**MODULE:1 :OVERVIEW OF IT INDUSTRY**

**LAB EXERCISE**

**\*Question : 01 :Write a simple “Hello world” program in two different programming languages of your choice. Compare the structure and syntax.**

**Answer 1: here are two languages we are programming Python and C languages.**

**Python : print(“Hello, world”)**

**Structure and syntax : 1)Minimal: one line of code. (2)No need for setup: No explicit function or header declarations. (3)Interpreted language: runs directly without compilation.**

**C language :**

**#include<stdio.h>**

**Main() {**

**Printf(“Hello, world!!\n”);**

**}**

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

**Answer 2:**

**1.Client(Device/browser):role:The starting point where user sends a request, such as typing URL in a browser or clicking link.**

**Process: The user sends HTTP/HTTPS request containing URL and relevant headers. The device prepares the data packet with the destination domain name.**

**2.Router:Role: The router is the first network device that directs the client data packets towards its destination.**

**Process: 1)The router checks its routing table to determine the best path for the packet.2)The packet is sent to the ISP’s network for further processing.**

**3.ISP(Internet Service Provider): Role: The ISP provides internet access and acts as an intermediary between the client and the broader internet.**

**Process:1)The ISP receives the packet and checks if it has a cached DNS record for the request domain. 2)If the DNS record is not cached, it forwards the request to a DNS server. Here cached mean storing data temporary in the system when we send request system checks that for particular website ip address already saved or not if saved then it does not go to DNS(Domain Name Server) because it already know IP address so directly it goes to IP address. And if IP address of that website is not known then first it goes for it to DNS then come back with IP address. And then go to that address.**

**4.DNS Server :Role:Translates the domain name(**[**www.google.com**](http://www.google.com)**) into its corresponding IP address eg(142.250.190.14)**

**Process:1)The client sends a query to the DNS Server. 2)The DNS server responds with the Ip address of the destination server.3)The Ip address is then included in the data packet for the routing.**

**5.Internet Backbone: Role: The global network of high-speed routers and fiber-optics cables that transmits data across the internet.**

**Process:1)The ISP forwards the packets to the internet backbone.**

**2)The packet traverses multiple networks and routers to reach the destination servers network**

**6.Destination Server: Role:The server that hosts the requested resource (eg:webpage, file or API)**

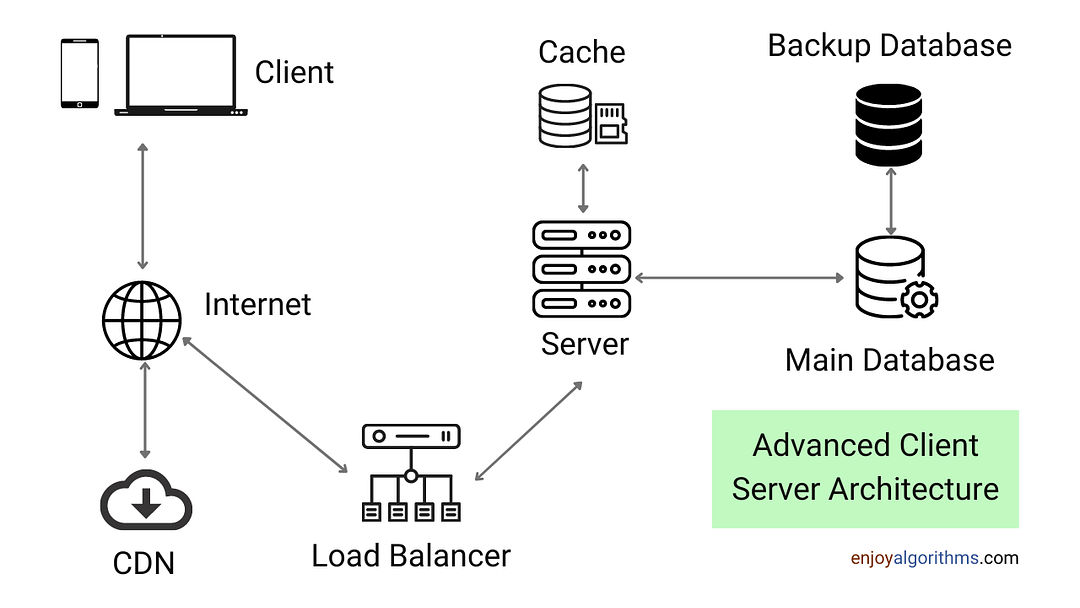
**Process:1)The servers receives the request process it, and retrieves the requested resource (2) It prepares a response (eg:HTML content )**

**And sends it back to the client**

**7)Response Delivery:Role: The return flow of data from server to the client.**

**Process:1)The Server’s response packet travels back through the same path : Internet backbone 🡪 ISP 🡪 Router 🡪 Client.**

**2)The browser or application on the client devices processes the response and displays the content(rendering a webpage)**

****

* **(4) Research different types of internet connections (e.g., broadband, fiber, satellite)and list their pros and cons?**
* **Answer 4 : 1)Broadband (DSL and Cable) :Description :Broadband refers to high-speed internet delivered through telephone lines(DSL) or coaxial cables (Cable)**

**Pros : 1)widely available in urban and suburban areas.**

**2)Relatively affordable compared to other options.**

**3)Reliable for streaming, browsing and light gaming.**

**Cons: 1)Slower speeds compared to fiber (2) Speed can decrease during peak hours (Cable) (3) DSL is distance sensitive :Far from the provider, slower is the speed;**

**2)Fiber Optic : Description : Fiber internet uses optical fibers to transmit data at the speed of light**

**Pros:1)Extremely high speed 2)low latency excellent for gaming and video conferencing 3)high reliability, even during peak usage times.**

**Cons : 1) limited availability, especially in rural areas. (2)Higher installation and subscription cost.**

**3.Satelite : Description :Internet provided through communication satelites, ideal for remote or rural areas**

**Pros: Available almost everywhere including rural and remote locations. 2)No dependency on physical cables or infrastructure.**

**Cons: 1)High latency due to distance signals must travel. 2)Limited data plans with high costs 3)susceptible to weather disruptions.**

**4)Mobile Data(3G/4G/5G) :Description :wireless internet provided via cellular networks, accessible on smartphones or through mobile hotspots.**

**Pros: portable and accessible wherever there is network coverage.**

**2)increasingly high speed 4G and 5G. 3)No need for physical connection.**

**Cons: 1)Data caps and throttling after exceeding limits 2)can be more expensive than fixed line connection.**

**5)Dial-up : Description : The earliest form of internet connection, using telephone lines.**

**Pros: 1)Extremely affordable 2)available whenever phone lines exist**

**Cons: 1)very low speed**

**6)Fixed wireless : Description : Uses radio signals to deliver internet to homes, often via a direct line-of-sight connection to a base station. Pros: 1)Good option for rural or underserved areas. 2)Faster installation compared to fiber**

**Cons:1)Susceptible to weather conditions and physical obstructions.**

**7)Hotspot (Public and private):Description :Internet access provided through wifi hotspot either in public space or via mobile device**

**Pros : Convinient and protable 2)No installation required for public hotspot**

**Cons: 1)Limited speed and coverage 2)Public hotspots can pose security risks.**

* **(5)** **: Identify and explain three common application security vulnerabilities. Suggest possible solutions?**
* **Answer : 1.SQL Injection : Explanation : 1)SQL Injections occurs when an attackers manipulates an applications database query by injecting malicious SQL code. (2) for example using “OR” “1”=1 in a login forms username field can bypass authentication (3)This vulneribity allows attackers to view, modify or delete sensitive data.**
* **Solutions : 1)Input validation : validate and sanitize all user inputs to prevent unexpected characters. 2)Prepared statements : Use parameterized queries or ORM frameworks like SQLAlchemy or hibernate to ensure SQL queries precompiled**
* **2: Cross-Site Scripting (XSS):Explanation :XSS occurs when malicious scripts are injected into web pages viewed by other users.2)for example attacker could inject <script> alert(‘hacked’);</script> into comment field. 3)this can lead to session hijacking, credential theft, or redirection to malicious websites. Solution : 1)Input Escaping :Escape special characters in user input before displaying it(eg: &, <,>). 2)Content Security Policy (CSP) : Configure a CSP to restrict the execution of scripts from unstructured sources.**

**\* Question : 6:** **: Identify and classify 5 applications you use daily as either system software or application software?**

**\*Answer : 6:**

**1. Operating System (e.g., Windows, macOS, Linux)**

* **Classification: System Software**
* **Reason: It manages hardware resources and provides services for other software.**

**2. Web Browser (e.g., Google Chrome, Mozilla Firefox)**

* **Classification: Application Software**
* **Reason: It is used to access the internet and interact with web-based applications.**

**3. Word Processor (e.g., Microsoft Word, Google Docs)**

* **Classification: Application Software**
* **Reason: It is used to create, edit, and format text documents.**

**4. Antivirus Software (e.g., Norton, McAfee)**

* **Classification: System Software**
* **Reason: It helps maintain the health of the system by protecting it from malware and viruses.**

**5. Messaging App (e.g., WhatsApp, Slack)**

* **Classification: Application Software**
* **Reason: It is used for communication and collaboration, not for system-level functions.**

**\*Question 7 : : Design a basic three-tier software architecture diagram for a web application?**

**\*Answer 7: 1.Presentation Layer(client/browser): Represents user interfaces like desktop, tablet, and smartphone**

**Application Layer : Consists of web servers and application servers managing business logic and processing requests.**

**Data layer: Feature the databases server and retrieving application data.**

**\*Question 8: Create a case study on the functionality of the presentation, business logic, and data access layers of a given software system?**

**\*Answer 8 : Case study : Online E-commerce Application :**

**Presentation Layer: functionality : The presentation layer is responsible for the user interface and interaction. It includes the website or mobile app where customers can browse products view details and make purchases.**

**Example scenario : A user opens the e-commerce app and searches for wireless headphones. The app displays a list of relevant products with prices and reviews**

**Key Technologies: Frontend Frame works : React.js, Angular, or vue.js. API Consumed : RESTful APIs provided by the business logic layer**

**\*Question 9: Write a report on the various types of application software and how they improve productivity.**

**Answer 9: Introduction : Application software is class of computer programs designed to perform specific tasks for users. Unlike system software, which manages the hardware and operating system, application software directly helps users complete tasks efficiently thereby improving productivity. This report explores the different types of application software and their impact on enhancing productivity.**

**Types of Application Software**

**1. Office Productivity Software**

**Examples: Microsoft Office Suite (Word, Excel, PowerPoint), Google Workspace, LibreOffice.**

**Purpose:**

**Word Processors: Create, edit, and format textual documents (e.g., MS Word).**

**Spreadsheets: Analyze, organize, and visualize data using formulas and charts (e.g., MS Excel).**

**Presentation Tools: Design and present ideas visually (e.g., MS PowerPoint).**

**Impact:**

**Automates repetitive tasks like calculations and formatting.**

**Facilitates collaboration via real-time editing and sharing.**

**Saves time by enabling template-based document creation.**

**2. Enterprise Software**

**Examples: ERP systems (Odoo, SAP, Salesforce), CRM platforms.**

**Purpose:**

**Manage business processes like inventory, finance, and customer relationships.**

**Impact:**

**Streamlines operations through integration and automation.**

**Improves decision-making with data analytics and reporting tools.**

**Enhances customer service and satisfaction by tracking interactions.**

**3. Communication Software**

**Examples: Microsoft Teams, Zoom, Slack, WhatsApp.**

**Purpose:**

**Facilitate communication through text, voice, and video.**

**Impact:**

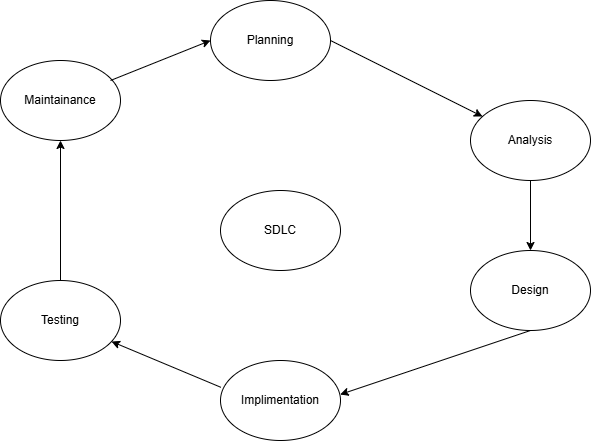
**Reduces communication delays with instant messaging and video conferencing.**

**Supports remote work and global collaboration.**

**Increases engagement through features like screen sharing and file transfer.**

**\*Question 10:Create a flow chart representing the software Development Life Cycle**

**\*Answer 10:**

****

**\*Question 11:** **Write a requirement specification for a simple library management system?**

**\*Answer 11:**

**1. Introduction**

This document outlines the functional and non-functional requirements for a simple Library Management System (LMS). The LMS will streamline library operations such as managing books, members, and transactions.

**2. Purpose**

**The system will:**

**Facilitate book management (addition, deletion, and search).**

**Enable member management (registration, update, and deletion).**

**Manage borrowing and returning of books.**

**Generate reports for better tracking and organization.**

**3. Functional Requirements**

* 1. **User Management**

**Admin Account:**

* + **Add, update, or delete user accounts.**
  + **Assign roles (e.g., librarian, admin).**

**Member Account:**

* + **Register members with details (name, ID, contact information).**
  + **Update member information.**
  + **Delete inactive member accounts.**

**3.2 Book Management**

**Add new books with details (title, author, ISBN, category, and quantity).**

* **Update book details (e.g., price, quantity).**
* **Delete records of books that are no longer available.**
* **Search books by title, author, category, or ISBN.**

**3.3 Borrowing and Returning**

* **Issue books to members:**
  + **Check book availability.**
  + **Record issue date and return due date.**
* **Process book returns:**
  + **Calculate late fees if applicable.**
  + **Update book stock.**

**3.4 Catalog Management**

* **Maintain an up-to-date book catalog for users to browse.**
* **Enable filtering and sorting of books (e.g., by author, genre, or availability).**

**3.5 Reporting**

* **Generate reports on:**
  + **Books borrowed or returned in a specified period.**
  + **Member borrowing history.**
  + **Books currently unavailable.**

**\*Question 12: Perform a functional analysis for an online shopping system.**

**\*Answer :**

**1. Introduction**

**An online shopping system allows customers to browse, search, and purchase products through a web or mobile interface. The system also supports sellers in managing product listings and orders. This analysis identifies the core and supplementary functions required for an efficient and user-friendly platform.**

**2. Functional Areas**

**2.1 User Management**

* **Customer Account Management:**
  + **Create accounts with email/phone-based registration.**
  + **Login/logout functionality.**
  + **Profile management (update personal details, change password).**
  + **Password recovery via email or SMS.**
* **Admin Account Management:**
  + **Add, update, or delete admin users.**
  + **Manage roles and permissions.**
* **Seller Account Management:**
  + **Register as a seller with business details.**
  + **Profile updates (e.g., business name, contact information).**

**2.2 Product Management**

* **For Sellers:**
  + **Add new products with details (title, description, price, category, images, stock).**
  + **Edit or delete product listings.**
  + **View product performance metrics (e.g., views, purchases).**
* **For Admins:**
  + **Approve or reject seller products.**
  + **Categorize and tag products.**

**2.3 Search and Browse**

* **Keyword-based search with autocomplete suggestions.**
* **Filters and sorting:**
  + **By price range, category, brand, ratings, and availability.**
* **Category-wise browsing with pagination.**
* **Display recommended products based on user behavior.**

**2.4 Shopping Cart and Wishlist**

* **Shopping Cart:**
  + **Add products to the cart.**
  + **Update quantity or remove items.**
  + **Calculate total price including applicable taxes and discounts.**
* **Wishlist:**
  + **Save products for future reference.**
  + **Move items from wishlist to cart.**

**2.5 Order Management**

* **For Customers:**
  + **Place orders with secure payment options.**
  + **Track orders with real-time status updates.**
  + **Cancel orders or request returns/refunds.**
* **For Sellers:**
  + **Manage incoming orders.**
  + **Update shipping status.**
  + **Handle refund and return requests.**

**2.6 Payment System**

* **Integration with multiple payment gateways (credit/debit cards, UPI, wallets).**
* **Option for Cash on Delivery (COD).**
* **Generate invoices for completed orders.**

**2.7 Notifications**

* **Email and SMS notifications for:**
  + **Order confirmations and updates.**
  + **Promotions and offers.**
  + **Delivery updates.**
* **Push notifications for mobile app users.**

**2.8 Review and Rating**

* **Allow customers to rate and review products.**
* **Display average ratings and user comments on product pages.**
* **Enable sellers to respond to reviews.**

**2.9 Analytics and Reporting**

* **For Admins:**
  + **View sales trends and user engagement metrics.**
  + **Monitor website traffic and popular products.**
* **For Sellers:**
  + **Analyze product performance and revenue.**
  + **Track inventory and restocking needs.**

**2.10 Security and Privacy**

* **Secure user authentication using encrypted passwords and multi-factor authentication.**
* **Protect sensitive data like payment information and personal details.**
* **Ensure compliance with data protection regulations (e.g., GDPR, CCPA).**

**3. Functional Workflow**

1. **Customer Interaction:**
   * **Browse products → Add to cart → Checkout → Payment → Order Confirmation → Delivery.**
2. **Seller Workflow:**
   * **Register → List products → Monitor sales → Fulfill orders → Manage returns.**
3. **Admin Oversight:**
   * **Approve sellers → Monitor transactions → Manage disputes → Analyze platform performance.**

**\*Question 13 : Design a basic system architecture for a food delivery app?**

**\*Answer 13:**

**Basic System Architecture for a Food Delivery App**

**A food delivery app integrates various components to connect customers, restaurants, and delivery partners seamlessly. Below is a modular system architecture that highlights the essential components and their interactions.**

**1. Core Components**

**1.1 User Interfaces**

* **Customer App:**
  + **Browse restaurants and menus.**
  + **Place orders, track delivery, and make payments.**
  + **View past orders and provide ratings/reviews.**
* **Restaurant App:**
  + **Manage menu items and availability.**
  + **Receive and confirm orders.**
  + **Update order preparation status.**
* **Delivery Partner App:**
  + **Accept delivery assignments.**
  + **View delivery route and customer details.**
  + **Update delivery status.**
* **Admin Dashboard:**
  + **Monitor platform activity and user accounts.**
  + **Manage disputes and support queries.**
  + **Generate reports and analytics.**

**1.2 Backend Services**

* **Authentication Service:**
  + **User login, signup, and authorization.**
  + **Token-based authentication for secure sessions.**
* **Order Management Service:**
  + **Handles order creation, updates, and tracking.**
  + **Manages communication between customers, restaurants, and delivery partners.**
* **Payment Gateway Integration:**
  + **Supports multiple payment methods (cards, wallets, UPI, COD).**
  + **Ensures secure and reliable transactions.**
* **Notification Service:**
  + **Sends order updates and promotional notifications via push, SMS, and email.**
* **Geolocation Service:**
  + **Maps integration for route optimization and location tracking.**
  + **Enables restaurant and delivery partner proximity detection.**

**1.3 Database Layer**

* **Relational Database (e.g., MySQL, PostgreSQL):**
  + **Stores structured data like user profiles, restaurant details, menu items, and orders.**
* **NoSQL Database (e.g., MongoDB, Firebase):**
  + **Handles unstructured or semi-structured data like user preferences and order history.**
* **Caching System (e.g., Redis, Memcached):**
  + **Accelerates frequently accessed data, such as popular menus or ongoing orders.**

**1.4 APIs and Middleware**

* **RESTful APIs or GraphQL:**
  + **Facilitates communication between frontend interfaces and backend services.**
  + **Exposes endpoints for order placement, menu retrieval, and status updates.**
* **Middleware:**
  + **Handles request validation, logging, and error handling.**

**1.5 Third-Party Integrations**

* **Payment Gateways: Stripe, PayPal, Razorpay.**
* **Mapping Services: Google Maps, Mapbox.**
* **Notification Services: Twilio, Firebase Cloud Messaging (FCM).**

**2. System Architecture Diagram**

**sql**

**Copy code**

**+----------------+ +----------------+**

**| Customer App | | Delivery App |**

**+----------------+ +----------------+**

**| |**

**| |**

**+----------------------------------------------+**

**| Backend Services |**

**| Authentication | Order Management |**

**| Payment Gateway | Notification Service |**

**+----------------------------------------------+**

**| |**

**+----------------+ +----------------+**

**| Restaurant App | | Admin Panel |**

**+----------------+ +----------------+**

**|**

**|**

**+-------------------------+**

**| Database Layer |**

**| Relational, NoSQL, |**

**| Caching Systems |**

**+-------------------------+**

**3. Data Flow**

1. **Customer Places Order:**
   * **The customer app sends a request to the backend with order details.**
   * **The backend validates the request, reserves the order in the database, and notifies the restaurant.**
2. **Restaurant Prepares Order:**
   * **The restaurant confirms the order via the app.**
   * **Updates preparation status for the customer and delivery partner.**
3. **Delivery Assignment:**
   * **The backend uses geolocation services to find the nearest delivery partner.**
   * **Assigns the task and shares order details.**
4. **Order Delivery:**
   * **The delivery partner updates the delivery status in real time.**
   * **The customer receives live updates until the order is delivered.**
5. **Post-Delivery:**
   * **The customer can rate and review the order.**
   * **Payment details are updated, and revenue is allocated to the restaurant and delivery partner.**

**4. Technologies**

**Frontend**

* **Frameworks: React Native, Flutter (cross-platform apps).**
* **Libraries: Axios (API requests), Redux (state management).**

**Backend**

* **Languages: Node.js, Python (Django/Flask), Java (Spring Boot).**
* **Frameworks: Express.js (Node.js), FastAPI (Python).**

**Database**

* **Relational: PostgreSQL, MySQL.**
* **NoSQL: MongoDB, DynamoDB.**

**Third-Party Services**

* **Maps: Google Maps API.**
* **Notifications: Firebase Cloud Messaging.**
* **Payments: Razorpay, Stripe.**

**5. Scalability and Future Enhancements**

* **Implement microservices for modularity.**
* **Add AI-driven recommendation systems for personalized menus.**
* **Incorporate chatbots for customer support.**
* **Support multi-country operations with currency and language adaptability.**

**This architecture serves as a blueprint for developing a scalable and efficient food delivery platform. Let me know if you'd like to dive deeper into any specific component!**

**\*Question 14: Develop test cases for a simple calculator program?**

**\*Answer:**

**1. Test Case: Addition**

* **Input: 5 + 3**
* **Expected Output: 8**
* **Description: Verify the calculator correctly adds two positive integers.**
* **Input: -5 + (-3)**
* **Expected Output: -8**
* **Description: Verify the calculator correctly adds two negative integers.**
* **Input: 5.5 + 3.2**
* **Expected Output: 8.7**
* **Description: Verify the calculator handles addition of floating-point numbers.**

**2. Test Case: Subtraction**

* **Input: 10 - 4**
* **Expected Output: 6**
* **Description: Verify subtraction of two positive integers.**
* **Input: -10 - (-5)**
* **Expected Output: -5**
* **Description: Verify subtraction involving two negative integers.**
* **Input: 5.5 - 2.2**
* **Expected Output: 3.3**
* **Description: Verify subtraction of floating-point numbers.**

**3. Test Case: Multiplication**

* **Input: 4 \* 3**
* **Expected Output: 12**
* **Description: Verify multiplication of two positive integers.**
* **Input: -4 \* 3**
* **Expected Output: -12**
* **Description: Verify multiplication of a positive and a negative integer.**
* **Input: 2.5 \* 4**
* **Expected Output: 10.0**
* **Description: Verify multiplication of a floating-point number and an integer.**

**4. Test Case: Division**

* **Input: 10 / 2**
* **Expected Output: 5**
* **Description: Verify division of two positive integers.**
* **Input: -10 / 2**
* **Expected Output: -5**
* **Description: Verify division of a negative integer by a positive integer.**
* **Input: 5 / 0**
* **Expected Output: Error message (e.g., "Division by zero is not allowed")**
* **Description: Verify error handling for division by zero.**

**5. Test Case: Edge Cases**

* **Input: 0 + 0**
* **Expected Output: 0**
* **Description: Verify the calculator handles addition of zero values.**
* **Input: 0 \* 5**
* **Expected Output: 0**
* **Description: Verify the calculator handles multiplication by zero.**
* **Input: 0 / 5**
* **Expected Output: 0**
* **Description: Verify the calculator handles division of zero by a positive number.**
* **Input: 9999999999 + 1**
* **Expected Output: 10000000000**
* **Description: Verify the calculator handles large integers without overflow.**

**6. Test Case: Invalid Input**

* **Input: "5 + abc"**
* **Expected Output: Error message (e.g., "Invalid input")**
* **Description: Verify the calculator handles non-numeric inputs gracefully.**
* **Input: " "**
* **Expected Output: Error message (e.g., "Input cannot be empty")**
* **Description: Verify error handling for empty input.**

**\*Question 15:** **Document a real-world case where a software application required critical maintenance?**

**\*Answer 15:**

**Overview**

**In 2018 and 2019, two Boeing 737 MAX airplanes crashed, resulting in the tragic loss of 346 lives. The crashes were linked to a software flaw in the aircraft's Maneuvering Characteristics Augmentation System (MCAS), which required critical maintenance and updates.**

**Details of the Problem**

1. **Software Involved:**
   * **The MCAS was designed to adjust the plane's nose angle automatically to prevent stalls during specific flight conditions.**
2. **Issue:**
   * **The software relied on data from a single angle-of-attack (AoA) sensor.**
   * **If the sensor provided erroneous data, the MCAS repeatedly pushed the nose of the plane downward, despite pilot attempts to correct it.**
3. **Impact:**
   * **Inaccurate sensor readings caused the system to behave unpredictably, leading to loss of control.**
   * **The software’s failure was a critical factor in the Lion Air Flight 610 and Ethiopian Airlines Flight 302 crashes.**

**Critical Maintenance Required**

1. **Identification of Flaws:**
   * **Investigations revealed the MCAS system had inadequate redundancy and insufficient fail-safes.**
   * **Pilot training and documentation inadequately covered how to manage MCAS malfunctions.**
2. **Software Updates:**
   * **Boeing developed a software patch to:**
     + **Use data from multiple sensors instead of a single AoA sensor.**
     + **Limit the MCAS's authority to push the nose downward.**
     + **Prevent repeated activation of the system.**
3. **Regulatory Involvement:**
   * **The Federal Aviation Administration (FAA) grounded the Boeing 737 MAX until the software was fixed and re-certified.**
   * **The grounding lasted for nearly 20 months, with rigorous testing and validation.**
4. **Pilot Training Updates:**
   * **Revised training manuals and simulator programs were provided to ensure pilots understood how to handle MCAS malfunctions.**

**Outcome**

* **The updated software was implemented, and the 737 MAX returned to service in late 2020.**
* **The incident highlighted the importance of robust software testing, redundancy, and comprehensive pilot training in critical systems.**

**Lessons Learned**

* **Redundancy: Critical systems should not rely on single points of failure (e.g., one sensor).**
* **Documentation & Training: End-users (e.g., pilots) must be well-informed about how to handle system failures.**
* **Testing: Software in critical systems must undergo extensive real-world scenario testing.**
* **Regulatory Oversight: Proper regulatory checks and balances can prevent issues from escalating.**

**This case underscores the importance of critical software maintenance in ensuring safety and reliability in real-world applications.**

**\*Question 16:** **Create a DFD for a hospital management system.**

**\*Answer 16:**

**Level 0 (Context Diagram)**

**This represents the overall system and its interaction with external entities.**

**Entities:**

1. **Patients**
2. **Doctors**
3. **Admin**
4. **Pharmacy**
5. **Billing System**

**Processes:**

1. **Hospital Management System**

**Data Flows:**

* **Patients → HMS: Personal details, appointments, complaints.**
* **Doctors → HMS: Availability, prescriptions, patient updates.**
* **Admin → HMS: Staff management, patient records.**
* **Pharmacy → HMS: Medicine stock, prescription processing.**
* **HMS → Billing System: Billing details, patient charges.**

**Level 1 (Main Processes)**

**Break down the HMS into its major components:**

**Processes:**

1. **Patient Registration**
   * **Input: Patient details (name, age, contact, etc.).**
   * **Output: Patient ID.**
2. **Appointment Management**
   * **Input: Patient ID, requested doctor/specialty, appointment date.**
   * **Output: Appointment confirmation, schedule.**
3. **Doctor Consultation**
   * **Input: Patient ID, symptoms.**
   * **Output: Diagnosis, prescription.**
4. **Pharmacy Management**
   * **Input: Prescription.**
   * **Output: Medicines issued, stock update.**
5. **Billing Management**
   * **Input: Patient ID, services used.**
   * **Output: Bill generation, payment status.**

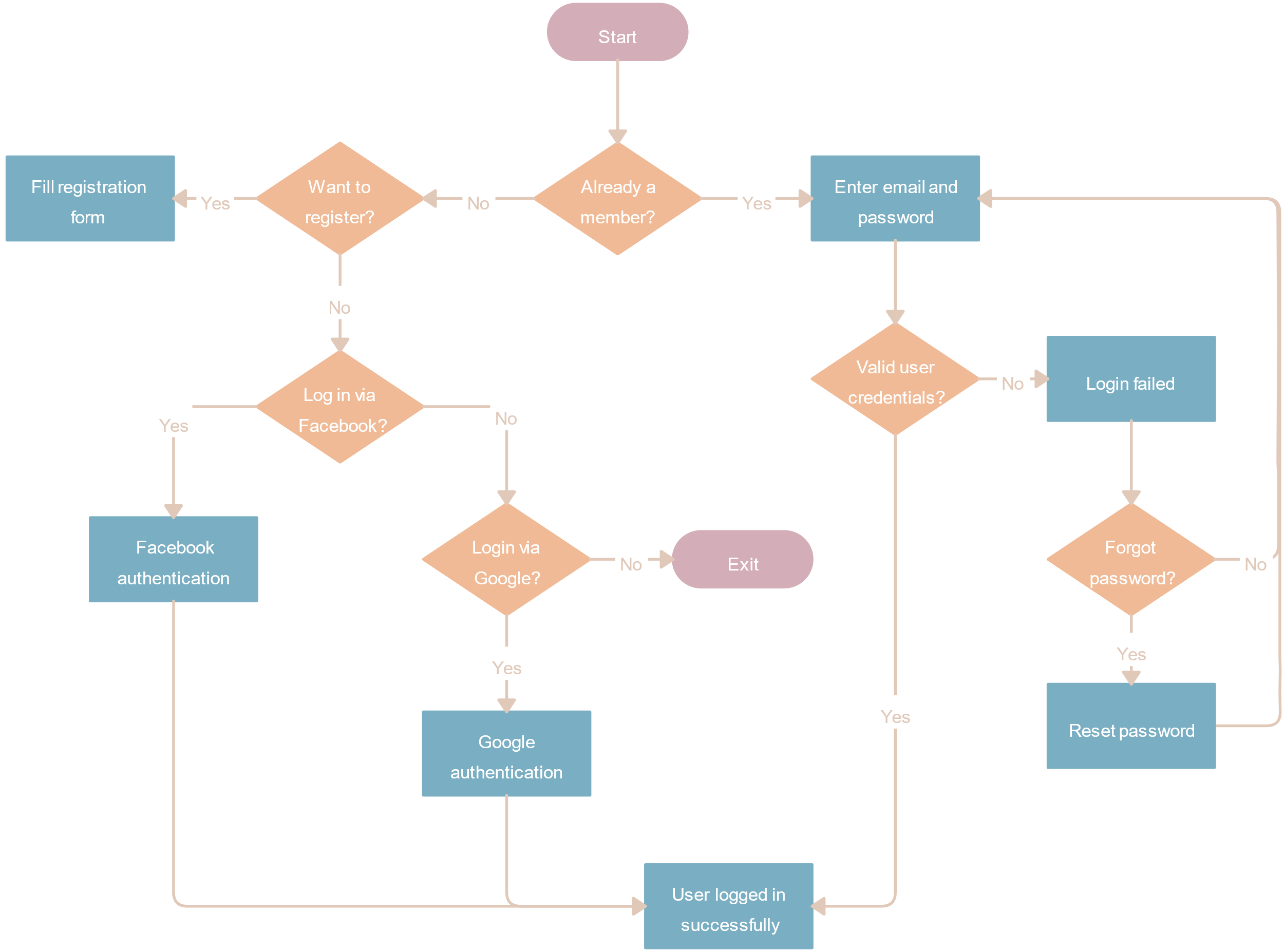
**DFD Level 1 Visual Description**

**Here’s how the data flows between entities and processes:**

1. **Patient Registration:**
   * **Patient → Registration Process → Patient Database.**
2. **Appointment Management:**
   * **Patient → Appointment Process → Doctor Schedule Database.**
3. **Doctor Consultation:**
   * **Patient & Doctor → Consultation Process → Patient Records.**
4. **Pharmacy Management:**
   * **Consultation Process → Pharmacy → Medicine Issued.**
5. **Billing Management:**
   * **Patient Records & Pharmacy → Billing Process → Payment Details.**

**\*Question 17 :** **: Draw a flowchart representing the logic of a basic online registration system.**

**\*Answer 17:**

****