# Overview of IT Industry

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

# python

print("Hello, World!")

# c

#include <stdio.h>

int main() {

printf("Hello, World!\n");

return 0;

}

# Comparison:

| **Feature** | **Python** | **C** |
| --- | --- | --- |
| **Verbosity** | Very concise (1 line) | More verbose (requires headers, main, etc.) |
| **Setup** | No setup required | Requires function and header declarations |
| **Output** | Uses print() | Uses printf() |
| **Syntax** | Minimal syntax | Uses semicolons, braces, and data types |
| **Compiled/Interpreted** | Interpreted | Compiled |
| **Ease of use** | Beginner-friendly | More complex, closer to hardware |

# *World Wide Web & How Internet Works*

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

Client Application

│

▼

Application Layer (HTTP, etc.)

│

Transport Layer (TCP)

│

Network / Internet Layer (IP)

│

Link / Data Link + Physical Layer

│

[Through Routers / ISPs / Internet Backbone]

│

Link / Data Link + Physical Layer at Server side

│

Network / Internet Layer (IP)

│

Transport Layer (TCP)

│

Application Layer (on Server)

│

Server Application

# *Network Layers on Client and Server*

# *Design a simple HTTP client-server communication in any language.*

// server.c

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <netinet/in.h>

#define PORT 8080

int main() {

int server\_fd, new\_socket;

struct sockaddr\_in address;

char buffer[3000] = {0};

char \*response =

"HTTP/1.1 200 OK\r\n"

"Content-Type: text/plain\r\n"

"Content-Length: 20\r\n"

"\r\n"

"Hello from C Server!";

int opt = 1;

int addrlen = sizeof(address);

// 1. Create socket

server\_fd = socket(AF\_INET, SOCK\_STREAM, 0);

// 2. Set socket options

setsockopt(server\_fd, SOL\_SOCKET, SO\_REUSEADDR, &opt, sizeof(opt));

// 3. Bind

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = INADDR\_ANY;

address.sin\_port = htons(PORT);

bind(server\_fd, (struct sockaddr \*)&address, sizeof(address));

// 4. Listen

listen(server\_fd, 3);

printf("Server is running on port %d...\n", PORT);

// 5. Accept and respond

new\_socket = accept(server\_fd, (struct sockaddr \*)&address, (socklen\_t\*)&addrlen);

read(new\_socket, buffer, sizeof(buffer));

printf("Received request:\n%s\n", buffer);

write(new\_socket, response, strlen(response));

close(new\_socket);

return 0;

}

# *6.Types of Internet Connections*

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

**📡 1. Fiber Optic Internet**

Uses light signals over fiber optic cables for extremely high-speed data transfer.

**✅ Pros:**

* **Very high speeds** (up to 1 Gbps or more)
* **Low latency** – great for gaming, streaming, video calls
* **Highly reliable** – less affected by weather or distance

**❌ Cons:**

* **Availability** is limited (mainly urban/suburban areas)
* **Installation cost** can be high in rural zones

**🧵 2. DSL (Digital Subscriber Line) – A type of broadband**

Uses regular telephone lines but with different frequencies for internet.

**✅ Pros:**

* Widely available (uses existing phone lines)
* **Dedicated line** – not shared with neighbors
* **Affordable**

**❌ Cons:**

* **Slower speeds** than cable/fiber
* Speed drops with **distance from provider’s hub**
* Limited capacity for streaming or gaming

**📺 3. Cable Internet**

Uses the same coaxial cables as cable TV.

**✅ Pros:**

* **Faster than DSL** (up to 1 Gbps in some areas)
* **Widely available** in urban/suburban areas

**❌ Cons:**

* **Shared bandwidth** – speed may drop during peak hours
* Higher latency than fiber
* Performance can vary depending on infrastructure quality

**🛰️ 4. Satellite Internet**

Beams internet data from a satellite orbiting Earth to a dish on your home.

**✅ Pros:**

* **Available anywhere**, including remote/rural areas
* Useful in places with no wired options

**❌ Cons:**

* **High latency** – signal travels 22,000+ miles
* **Slower speeds** than fiber/cable
* Affected by **weather and obstructions**
* **Data caps** often apply

*Example providers: Starlink, Viasat, HughesNet*

**📶 5. Mobile (4G / 5G) Internet**

Wireless internet via mobile networks using cellular towers.

**✅ Pros:**

* **Portable and wireless**
* **Fast** with 5G (can exceed 100 Mbps or more)
* Useful as backup internet

**❌ Cons:**

* **Signal strength varies** by location
* **Data caps** or throttling common
* May be affected by **network congestion**

**📞 6. Dial-Up**

Connects via a standard telephone line and a modem; now mostly obsolete.

**✅ Pros:**

* **Very cheap**
* **Available anywhere** with a phone line

**❌ Cons:**

* **Extremely slow** (~56 Kbps)
* Cannot use phone and internet simultaneously
* Not suitable for modern web usage

# 7. *Protocols*

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

# 1. **1. HTTP GET Request in C (with libcurl)**

**// http\_get.c**

**#include <stdio.h>**

**#include <curl/curl.h>**

**size\_t write\_callback(void \*ptr, size\_t size, size\_t nmemb, void \*userdata) {**

**fwrite(ptr, size, nmemb, stdout);**

**return size \* nmemb;**

**}**

**int main(void) {**

**CURL \*curl = curl\_easy\_init();**

**if (curl) {**

**curl\_easy\_setopt(curl, CURLOPT\_URL, "http://example.com");**

**curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, write\_callback);**

**curl\_easy\_perform(curl); // perform the request**

**curl\_easy\_cleanup(curl); // clean up**

**}**

**return 0;**

**}**

# **FTP File Download in C (with libcurl)**

// ftp\_download.c

#include <stdio.h>

#include <curl/curl.h>

int main(void) {

CURL \*curl = curl\_easy\_init();

if (curl) {

FILE \*fp = fopen("downloaded.txt", "wb");

curl\_easy\_setopt(curl, CURLOPT\_URL, "ftp://speedtest.tele2.net/1KB.zip");

curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, NULL);

curl\_easy\_setopt(curl, CURLOPT\_WRITEDATA, fp);

curl\_easy\_perform(curl);

curl\_easy\_cleanup(curl);

fclose(fp);

}

return 0;

# **8.** *Application Security*

# *Identify and explain three common application security vulnerabilities. Suggest possible solutions.*

# SQL INJECTION(SQLI)

* SQL Injection occurs when an attacker inserts malicious SQL code into a query, often through a user input field. If the input isn't properly validated or sanitized, the attacker can manipulate the database.

**✅ Solution:**

* **Use Prepared Statements / Parameterized Queries**  
  e.g., in Python (with SQLite):
* **Sanitize and validate user inputs**
* **Use ORM libraries** (e.g., SQLAlchemy, Django ORM)

# **Cross-Site Scripting (XSS)**

XSS occurs when an attacker injects malicious scripts (usually JavaScript) into web pages viewed by other users.

**✅ Solution:**

* **Escape all user input** before rendering it in HTML.
* Use secure frameworks with **automatic output encoding** (e.g., React, Angular).
* Apply **Content Security Policy (CSP)** headers to limit script execution.

# **Broken Authentication**

Weak or flawed authentication systems can allow attackers to compromise accounts or impersonate users.

**✅ Solution:**

* **Enforce strong password policies**
* **Implement multi-factor authentication (MFA)**
* **Use secure session management:**
  + Regenerate session IDs after login
  + Use HttpOnly and Secure flags on cookies
* **Rate-limit login attempts** to prevent brute-force attacks

# 9. *Software Applications and Its Types*

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

**Definitions :**

* **System Software**: Helps run the computer hardware and provides a platform for other software. Examples: operating systems, device drivers, utilities.
* **Application Software**: Designed to help users perform specific tasks. Examples: browsers, word processors, games, media players.

| **Application Name** | **Type of Software** | **Classification** | **Explanation** |
| --- | --- | --- | --- |
| **Google Chrome** | Web Browser | Application Software | Used to browse the internet; designed for end-user tasks. |
| **Windows 10** | Operating System | System Software | Manages computer hardware and software; essential for running other applications. |
| **Microsoft Word** | Word Processor | Application Software | Helps users create and edit documents; serves a specific user function. |
| **Windows Defender** | Security Tool | System Software | Runs in the background to protect the system; helps manage system security. |
| **Spotify** | Music Streaming App | Application Software | Allows users to stream music; not essential for system operation. |

# 10. *Software Architecture*

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



# 11. *Layers in Software Architecture*

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

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

**Functionality:**

* Acts as the front-end interface for users to interact with the system.
* Displays product catalogs, shopping carts, and order forms.
* Collects user input, such as search queries, product selections, and payment details.
* Sends user requests to the Business Logic Layer and displays processed results or error messages.

**Example Features:**

* Product listing pages with images and descriptions.
* Interactive shopping cart with add/remove items.
* Checkout forms to input shipping and payment information.

**Technologies:**

* HTML, CSS, JavaScript frameworks (React, Angular)
* Responsive design for accessibility on various devices

**2. Business Logic Layer (Application Layer)**

**Functionality:**

* Processes all business rules and workflows.
* Validates user inputs received from the Presentation Layer.
* Manages shopping cart operations like calculating totals, applying discounts, and taxes.
* Handles user authentication and authorization.
* Coordinates order processing, payment validation, and inventory updates.
* Serves as an intermediary between Presentation Layer and Data Access Layer.

**Example Features:**

* Check product availability before adding to cart.
* Apply discount codes based on promotion rules.
* Validate payment details and handle transaction logic.
* Manage user sessions and profiles.

**Technologies:**

* Server-side languages (Java, C#, Python)
* Frameworks like Spring Boot, ASP.NET, Django

**3. Data Access Layer (Persistence Layer)**

**Functionality:**

* Responsible for all interactions with the database.
* Performs CRUD (Create, Read, Update, Delete) operations.
* Ensures data integrity and consistency.
* Abstracts database queries and connection management from other layers.
* Optimizes data retrieval with indexing, caching where necessary.

**Example Features:**

* Store and retrieve product details, user profiles, and order history.
* Update inventory counts when orders are placed.
* Save payment transaction records securely.

**Technologies:**

* Relational Databases: MySQL, PostgreSQL
* NoSQL Databases: MongoDB
* ORM Tools: Hibernate, Entity Framework

# 12. *Software Environments*

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

**1. Development Environment**

* **Purpose:** Where developers write and debug code.
* **Characteristics:**
  + Contains tools like IDEs, debuggers, and version control.
  + Frequently updated with new code.
  + Less strict on performance and security.
  + May have mock data or limited datasets.

**2. Testing (or QA) Environment**

* **Purpose:** To test new features and fixes before release.
* **Characteristics:**
  + Mimics the production environment as closely as possible.
  + Used by QA engineers to find bugs or issues.
  + Automated and manual testing performed here.
  + Data often sanitized or anonymized.

**3. Production Environment**

* **Purpose:** The live environment where the software is used by end-users.
* **Characteristics:**
  + High stability, performance, and security requirements.
  + Minimal downtime allowed.
  + Real user data.
  + Strict access controls.

# 13. *Source Code*

# *Write your first source code file to Github.*

#include <stdio.h>

int main() {

printf("Hello, World!\n");

return 0;

}

# 16. *Types of Software*

# *Create a list of software you use regularly and classify them into the following categories: system, application, and utility software.*

**Definitions:**

* **System Software:** Software that manages the computer hardware and provides a platform for running application software.
* **Application Software:** Software designed for end-users to perform specific tasks.
* **Utility Software:** Programs that perform maintenance and general-purpose tasks to help manage the system.

**List of Software and Their Classification**

| **Software Name** | **Category** | **Explanation** |
| --- | --- | --- |
| **Windows 10** | System Software | Operating system that manages hardware and system resources. |
| **Google Chrome** | Application Software | Web browser used to access websites and web apps. |
| **Microsoft Word** | Application Software | Word processor for creating and editing documents. |
| **File Explorer** | Utility Software | Tool to manage files and folders on the computer. |
| **Windows Defender** | Utility Software | Antivirus and security tool protecting the system. |

# 18. *Application Software*

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

Introduction

Application software refers to programs designed to perform specific tasks for users, enhancing their ability to complete work efficiently. These software types cater to diverse needs, from document creation to communication, data management, and creative work. This report explores several key types of application software and explains how they improve productivity in both personal and professional contexts.

Types of Application Software

1. Word Processing Software

Examples: Microsoft Word, Google Docs, LibreOffice Writer

Function: Allows users to create, edit, format, and print text documents.

Productivity Impact:

Enables fast and flexible document creation, editing, and sharing. Features like spell check, templates, and collaboration tools reduce errors and save time.

2. Spreadsheet Software

Examples: Microsoft Excel, Google Sheets, LibreOffice Calc

Function: Organizes data in rows and columns, performs calculations, and creates charts.

Productivity Impact:

Automates complex calculations and data analysis, helping users make informed decisions quickly. Pivot tables and formulas streamline data management.

3. Presentation Software

Examples: Microsoft PowerPoint, Google Slides, Keynote

Function: Creates slide-based presentations to communicate ideas visually.

Productivity Impact:

Facilitates effective communication and persuasion in meetings and training. Templates and multimedia integration save preparation time and enhance audience engagement.

4. Database Management Software

Examples: Oracle, MySQL, Microsoft Access

Function: Stores, organizes, and manages large volumes of data efficiently.

Productivity Impact:

Enables quick data retrieval, reduces redundancy, and supports multi-user environments, improving accuracy and collaboration.

5. Communication Software

Examples: Microsoft Teams, Slack, Zoom, Outlook

Function: Facilitates messaging, video conferencing, email, and collaboration.

Productivity Impact:

Enhances real-time communication and remote collaboration, reducing delays and improving team coordination.

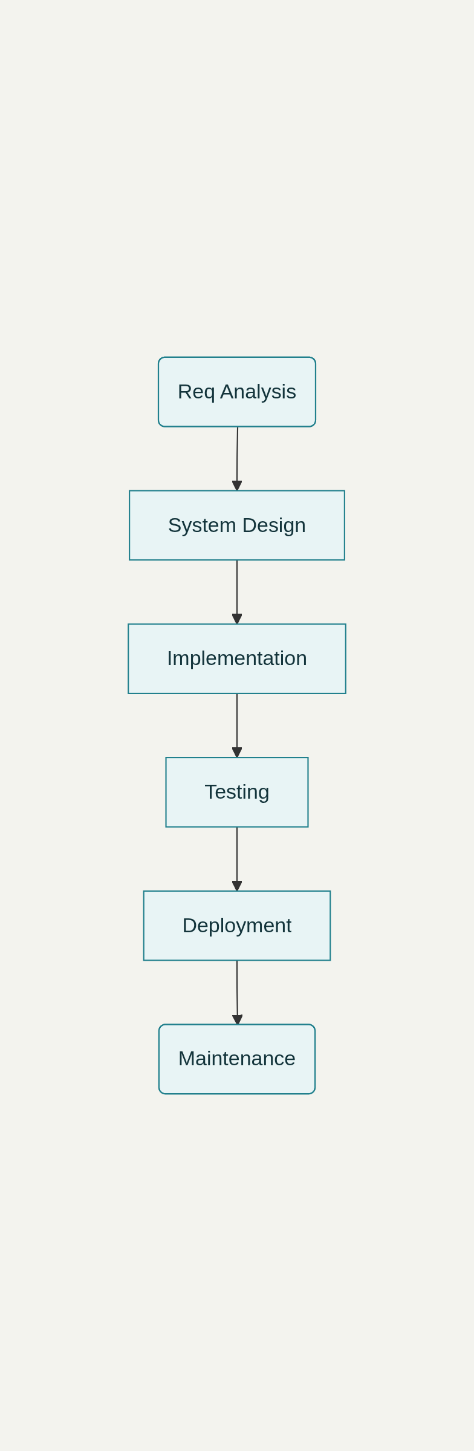
6. Graphic Design and Multimedia Software

Examples: Adobe Photoshop, Canva, Final Cut Pro

Function: Creates and edits images, videos, and animations.

# 19. *Software Development Process*

# *Create a flowchart representing the Software Development Life Cycle (SDLC).*



# 20. *Software Requirement*

# *Write a requirement specification for a simple library management system.*

**Library Management System Requirement Specification**

**1. Introduction**

The Library Management System (LMS) will facilitate efficient management of library resources, including book cataloging, membership management, borrowing, and returning of books. The system aims to streamline library operations and improve user access to resources.

**2. Functional Requirements**

* User Registration and Management
  + Members can register and create an account.
  + Librarians can add, update, or delete member records.
* Book Catalog Management
  + Librarians can add new books with details like title, author, ISBN, genre, and number of copies.
  + Update or remove existing book records.
* Book Search
  + Users can search for books by title, author, genre, or ISBN.
* Borrowing and Returning Books
  + Members can borrow available books for a fixed lending period.
  + System tracks due dates and borrowing history.
  + Members can return books, updating inventory accordingly.
  + Automatic notification for overdue books (optional).
* Inventory Management
  + Real-time tracking of book availability.
  + System prevents borrowing if no copies are available.
* Reports
  + Generate reports on borrowed and overdue books.
  + Summary of popular books based on borrowing frequency.

**3. Non-Functional Requirements**

* Usability: The system should have a user-friendly interface for both members and librarians.
* Reliability: Ensure accurate tracking of book inventory and member transactions.
* Performance: System should handle multiple users concurrently without performance degradation.
* Security: Protect member data and restrict librarian functionalities to authorized personnel.
* Maintainability: Easy to update and add new features like notifications or fine management.

**4. User Roles**

* Members: Can register, search books, borrow, and return books.
* Librarians: Manage books, members, and generate reports.

**5. Constraints**

* System operates on desktop and web platforms.
* Internet connection required for multi-user synchronization.

# 21. *Software Analysis*

# *Perform a functional analysis for an online shopping system.*

**Functional Analysis: Online Shopping System**

**1. User Registration and Authentication**

* Users can create accounts with personal information.
* Secure login/logout functionality.
* Password recovery and account management features.

**2. Product Catalog Management**

* Display product listings with images, descriptions, prices, and availability.
* Categories and filters for product search and browsing.
* Admin functionality to add, update, or remove products.

**3. Search and Navigation**

* Keyword-based product search.
* Advanced filters (price range, brand, ratings).
* Easy navigation through categories and promotions.

**4. Shopping Cart Management**

* Users can add or remove products to/from their cart.
* Update quantities or save items for later.
* Display cart summary with total cost and applicable taxes.

**5. Order Processing**

* Order placement with selected payment methods (credit card, digital wallets, COD).
* Order confirmation messages and emails.
* Order history and tracking capabilities for users.

**6. Payment Processing**

* Secure payment gateway integration.
* Multiple payment options supported.
* Handle payment errors and provide user feedback.

**7. Shipping and Delivery**

* Users provide shipping address during checkout.
* Real-time shipping cost calculation and delivery date estimates.
* Tracking number provided post-dispatch.

**8. Customer Support**

* Contact forms or live chat for customer queries.
* FAQs and help section integrated within the system.

**9. Reviews and Ratings**

* Users can rate and review purchased products.
* Moderation and display of reviews to assist purchase decisions.

**10. Administration and Reporting**

* Admin dashboard for monitoring sales, orders, and customer data.
* Tools for managing inventory, promotions, and user accounts.
* Generate reports on sales trends, product performance, and customer behavior.

# 22. *System Design*

# *Design a basic system architecture for a food delivery app.*

**Overview**

The food delivery app connects three main user types: **Customers**, **Restaurants**, and **Delivery Partners**. The architecture is typically **client-server based**, often using a **three-tier model**:

1. **Presentation Layer (Client Apps)**
2. **Application Layer (Backend / Business Logic)**
3. **Data Layer (Database & Storage)**

**Components & Flow**

**1. Presentation Layer (Client Apps)**

* **Customer App:** Interface for users to browse restaurants, place orders, track deliveries, and make payments.
* **Restaurant App:** Interface for restaurants to receive orders, update menu, and manage order status.
* **Delivery Partner App:** Interface for delivery agents to receive delivery requests and update delivery status.

**2. Application Layer (Backend Server)**

* **API Server:** Handles requests from all client apps and routes them appropriately.
* **Order Management:** Processes order placement, updates, cancellations.
* **Payment Processing:** Integrates with payment gateways for secure transactions.
* **Notification Service:** Sends push notifications or SMS to users about order status.
* **Authentication & Authorization:** Manages user login, registration, and roles.
* **Real-time Tracking:** Tracks delivery partners' locations and order progress.
* **Recommendation Engine (Optional):** Suggests restaurants or dishes based on user preferences.

**3. Data Layer**

* **Database:** Stores user data, restaurant info, menu items, orders, payments, and delivery details.
* **Cache:** For frequently accessed data like restaurant menus to improve response times.
* **File Storage:** For images (restaurant logos, dish photos).

# 32. flowchart

# *Draw a flowchart representing the logic of a basic online registration system.*

