the database.

Example:
Entering something like
' OR '1'='1
to bypass login authentication.

Solution:

- Use parameterized queries or prepared statements
- Validate and sanitize all inputs
- Use ORM frameworks when possible

---

2. Cross-Site Scripting (XSS):
An attacker injects harmful scripts into web pages that later run in the user's browser.

Example:
A comment box that accepts <script>alert('Hacked')</script> and displays it to users.

Solution:

- Escape and sanitize user input
- Implement Content Security Policy (CSP)
- Use frameworks that auto-escape output

---

3. Broken Authentication:

Weak or improperly implemented login systems allow unauthorized users to gain access.

Example:
Allowing weak passwords or not expiring session tokens.

Solution:

- Enforce strong passwords
- Use multi-factor authentication (MFA)
- Implement secure session management
- Store passwords using hashing algorithms like bcrypt

4. Cross-Site Request Forgery (CSRF):

CSRF tricks a logged-in user into performing unwanted actions on a website without knowing it. For example, if you are logged in to your bank account and click a malicious link, it could trigger a money transfer request in the background.

Solution:

- Use CSRF tokens in forms
- Validate request origins using SameSite cookies
- Ask for user confirmation before sensitive actions

---

5. Insecure APIs:

APIs expose application data and functions. If an API lacks proper authentication or validation, attackers can misuse it to access or manipulate data.

Example:
An API endpoint that returns user details without checking if the requester is authorized.

Solution:

- Require proper authentication and authorization
- Validate inputs for every request
- Rate limit API calls
- Avoid exposing unnecessary endpoints

---

6. Unrestricted File Upload

When a website allows users to upload files without checking what they upload, attackers can upload harmful files.
The most dangerous case is uploading a script file that can run on the server.

Solution:

- Allow only specific file types
- Rename files on upload
- Scan files for malware
- Store uploaded files outside the server’s executable directory

Q.10. What is the difference between system software and application software?

Ans.: Application software:

- The most common type of software,
- application software is a computer software package that performs a specific function for a user, or in some cases, for another application.
- An application can be self-contained, or it can be a group of programs that run the application for the user.
- Examples of Modern Applications include office suites, graphics software, databases and database management programs, web browsers, word processors, software development tools, image editors and communication platforms.
- Example: Microsoft Office, Paint, PowerPoint etc..

System Software:

- These software programs are designed to run a computer's application programs and hardware.
- System software coordinates the activities and functions of the hardware and software.
- It controls the operations of the computer hardware and provides an environment or platform for all the other types of software to work in.
- The OS is the best example of system software; it manages all the other computer programs.
- Other examples of system software include the firmware, computer language translators and system utilities.
- Example: Notepad, Calculator etc..

LAB EXERCISE:

Identify and classify 5 applications you use daily as either system software or application software.

Ans:

Application	**Type of Software**	**Reason**
Windows Operating System	System Software	It manages hardware and provides the base for running other programs.
Google Chrome	Application Software	Used for browsing the internet and performing user tasks.
MS Word	Application Software	Helps create documents and is used directly by the user.
Antivirus (e.g., Quick Heal / Windows Defender)	System Software	Protects the system, monitors processes, and works in the background.
VS Code	Application Software	Used by developers to write and run code.

Q.11. What is the significance of modularity in software architecture?

Ans.: Modularity means breaking a software system into smaller, independent modules that each handle one clear responsibility. Modularity makes software easier to build, easier to manage, and easier to grow.

Why modularity matters

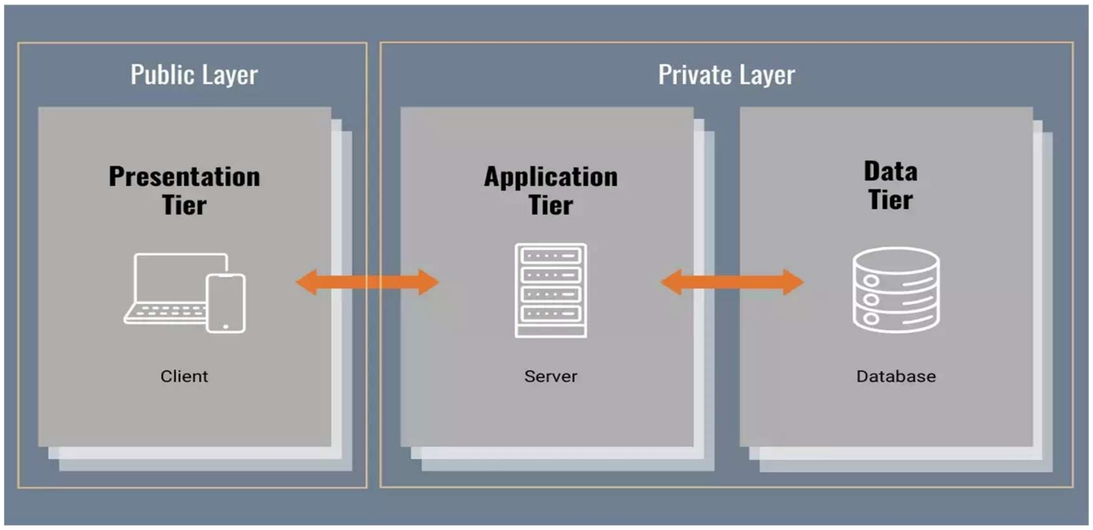
1. Easier to maintain
   When each part of the system is separate, you can update or fix one module without affecting others.
2. Improves readability
   Code becomes easier to understand because each module has a clear purpose.
3. Better reusability
   A module written once can be reused in other projects or parts of the system.

4. Helps with teamwork
   Different developers or teams can work on different modules at the same time without conflicts.
5. Reduces bugs
   Smaller, isolated modules are easier to test and debug.
6. Scalability becomes simpler
   You can improve or replace individual modules as your app grows.

LAB EXERCISE:

Design a basic three-tier software architecture diagram for a web application.

Ans:



A basic three-tier software architecture divides a web application into three logical and often physical computing tiers: the Presentation Tier, the Application Tier, and the Data Tier. This separation improves scalability, maintainability, and security.

Diagram Description: -

The architecture is structured as follows, with data flowing sequentially between the tiers:

- **Presentation Tier (Client/User Interface):** This is the top-most tier that the user interacts with. It typically runs on a user's web browser or mobile application and is built using technologies like HTML, CSS, and JavaScript. This tier communicates with the Application Tier to process user requests and display information.
- **Application Tier (Business Logic/Middle Tier):** This middle layer acts as an intermediary between the user interface and the database. It contains the core business logic, processes user inputs, and determines how to handle specific tasks. It communicates with both the Presentation Tier (receiving requests) and the Data Tier (retrieving/storing data).

- **Data Tier (Database Layer):** This bottom tier is responsible for storing, managing, and retrieving application data. It can include databases (like MySQL, PostgreSQL, or SQL Server) and other persistent storage solutions. The application tier accesses this data, but the presentation tier generally does not interact with the data tier directly.

This model ensures a clear **separation of concerns**, allowing each tier to be developed, managed, and scaled independently.

Q.12. Why are layers important in software architecture?

Ans.: Layers are important because they divide a system into clear parts, making it easier to build, understand, and manage.

Why layers matter

1. Separation of responsibilities
   Each layer does one type of job. UI handles visuals, business layer handles processing, data layer handles storage.
2. Easier maintenance
   You can change one layer without touching others. For example, updating the database doesn't affect the UI.
3. Improved testing
   Since each layer is independent, it is easier to test them separately and catch bugs early.
4. Better scalability
   You can upgrade or replace one layer (like switching from MySQL to MongoDB) without redesigning the whole system.
5. Team collaboration
   Different developers or teams can work on different layers at the same time without conflict.
6. Reuse of code
   Business logic and data access can be reused by web apps, mobile apps, and admin panels.

In simple words:

Layers keep software clean, organized, and easier to grow.

LAB EXERCISE:

Create a case study on the functionality of the presentation, business logic, and data access layers of a given software system.

Ans: **1. Presentation Layer (User Interface)**

Functionality:

- Displays menus, restaurant lists, item details, prices, and cart information
- Allows users to search, add items to cart, place orders, and make payments
- Validates basic input such as empty fields, incorrect formats, or missing information
- Communicates with the business layer through API calls

Example Activities:

- User views restaurants
- User clicks "Add to Cart"
- User submits an order

Technologies Used:
HTML, CSS, JavaScript, React, or mobile app interfaces

---

2. Business Logic Layer (Application Layer)

Functionality:

- Handles all the processing based on rules
- Calculates item totals, applies discounts, manages delivery fees
- Verifies user login and authentication
- Processes order logic: validate cart, check item availability, confirm order
- Sends and receives data from both UI and database layer

Example Activities:

- Check if a restaurant is open
- Validate payment status
- Generate order ID
- Apply coupon codes

Technologies Used:
Node.js, Java, .NET, Python backend, APIs

---

3. Data Access Layer (Database Layer)

Functionality:

- Stores, updates, and retrieves data
- Manages tables for users, restaurants, menu items, orders, payments
- Ensures data integrity and security
- Executes queries requested by the business layer

Example Activities:

- Save new orders in the database
- Fetch restaurant menu items
- Update order status from “Pending” to “Delivered”

Technologies Used:
MySQL, MongoDB, PostgreSQL

Q.13. Explain the importance of a development environment in software production.

Ans.: A development environment is important because it gives developers a safe and controlled place to write and test code before it reaches users. A development environment is the developer’s playground where ideas are built safely before they go live.

Why it matters

1. Prevents breaking the live system
   Developers can experiment, build features, and make mistakes without affecting real users.
2. Improves productivity
   Tools like debuggers, compilers, and local servers help developers work faster.
3. Allows proper testing
   You can test each feature locally before sending it to testing or production.
4. Ensures consistency
   Every developer works with the same setup, which reduces unexpected bugs.

5. Supports version control and teamwork
   Git and other tools are easier to manage in a structured environment.
6. Helps catch issues early
   Bugs found during development are cheaper and easier to fix compared to bugs found in production.

LAB EXERCISE:

Explore different types of software environments (development, testing, production).Set up a basic environment in a virtual machine.

Ans:

1. Development Environment

Purpose:
Used by developers to write, debug, and test code while building features.

Common Tools:

- Code editors (VS Code)
- Local servers (Node.js, XAMPP)
- Databases (MySQL, MongoDB)
- Version control (Git)

Example Tasks:

- Writing code
- Running local builds
- Debugging errors

---

2. Testing Environment

Purpose:
Used to test the software after development to find bugs and check performance.

Common Activities:

- Functional testing
- Load testing
- UI testing
- Integration testing

Tools:
Selenium, Postman, JMeter, Jest, Cypress

---

3. Production Environment

Purpose:
This is where the software goes live for real users.

Characteristics:

- Highly stable
- Secure
- Monitored 24/7
- Real data and real traffic

Examples:
A live website, e-commerce store, banking app, company dashboard

---

Setting Up a Basic Environment in a Virtual Machine

You can use a tool like **VirtualBox** or **VMware Workstation** to create a virtual machine.

Steps:

1. Install VirtualBox
2. Create a new VM and choose an OS (Windows or Linux)
3. Allocate RAM, CPU, and disk space
4. Install the OS inside the VM
5. Set up your tools:
    - VS Code
    - Node.js or Python
    - Database server
    - Git
6. Use this VM as your development or testing environment

This keeps your main computer safe and allows you to experiment freely.

Q.14. What is the difference between source code and machine code?

Ans.: Source Code:

- Written by programmers in human-readable languages like C, Java, Python, JavaScript
- Easy to understand and edit
- Needs to be compiled or interpreted before running

Example:

```
print("Hello World")
```

Machine Code

- Binary instructions understood directly by the computer
- Not readable by humans
- Generated by compilers or interpreters from source code
- Runs directly on the CPU

Example:

```
01001001 00000001 10101010
```

LAB EXERCISE:

Write and upload your first source code file to GitHub.

Ans: **1. Create a simple source code file**

Example: **hello.js**

```
console.log("Hello, Mithlesh!");
```

You can use any language (C, Java, Python, JavaScript).

2. Upload it to GitHub

Steps:

1. Go to **github.com** and log in
2. Click **New Repository**

3. Give it a name (example: first-source-code)
4. Click **Create Repository**
5. On your computer, open the folder with your file
6. Run these commands in terminal or PowerShell:

```
git init
git add hello.js
git commit -m "First source code"
git branch -M main
git remote add origin https://github.com/your-username/first-source-code.git
git push -u origin main
```

Q.15. Why is version control important in software development?

Ans.: Version control helps developers track, manage, and save changes made to the code over time. Version control keeps your project safe, organized, and easy to manage, whether you work alone or with a team.

Why it matters

1. Tracks every change
   You can see what changed, when it changed, and who changed it.
2. Prevents data loss
   Even if your computer crashes, your code history is safely stored.
3. Enables teamwork
   Many developers can work on the same project without overwriting each other's work.
4. Easy to undo mistakes
   You can roll back to previous versions if something breaks.
5. Supports branching
   Developers can create branches to build new features without disturbing the main code.
6. Improves project quality
   Code reviews, collaboration, and history help maintain clean and stable software.

LAB EXERCISE:

Create a GitHub repository and document how to commit and push code changes

Ans:

1. Create a New GitHub Repository

1. Go to **github.com** and log in
2. Click **New Repository**
3. Enter a name (example: my-first-repo)
4. Keep it **Public**
5. Select **Add a README file**
6. Click **Create Repository**

---

2. Clone the Repository to the Computer

Open your terminal or PowerShell and run:

```
git clone https://github.com/your-username/my-first-repo.git
```

3. Add the Code File

Inside the repo folder, create a file like:

hello.js

```
console.log("Hello, GitHub!");
```

4. Commit the Changes

Run these commands:

```
git add .
git commit -m "Added my first code file"
```

- **git add .** **- stages all changes**
- **git commit -m ""** **- saves them with a message**

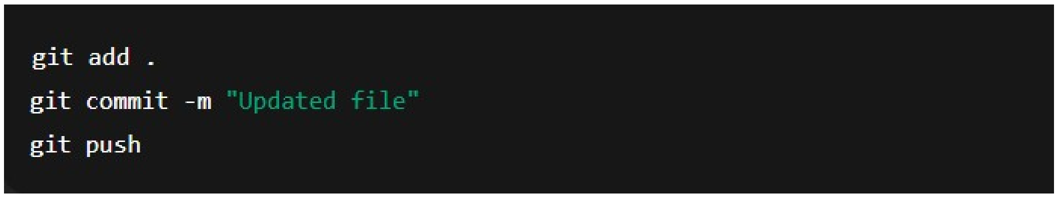
5. Push to GitHub

Send the changes to GitHub:

```
git push
```

Now the code is live in the GitHub repository.

6. Update Code and Push Again
If change made something later:



```
git add .
git commit -m "Updated file"
git push
```

Q.16. What are the benefits of using GitHub for students?

Ans.: GitHub is one of the best tools for students learning programming or software development. GitHub helps students learn faster, work better in teams, and build a portfolio that improves their career opportunities.

Key benefits for students

1. Builds a strong portfolio
   Every project you upload becomes proof of your skills.
2. Helps learn real-world development
   Students get experience with version control, branches, commits, pull requests, and collaboration.
3. Free tools through Student Pack
   Access to premium tools like GitHub Pro, JetBrains IDEs, Canva, Microsoft Azure credits, and more.
4. Easy collaboration
   Students can work together on projects without losing files or overwriting each other's work.
5. Tracks progress
   GitHub shows your activity graph, commits, and improvements over time.
6. Boosts job chances
   Recruiters often check GitHub profiles to see actual code and projects.
7. Safe cloud backup
   Your code stays online even if your laptop crashes.
8. Learn open-source culture
   Students can contribute to open-source projects and learn from others' code.

LAB EXERCISE:

Create a student account on GitHub and collaborate on a small project with a classmate.

Ans: 1. Create a GitHub Student Account

GitHub offers the GitHub Student Developer Pack with free tools.

Steps:

1. Go to GitHub Education Pack (search: *GitHub Student Pack*).
2. Sign in with your GitHub account.
3. Click Get Benefits.
4. Verify using:
   - Your college email (preferred), or
   - Upload your student ID card
5. Submit the form.
6. GitHub activates the student account usually within a few days.

---

2. Create a Small Collaborative Project

Pick a simple project like:

- A basic website
- A JavaScript calculator
- A To-do app
- A small Python script

Steps to Collaborate:

A. Create a new repository

1. Click New Repository
2. Name it (example: student-mini-project)
3. Add a README
4. Create the repo

B. Add your classmate as a collaborator

1. Go to Settings › Collaborators and teams
2. Click Add collaborator
3. Enter your classmate's GitHub username
4. They will get an invitation

C. Both students work together

Each student will:

```
git clone <repo-url>
```

To push changes:

```
git add .
git commit -m "Updated feature"
git push
```

To get updates from teammate:

```
git pull
```

This completes the collaboration task.

Q.17. What are the differences between open-source and proprietary software?

Ans.: Open-Source Software

- The source code is publicly available.
- Anyone can view, modify, and distribute it.
- Usually free to use.
- Community-driven development.
- Examples: Linux, Firefox, LibreOffice, Android.

Proprietary Software

- Source code is private and controlled by a company or individual.
- You cannot modify or distribute it.
- Usually paid or licensed.
- Official support and regular updates come from the company.
- Examples: Windows OS, Microsoft Office, Adobe Photoshop.

LAB EXERCISE

Create a list of software you use regularly and classify them into the following categories:

Ans: Types of Software

1. System Software

System software manages and controls the hardware so other software can work properly.

- Windows 10 / Windows 11
- Android OS
- BIOS / UEFI Firmware
- Device Drivers (Graphics Driver, Audio Driver)

2. Application Software

Application software helps you perform specific tasks.

- Google Chrome
- Microsoft Word
- VS Code
- WhatsApp
- Spotify
- Adobe Photoshop
- YouTube App
- Gmail
- MS Excel

3. Utility Software

Utility software helps maintain, protect, or optimize the computer.

- Windows Defender
- CCleaner
- WinRAR / 7-Zip
- Antivirus Software (Quick Heal, Kaspersky, etc.)
- Disk Cleanup
- Backup and Restore Tool

Q.18. How does GIT improve collaboration in a software development team?

Ans.: GIT makes teamwork smoother and more organized. Here's a clear explanation:

1. **Everyone can work at the same time:**
   Each developer creates their own branch to work on features without affecting others. No one overwrites someone else's work.
2. **Tracks every change:**
   GIT records who changed what and when. If something breaks, you can review the history and fix it easily.
3. **Easy merging of code:**
   Branches can be merged into the main project whenever features are ready. GIT handles differences and helps resolve conflicts.
4. **Backups and version control:**
   Every clone is a backup. Even if the server goes down, your local copy still has full history.
5. **Clear workflow:**
   Teams can follow workflows like feature branching, pull requests, and code reviews to maintain quality.
6. **Integrates with GitHub/GitLab/Bitbucket:**
   These platforms help developers review code, discuss changes, and manage issues, making collaboration even better.

LAB EXERCISE: Follow a GIT tutorial to practice cloning, branching, and merging repositories.

Ans: 1. Cloning a Repository

- Used the command
- This created a local copy of the remote project.

```
git clone <repository-url>
```

2. Creating a New Branch

- Ran
- This allowed me to work on new changes without affecting the main branch.

```
git branch feature-demo
git checkout feature-demo
```

3. Merging Branches
    - After completing the changes, switched to the main branch and merged:
    - The updates from the feature branch were added to the main codebase.

```
git checkout main
git merge feature-demo
```

4. Pushing Changes to GitHub
    - Used
    - This uploaded the latest changes to the remote repository.

```
git push origin main
```

Q.19. What is the role of application software in businesses?

Ans.: Application software plays a major role in helping businesses operate smoothly and efficiently. Here's a clear explanation:

1. **Automates routine tasks:**
   Software handles repetitive work like data entry, billing, and reporting. This saves time and reduces errors.
2. **Improves communication:**
   Email clients, messaging apps, and video meeting tools help employees share information quickly.
3. **Enhances decision-making:**
   Tools like spreadsheets and data analysis software help businesses analyze trends and make better decisions.
4. **Increases productivity:**
   Tools like word processors, project management apps, and CRM systems help employees work faster and stay organized.
5. **Supports customer management:**
   CRM software stores customer data and helps businesses manage sales, support, and marketing.

6. **Improves accuracy:**
   Business applications reduce human errors in calculations, record-keeping, and financial work.

7. **Enables remote work:**
   Cloud-based applications allow employees to access data from anywhere, improving flexibility.

LAB EXERCISE

Write a report on the various types of application software and how they improve productivity.

Ans: Application software is designed to help users perform specific tasks more efficiently. It covers a wide range of programs that support personal, educational, and business activities. Different types of application software improve productivity by automating work, reducing manual effort, and simplifying complex tasks. Below are the major categories:

1. **Word Processing Software:**

   Examples: Microsoft Word, Google Docs
   These tools help users create, edit, and format documents. They improve productivity by providing spelling checks, templates, and easy formatting options.

2. **Spreadsheet Software:**

   Examples: Microsoft Excel, Google Sheets
   Spreadsheets help in organizing data, performing calculations, creating charts, and analyzing information. They save time by automating formulas and reducing manual calculations.

3. **Presentation Software:**

   Examples: Microsoft PowerPoint, Google Slides
   These tools help create visual presentations. They improve productivity by offering ready-made templates, easy slide layouts, and quick editing features.

4. **Database Management Software:**

   Examples: MySQL, Microsoft Access
   These systems store, manage, and retrieve data efficiently. They help businesses maintain accurate records and reduce data handling errors.

5. **Graphic Design and Multimedia Software:**

   Examples: Adobe Photoshop, Canva, CorelDRAW
   These tools help create images, videos, and graphics. They boost productivity by offering editing tools, filters, and templates that speed up creative work.

6. **Web Browsers:**

Examples: Google Chrome, Firefox
Browsers allow users to access online resources quickly. They improve productivity by giving fast access to information and web-based tools.

7. **Communication Software:**

Examples: WhatsApp, Gmail, Zoom
These tools support messaging, email, and video conferencing. They improve productivity by enabling fast communication and remote collaboration.

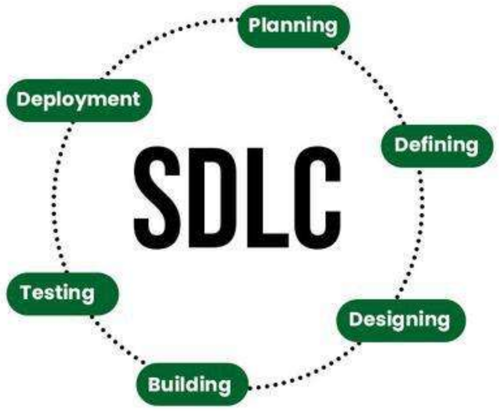
8. **Project Management Software**:

Examples: Trello, Asana, Jira
These help track tasks, assign deadlines, and monitor project progress. They keep teams organized and reduce time spent on planning manually.

Q.20. Create a flowchart representing the Software Development Life Cycle (SDLC). What are the main stages of the software development process?

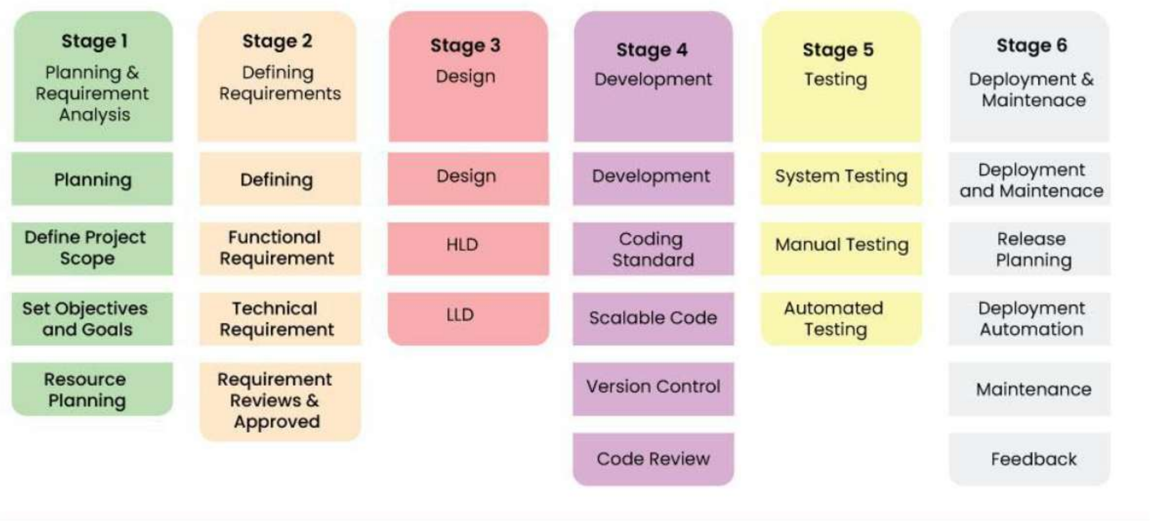
Ans.: Software Development Life Cycle (SDLC) is a structured process that is used to design, develop, and test high-quality software. SDLC, or software development life cycle, is a methodology that defines the entire procedure of software development step-by-step. The goal of the SDLC life cycle model is to deliver high-quality, maintainable software that meets the user's requirements.

SDLC in software engineering models outlines the plan for each stage so that each stage of the software development model can perform its task efficiently to deliver the software at a low cost within a given time frame that meets users requirements. In this article we will see Software Development Life Cycle (SDLC) in detail.



Stages of the Software Development Life Cycle:

SDLC specifies the tasks to be performed at various stages by a software engineer or developer. It ensures that the end product is able to meet the customer's expectations and fits within the overall budget. Hence, it's vital for a software developer to have prior knowledge of this software development process. SDLC is a collection of these six stages, and the stages of SDLC are as follows:



Stage 1: Planning and Requirement Analysis:

Planning is a crucial step in everything, just as in software development. In this same stage, requirement analysis is also performed by the developers of the organization. This is attained from customer inputs, and sales department/market surveys.

The information from this analysis forms the building blocks of a basic project. The quality of the project is a result of planning. Thus, in this stage, the basic project is designed with all the available information.

Stage 2: Defining Requirements:

In this stage, all the requirements for the target software are specified. These requirements get approval from customers, market analysts, and stakeholders.
This is fulfilled by utilizing SRS (Software Requirement Specification). This is a sort of document that specifies all those things that need to be defined and created during the entire project cycle.

Stage 3: Designing Architecture:

SRS is a reference for software designers to come up with the best architecture for the software. Hence, with the requirements defined in SRS, multiple designs for the product architecture are present in the Design Document Specification (DDS).

This DDS is assessed by market analysts and stakeholders. After evaluating all the possible factors, the most practical and logical design is chosen for development.

Stage 4: Developing Product:

At this stage, the fundamental development of the product starts. For this, developers use a specific programming code as per the design in the DDS. Hence, it is important for the coders to follow the protocols set by the association. Conventional programming tools like compilers, interpreters, debuggers, etc. are also put into use at this stage. Some popular languages like C/C++, Python, Java, etc. are put into use as per the software regulations.

Stage 5: Product Testing and Integration:

After the development of the product, testing of the software is necessary to ensure its smooth execution. Although, minimal testing is conducted at every stage of SDLC. Therefore, at this stage, all the probable flaws are tracked, fixed, and retested. This ensures that the product confronts the quality requirements of SRS.

Documentation, Training, and Support: Software Documentation is an essential part of the software development life cycle. A well-written document acts as a tool and means to information repository necessary to know about software processes, functions, and maintenance. Documentation also provides information about how to use the product. Training to improve the current or future employee performance by increasing an employee's ability to work through learning, usually by changing his attitude and developing his skills and understanding.

Stage 6: Deployment and Maintenance of Products:

After detailed testing, the conclusive product is released in phases as per the organization's strategy. Then it is tested in a real industrial environment. It is important to ensure its smooth performance. If it performs well, the organization sends out the product as a whole. After retrieving beneficial feedback, the company releases it as it is or with auxiliary improvements to make it further helpful for the customers. However, this alone is not enough.

Q.21. Why is the requirement analysis phase critical in software development?

Ans.: Requirement analysis is one of the most important steps in software development because:

1. It defines what the software should do
   Without clear requirements, developers may build the wrong features.
2. Prevents misunderstandings
   Requirements ensure that developers, clients, and users all have the same expectations.
3. Saves time and cost
   Fixing mistakes later is expensive. Understanding the requirements early avoids rework.
4. Helps in planning and design
   The design, architecture, and workflow depend entirely on accurate requirements.
5. Improves quality
   When requirements are well-defined, the final software is more likely to meet user needs.

6. Forms the base for testing
   Testers use requirements to check whether the software works as expected.

LAB EXERCISE

Write a requirement specification for a simple Library Management System.

Ans:

1. Introduction

The Library Management System is designed to help manage books, members, borrowing records, and returns. It reduces manual paperwork and makes the library more organized.

2. Functional Requirements

2.1 User Management

- The system should allow the admin to add new members.
- Admin should be able to update or delete member information.
- Members should be able to view their issued books and due dates.

2.2 Book Management

- Admin can add new books with details like title, author, category, and ISBN.
- Admin can update book information.
- Admin can remove books from the catalog.
- Users can search for books by title, author, or category.

2.3 Issue and Return Management

- Admin can issue books to members.
- The system should record the issue date and due date.
- Admin can mark a book as returned.
- The system should calculate fines for late returns.

2.4 Inventory Tracking

- The system should maintain the number of available copies.
- When a book is issued, the available count should decrease.
- When a book is returned, the count should increase.

2.5 Reports

- Admin can view reports such as:
    - List of issued books
    - List of overdue books
    - Total members
    - Total books in the library

3. Non-Functional Requirements

3.1 Usability : The interface should be simple and easy to navigate.

3.2 Performance: The system should respond quickly even when handling large amounts of data.

3.3 Security: Only authorized users should access the admin features.

3.4 Reliability: The system should store data safely and should not lose records.

Q.22. What is the role of software analysis in the development process?

Ans.: Software analysis plays a major role in ensuring a project starts with clarity and direction. Here's a simple explanation:

1. Identifies what the system must do
   It helps gather and understand all the functional and non-functional requirements before development starts.
2. Reduces misunderstandings
   Analysis ensures that developers, clients, and stakeholders share a common understanding of the project.
3. Helps in planning
   By analysing the requirements, the team can estimate time, cost, and resources needed for the project.
4. Creates a strong foundation
   Good analysis leads to better system design, smoother development, and fewer changes later.
5. Improves overall quality
   When you clearly know what to build, the final software is more accurate and reliable.
6. Supports testing
   Test cases are designed based on the analysis, which ensures all requirements are covered.

LAB EXERCISE

Write a requirement specification for a simple Library Management System.

Ans:

1. User Roles

- Customer: Browses products, adds to cart, places orders.
- Admin: Manages products, categories, orders, and customer records.

2. Functional Requirements

2.1 User Registration and Login

- Customers can create an account with email, phone, and password.
- Users can log in and log out securely.
- Admin has a separate login panel.

2.2 Product Browsing and Search

- Users can view all products by category.
- Users can search products by name, price, or category.
- Product details page shows images, price, description, and reviews.

2.3 Shopping Cart

- Users can add products to the cart.
- Users can update quantity or remove items.
- The system displays the total price automatically.

2.4 Checkout and Payment

- Users enter address and contact details.
- Users can choose payment methods such as UPI, card, or cash on delivery.
- After payment, an order confirmation is generated.

2.5 Order Management

- Users can view order history and track order status.
- Admin can view, update, or cancel orders.
- Order statuses include: Pending, Packed, Shipped, Delivered, Cancelled.

2.6 Product Management (Admin)

- Admin can add new products with images and details.

- Admin can update product information.
- Admin can delete products or categories.

2.7 Review and Rating System

- Customers can leave reviews after purchasing.
- Reviews display on the product detail page.

2.8 Notifications

- Users receive email notifications for orders, cancellations, and delivery updates.

Q.23. What are the key elements of system design?

Ans: 1. Architecture Design:

Defines how the system is structured, including components, modules, and how they connect.

2. Data Design:

Covers how data will be stored, managed, and accessed. Includes database structure, tables, and relationships.

3. Interface Design:

Focuses on user interfaces, dashboards, and how users interact with the system.

4. Component Design:

Breaks the system into smaller functional units. Each component handles a specific task.

5. Security Design:

Ensures the system is protected from unauthorized access and data breaches.

6. Performance Design:

Makes sure the system can handle heavy loads, high traffic, and fast response times.

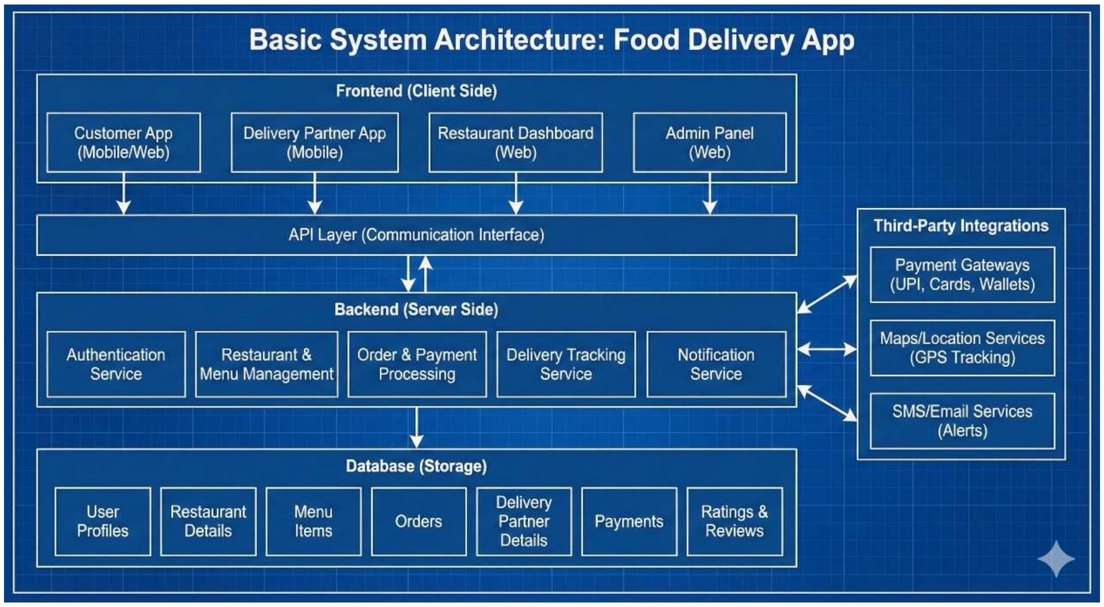
7. Integration Design:

Defines how the system will communicate with external services such as payment gateways, email APIs, or map services.

LAB EXERCISE

Design a basic system architecture for a food delivery app.

Ans:


Basic System Architecture: Food Delivery App
Frontend (Client Side)
Customer App (Mobile/Web)
Delivery Partner App (Mobile)
Restaurant Dashboard (Web)
Admin Panel (Web)
API Layer (Communication Interface)
Backend (Server Side)
Authentication Service
Restaurant & Menu Management
Order & Payment Processing
Delivery Tracking Service
Notification Service
Third-Party Integrations
Payment Gateways (UPI, Cards, Wallets)
Maps/Location Services (GPS Tracking)
SMS/Email Services (Alerts)
Database (Storage)
User Profiles
Restaurant Details
Menu Items
Orders
Delivery Partner Details
Payments
Ratings & Reviews

1. User Roles

- Customer: Orders food.
- Restaurant: Manages menu and order preparation.
- Delivery Partner: Picks up and delivers orders.
- Admin: Manages users, restaurants, and system settings.

2. Basic System Architecture

A. Frontend (Client Side)

- Mobile App or Web App for customers
- Separate app for delivery partners
- Dashboard for restaurants
- Admin panel

B. Backend (Server Side)

- Handles authentication
- Manages restaurants and menus
- Processes orders and payments
- Tracks delivery status
- Sends notifications
- Stores user information

C. Database

Stores the following:

- User profiles
- Restaurant details
- Menu items
- Orders
- Delivery partner details
- Payments
- Ratings and reviews

D. API Layer

- Provides communication between frontend and backend
- Example APIs:
    - Login API
    - Menu API
    - Order API
    - Payment API
    - Delivery tracking API

E. Third-Party Integrations

- Payment gateways: UPI, Cards, Wallets
- Maps/Location services: GPS for live tracking
- SMS/Email: Send alerts and updates

F. Workflow Overview

1. Customer browses menu
2. Customer places an order
3. Restaurant receives and accepts the order
4. Delivery partner is assigned
5. Order is picked up and delivered
6. Customer receives notification
7. Payment and review are stored

Q.24. Why is software testing important?

Ans: Software testing is important because it ensures the product works the way it should. Here's a clear explanation:

1. **Finds errors early:**
   Testing helps detect bugs before the software reaches users.
2. **Improves quality:**
   A tested product is more reliable, stable, and safe to use.
3. **Prevents failures:**
   Bugs can cause crashes, data loss, or incorrect results. Testing avoids these problems.
4. **Builds user trust:**
   When software works smoothly, customers trust the product and the company.
5. **Reduces cost:**
   Fixing bugs during development is cheaper than fixing them after release.
6. **Ensures the software meets requirements:**
   Testers check whether the final product matches what was planned.
7. **Helps maintain performance and security:**
   Testing identifies issues that affect speed, performance, and safety.

LAB EXERCISE

Develop test cases for a simple calculator program.

Ans:

Test Case ID	**Input**	**Operation**	**Expected Output**	**Description**
TC01	5, 3	Addition	8	Adds two positive numbers
TC02	-5, 3	Addition	-2	Adds a negative and a positive number
TC03	10, -4	Addition	6	Adds a positive and a negative number

Test Case ID	Input	Operation	Expected Output	Description
TC04	7, 3	Subtraction	4	Subtracts a smaller number from a larger number
TC05	3, 7	Subtraction	-4	Subtracts a larger number from a smaller number
TC06	-5, -2	Subtraction	-3	Subtracts two negative numbers
TC07	6, 4	Multiplication	24	Multiplies two positive numbers
TC08	-3, 5	Multiplication	-15	Multiplies a negative and a positive number
TC09	-4, -6	Multiplication	24	Multiplies two negative numbers
TC10	8, 2	Division	4	Divides two positive numbers
TC11	9, 0	Division	Error	Division by zero should return error/exception
TC12	-12, 3	Division	-4	Divides a negative number by a positive number

Q.24. What types of software maintenance are there?

Ans: There are four main types of software maintenance:

1. Corrective Maintenance:

Fixes errors or bugs found after the software is released.
Example: fixing a crash, removing a broken feature.

2. Adaptive Maintenance

Updates the software to keep it working in a changing environment.
Example: updating software to work with new operating systems or browsers.

3. Perfective Maintenance:

Improves the software by adding enhancements or optimizing performance.
Example: speeding up load times or adding new features based on user feedback.

4. Preventive Maintenance:

Prevents future issues by cleaning code, improving structure, and updating libraries.
Example: refactoring code or updating security patches.

LAB EXERCISE

Document a real-world case where a software application required critical maintenance.

Ans:

Real-World Case: WhatsApp Server Outage (2022)

In October 2022, WhatsApp experienced a global outage that affected millions of users. People were unable to send or receive messages for almost two hours. This issue occurred due to a server-side configuration error.

Why Critical Maintenance Was Needed

- Millions of users rely on WhatsApp for communication.
- Businesses, customer support teams, and delivery companies faced delays.
- The outage affected multiple countries, so quick action was necessary.

Maintenance Performed

- The backend engineering team investigated the server configuration.
- They identified an issue in the routing layer that was blocking message delivery.
- The team performed emergency corrective maintenance by fixing the faulty configuration.
- After the repair, they monitored servers to make sure stability was restored.

Outcome

- WhatsApp resumed normal service within a few hours.
- Meta shared updates to maintain transparency.
- The incident showed the importance of monitoring, quick diagnosis, and timely maintenance.

Q.25. What are the key differences between web and desktop applications?

Ans: **1. Installation**

- Web Applications: No installation needed. They run in a browser.
- Desktop Applications: Must be installed on the computer.

2. Platform Dependency

- Web Applications: Work on any device with a browser and internet.
- Desktop Applications: Often platform-specific (Windows, macOS, Linux) unless built cross-platform.

3. Updates

- Web Applications: Updated directly on the server. Users always get the latest version.
- Desktop Applications: Users need to download updates manually or through an installer.

4. Accessibility

- Web Applications: Accessible from anywhere with internet access.
- Desktop Applications: Limited to the system where they are installed.

5. Performance

- Web Applications: Depend on internet speed and browser performance.
- Desktop Applications: Usually faster because they use local system resources.

6. Storage

- Web Applications: Store data on cloud servers or remote databases.
- Desktop Applications: Store data locally on the device unless cloud sync is added.

7. Security

- Web Applications: Need strong security because they are publicly accessible online.
- Desktop Applications: Less exposed but still need protection from local threats.

8. Development Complexity

- Web Applications: Require web technologies like HTML, CSS, JavaScript, and backend services.
- Desktop Applications: Use system-specific programming languages or frameworks.

9. Offline Use

- Web Applications: Usually need internet, though some support offline mode.

- Desktop Applications: Work offline by default.

Q.26. What are the advantages of using web applications over desktop applications?

Ans: 1. No Installation Needed

Users can access the application directly from a browser. No need to download or install anything.

2. Accessible from Anywhere

As long as there is internet, users can access the app from any device, anywhere in the world.

3. Works on Multiple Devices

Web apps run on laptops, phones, tablets, and even smart TVs without platform restrictions.

4. Easy and Instant Updates

Developers can update the app on the server. Users get the latest version automatically without manual updates.

5. Lower Hardware Requirements

Since processing happens mostly on the server, users don't need powerful devices to run the app.

6. Easier Maintenance

Bugs, patches, and new features can be fixed or added in one place instead of updating each user's device.

7. Centralized Data Storage

Web apps store data on servers or cloud systems, making it easy to back up, recover, and sync across devices.

8. Cost-Effective

No installation, fewer compatibility issues, and centralized management reduce both development and support costs.

9. Easy Collaboration

Multiple users can work together in real time.
Example: Google Docs, Trello, Notion.

10. Scalable

New features, more users, and higher storage can be handled easily through server-side scaling.

Q.27. What role does UI/UX design play in application development?

Ans: UI/UX design plays a major role in how users experience and interact with an application.

1. Makes the app easy to use:

A good UI/UX design helps users understand the app quickly without confusion. Simple navigation and clear layouts improve usability.

2. Improves user satisfaction:

When the app looks good and works smoothly, users enjoy using it. This increases trust and encourages them to return.

3. Reduces user errors:

Clear buttons, readable labels, and logical workflows help people avoid mistakes while performing tasks.

4. Helps achieve business goals:

A well-designed app increases user engagement leads to more conversions, and supports the purpose of the product.

5. Enhances accessibility:

UI/UX makes sure the app is usable for all kinds of users, including those with disabilities, by following accessibility principles.

6. Saves development time:

Proper UI/UX planning avoids major redesigns later. Developers get clear guidelines and can build features without confusion.

7. Creates a strong first impression:

Users decide quickly whether to continue using an app. Good UI/UX helps create a positive, trustworthy impression.

8. Supports consistency:

UI/UX guidelines ensure that fonts, colors, buttons, and layouts look uniform across all screens, making the app feel professional.

Q.28. What role does UI/UX design play in application development?

Ans: Native and hybrid mobile apps differ mainly in how they are built, how they perform, and how they access device features.

Native apps:

Native Apps are built specifically for one platform, like Android or iOS. Developers use platform languages such as Kotlin or Java for Android and Swift or Objective-C for iOS. Because they are built for a particular operating system, they run faster, give smoother animations, and can use device features like camera, GPS, and sensors without limitations. Their user experience also feels more consistent with the platform.

Hybrid apps:

Hybrid apps are built using web technologies like HTML, CSS, and JavaScript, but they run inside a native shell. Frameworks like Ionic, Cordova, and React Native help bridge web code with mobile features. Hybrid apps take less time and money to develop because one codebase works on both Android and iOS. But they may not perform as fast as native apps, especially for heavy graphics or animations, and sometimes face limitations with advanced device features.

native apps win in performance and platform experience, while hybrid apps save time and cost by allowing cross-platform development with a single codebase.

Q.29. What is the significance of DFDs in system analysis?

Ans: A Data Flow Diagram (DFD) is one of the most essential tools in the structured systems analysis and design method. It is a graphical representation of the "flow" of data through an information system.

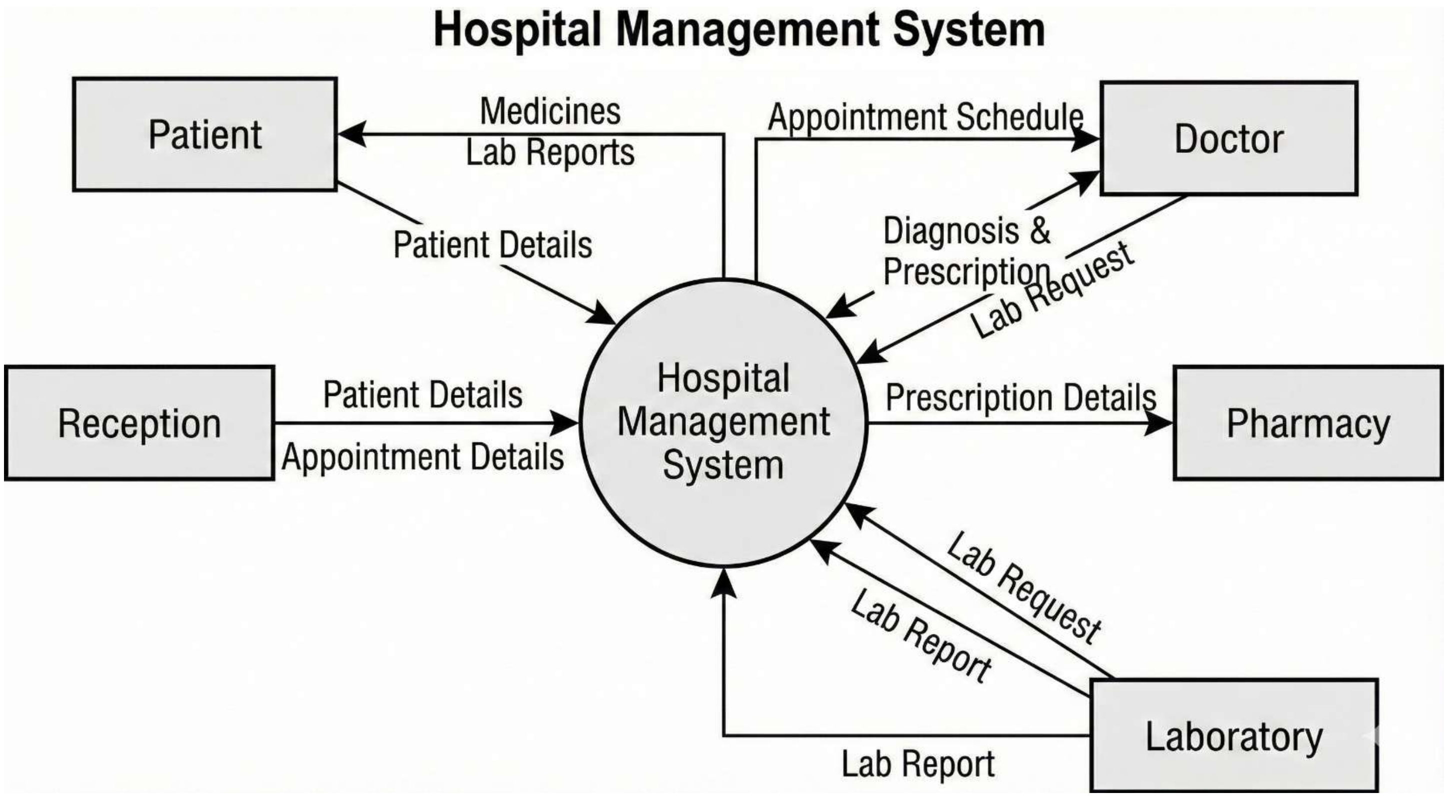
Significance of DFDs In System Analysis:

- DFDs help analysts understand how data moves inside a system.
- They show where the data starts, how it gets processed, and where it ends up.
- This makes it easier to spot gaps, unnecessary steps, or problems in the workflow.
- A DFD also helps both technical and non-technical people visualize the system clearly without getting into code.
- It supports better planning, reduces misunderstandings, and creates a strong foundation before actual development begins.

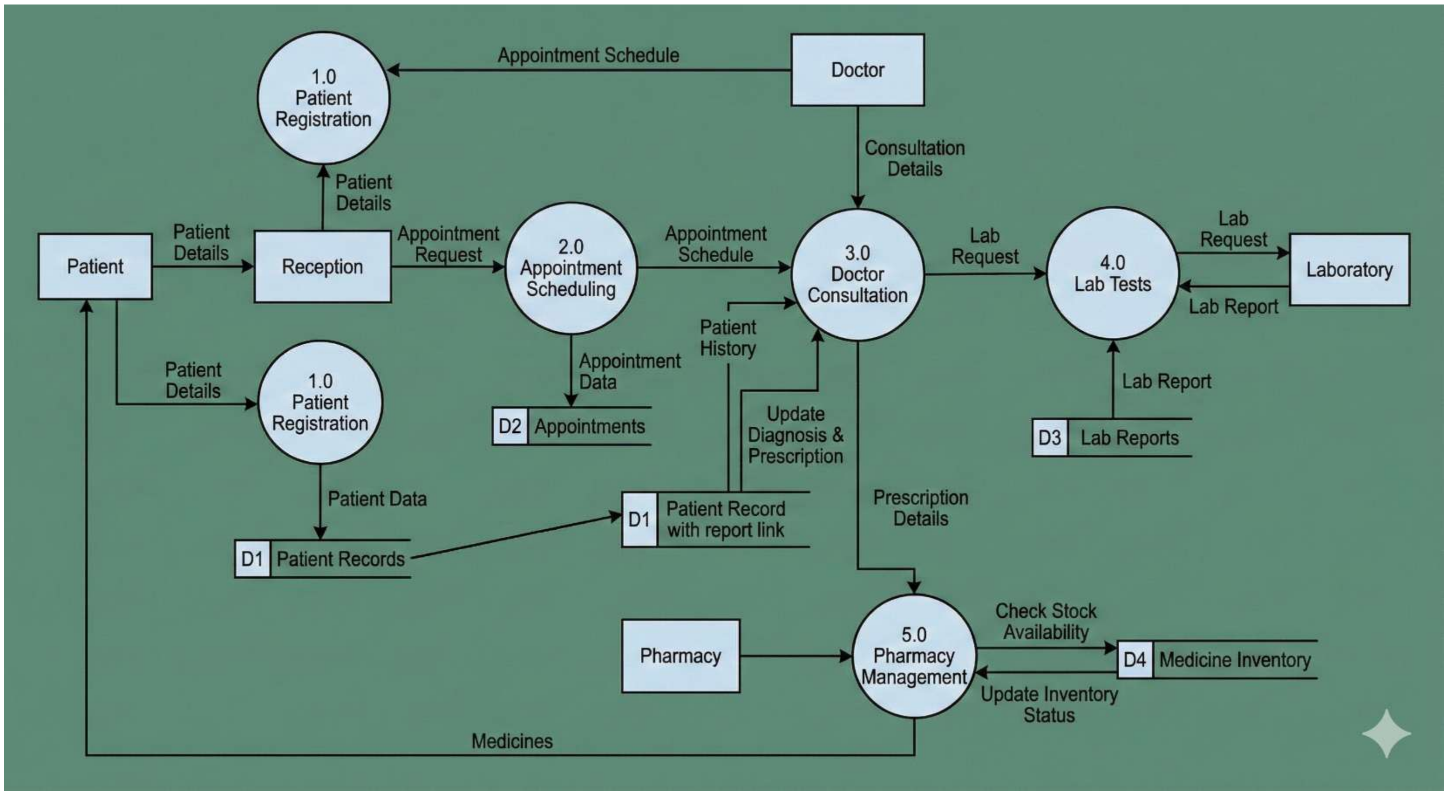
LAB EXERCISE: Create a DFD for a hospital management system.

Ans:

0 level:



1 level:



Q.30. What are the pros and cons of desktop applications compared to web applications?

Ans: Pros of Desktop Applications

1. **Better performance**
   Desktop apps usually run faster because they use the full power of the device without depending on the Internet.

2. **Offline access**
   They work even without an Internet connection, which makes them reliable for tasks like editing documents, gaming, or media production.

3. **Stronger access to system resources**
   They can interact more deeply with hardware such as printers, USB devices, GPUs, and file systems.

4. **Higher stability**
   Once installed, they are less affected by browser updates, connection drops, or server issues.

Cons of Desktop Applications

1. **Installation required**
   Users must download and install the application, which can be a barrier.

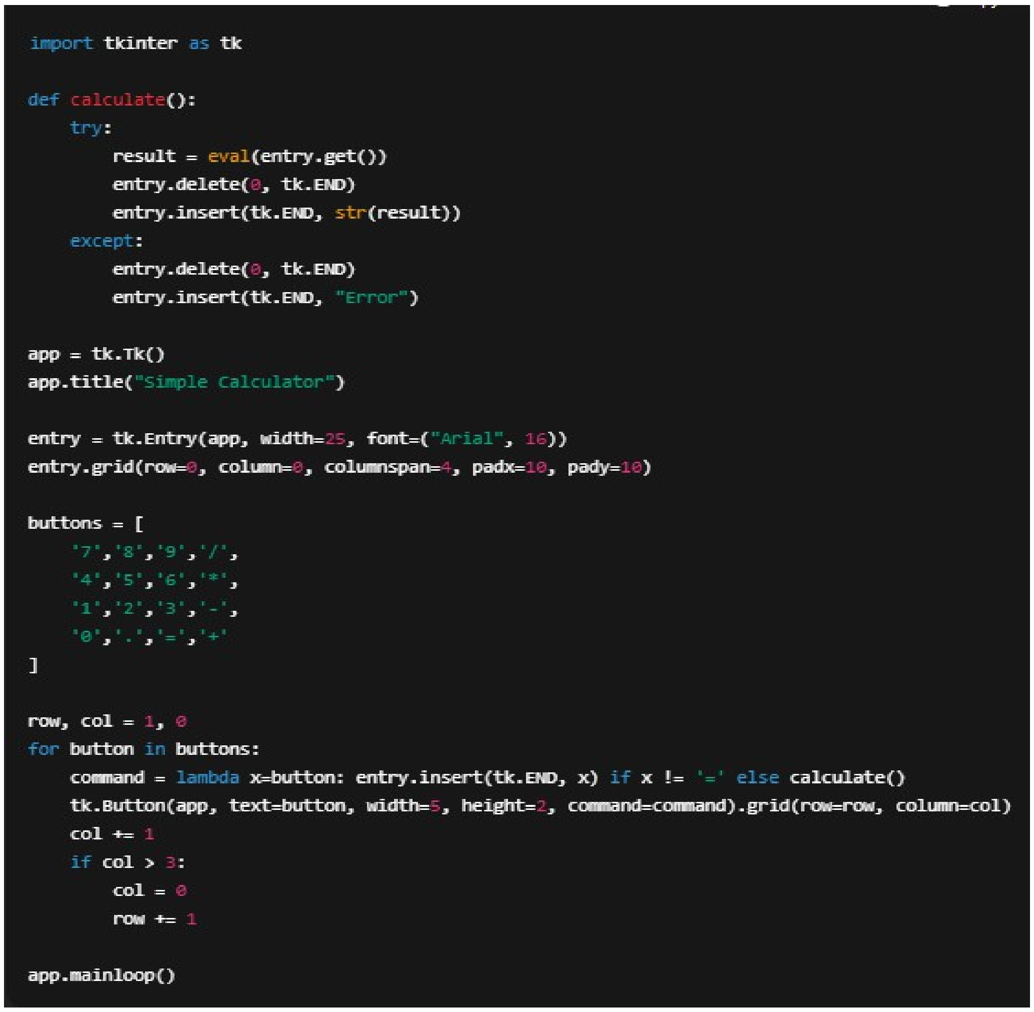
2. **Platform dependent**
   A desktop app built for Windows won't run on Mac or Linux unless separately developed.

3. **Harder to update**
   Updates have to be installed frequently, and older versions may cause compatibility issues.

4. **Limited accessibility**
   Users can access it only on the device where it is installed, unlike web apps that run anywhere through a browser.

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

Ans: Here's a small example (Python + Tkinter)



```python
import tkinter as tk

def calculate():
    try:
        result = eval(entry.get())
        entry.delete(0, tk.END)
        entry.insert(tk.END, str(result))
    except:
        entry.delete(0, tk.END)
        entry.insert(tk.END, "Error")

app = tk.Tk()
app.title("Simple Calculator")

entry = tk.Entry(app, width=25, font=("Arial", 16))
entry.grid(row=0, column=0, columnspan=4, padx=10, pady=10)

buttons = [
    '7','8','9','/',
    '4','5','6','*',
    '1','2','3','-',
    '0','.','=','+'
]

row, col = 1, 0
for button in buttons:
    command = lambda x=button: entry.insert(tk.END, x) if x != '=' else calculate()
    tk.Button(app, text=button, width=5, height=2, command=command).grid(row=row, column=col)
    col += 1
    if col > 3:
        col = 0
        row += 1

app.mainloop()
```

Q.31. How do flowcharts help in programming and system design?

Ans: Flowcharts make the planning and development process easier by giving a visual explanation of how a system works. Here's a clear and simple answer:

1. They help you understand the logic clearly

A flowchart shows how the program flows step by step. This makes it easier to understand the process before writing any code.

2. They reduce errors

When you visualize each step, it becomes easy to spot mistakes or missing conditions early.

3. They simplify communication

Flowcharts help developers, students, and clients understand the system without needing technical language.

4. They make debugging easier

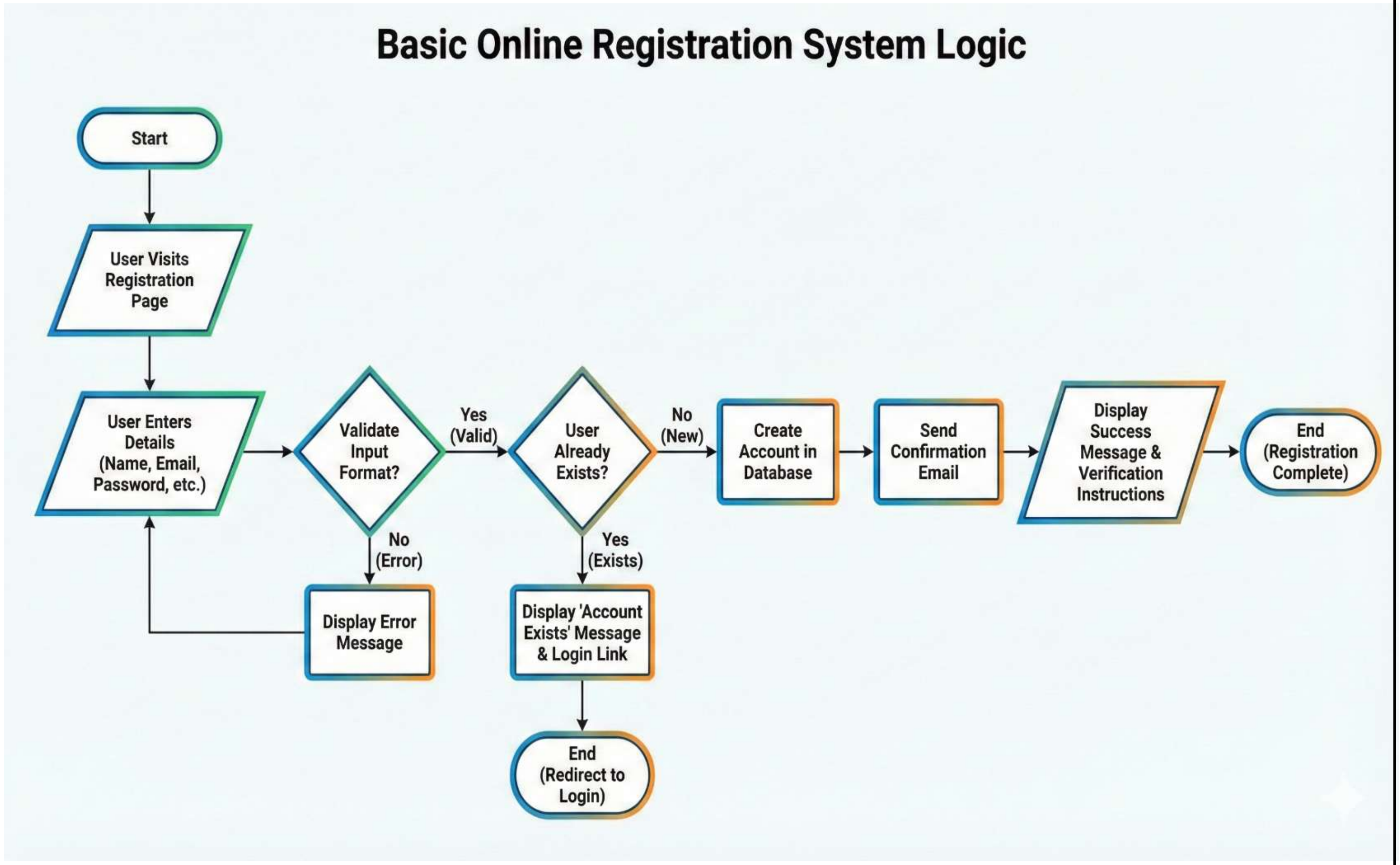
When something goes wrong, you can refer to the flowchart to see which step might be failing.

5. They help with proper documentation

Flowcharts become part of the project documentation, useful for future developers and maintenance.

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

Ans:


Basic Online Registration System Logic
Start
User Visits Registration Page
User Enters Details (Name, Email, Password, etc.)
Validate Input Format?
Yes (Valid)
No (Error)
Display Error Message
User Already Exists?
No (New)
Yes (Exists)
Display 'Account Exists' Message & Login Link
End (Redirect to Login)
Create Account in Database
Send Confirmation Email
Display Success Message & Verification Instructions
End (Registration Complete)