# **[LAB ASSIGNMENT]**

* **[Lab exercise] : Write a simple "Hello World" program in two different programming languages of your choice. Compare the structure and syntax.**
* **C :**
* #include<stdio.h>

Void main()

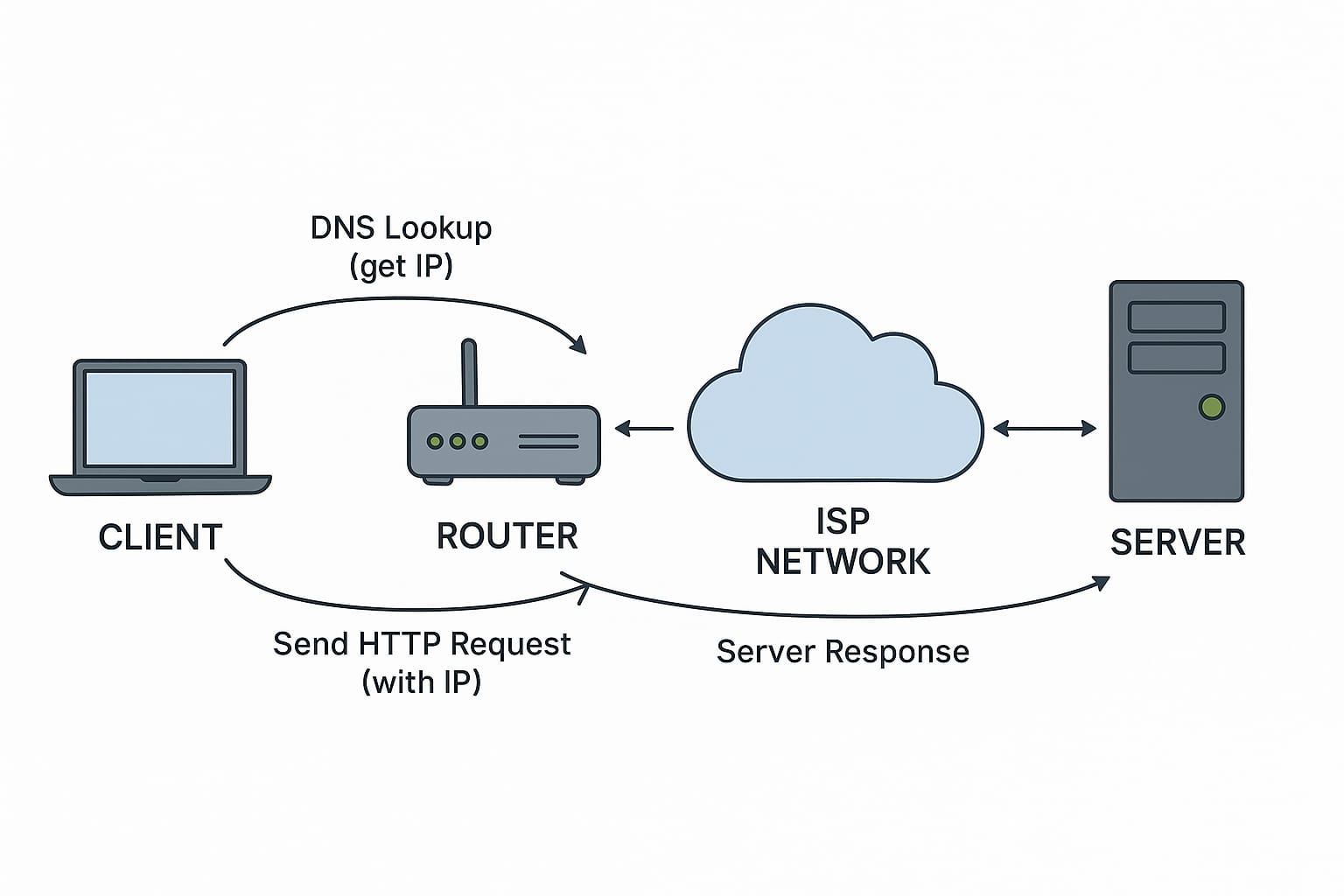
{

printf(“Hello World”);

getch();

}

* **PYTHON :**
* Print(“Hello World”)
* **Python** is concise, user-friendly, and great for quick scripts.
* **C** is more verbose, requiring explicit structure and compilation, reflecting its low-level control and performance orientation.

* **[LAB EXERCISE]: Research and create a diagram of how data is transmitted from a client to a server over the internet.**
* The diagram shows how multiple clients (like a laptop, phone, or desktop) connect to a server through the internet to send requests and receive responses, representing a basic client-server communication model.
* **[LAB EXERCISE]: Research different types of internet connections (e.g., broadband, fiber,satellite)and list their pros and cons.**

**1. Dial-Up Internet:**

: Pros:

• Very low cost

* + Available almost everywhere with a landline

Cons:

• Extremely slow

• Ties up the phone line

**2. DSL (Digital Subscriber Line)**

Pros:

• Available in many areas

• Doesn't interfere with phone calls

Cons:

• Speed decreases with distance from the provider

• Slower than cable or fiber

**3. Fiber Optic Internet**

Pros

• Ultra-fast speeds

• Highly reliable and low latency

Cons:

• Expensive installation

• Limited availability, especially in rural areas

**4. Satellite Internet**

Pros:

• Available in remote and rural locations

• No need for physical cables

Cons:

• Weather can affect signal

• Data caps and higher cost

**5. broadband**

Pros:

• High-speed internet

• Supports multiple devices

Cons:

• Can be expensive

• Some plans have data limits

* **[LAB EXERCISE]: Identify and explain three common application security vulnerabilities. Suggest possible solutions.**
* Three common application security vulnerabilities are SQL Injection, Cross-Site Scripting (XSS), and Broken Authentication.

1. SQL Injection

An attacker inserts harmful SQL code into a form input (like login fields) to access or manipulate the database.

Solution: Use prepared statements and validate input.

2. Cross-Site Scripting (XSS)

An attacker injects Malicious scripts are injected into web pages to attack users.

Solution: Sanitize input and escape output.

3. Broken Authentication

Poorly designed login systems allow attackers to gain unauthorized access to user accounts.

Solution: Use strong passwords, MFA, and secure sessions.

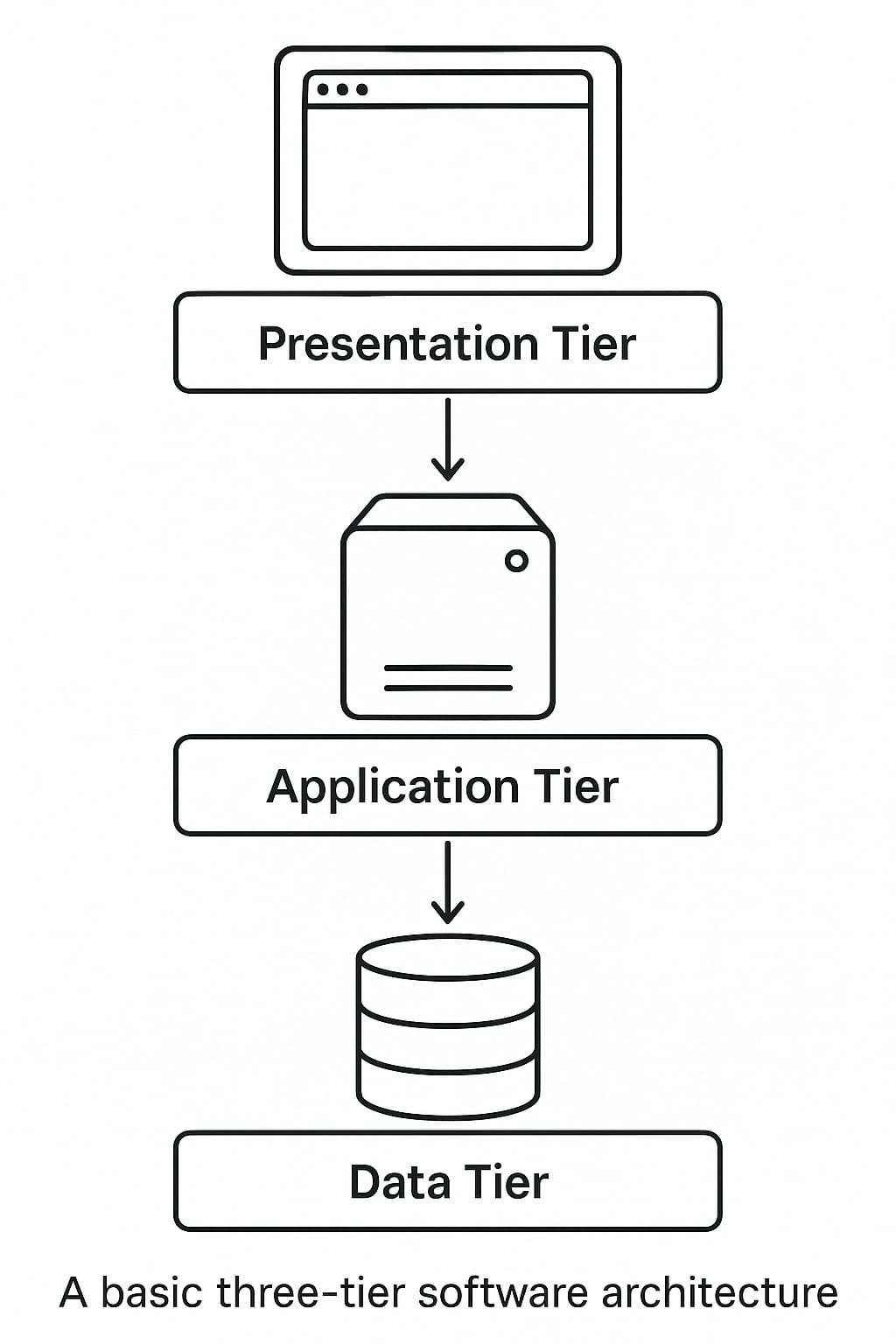
* **[LAB EXERCISE]: Identify and classify 5 applications you use daily as either system software Or application software**.

I use the following 5 applications daily:

1. **Google Chrome** – *Application Software* (for browsing the internet)
2. **Microsoft Word** – *Application Software* (for writing documents)
3. **WhatsApp** – *Application Software* (for messaging and calling)
4. **Windows OS** – *System Software* (runs the computer and manages hardware)
5. **Antivirus Software** – *System Software* (protects the system from threats)

These are classified based on whether they help users do tasks (application) or help the system run (system).

* **[LAB EXERCISE] : Design a basic three-tier software architecture diagram for a web application.**



* **Presentation Tier** – The user interface (e.g., web browser) where users interact with the application.
* **Application Tier** – The logic layer that processes user input, makes decisions, and handles operations.
* **Data Tier** – The database layer where all application data is stored and managed.
* **[LAB EXERCISE]: Create a case study on the functionality of the presentation, business logic, and Data access layers of a given software system.**

📚 **Case Study Example: Hospital Management System**

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

This is what the user sees: pages to book appointments, view doctor schedules, or access reports.

**Example:** A patient logs into the hospital portal to book an appointment with a doctor.

**2. Business Logic Layer (Processing Side)**

This checks the rules: Is the doctor available? Is the patient registered?

**Example:** The system checks if the doctor is free at the selected time and if the patient’s ID is valid.

**3. Data Access Layer (Database Side)**

This layer interacts with the database: stores appointments, retrieves patient records.

**Example:** The system saves the appointment details to the database and updates the doctor’s schedule.

* **[LAB EXERCISE]: Explore different types of software environments (development, testing,production).Set up a basic environment in a virtual machine.**
* **Different Types of Software Environments**

1. **Development Environment**
   * This is where programmers write and create the software.
   * It has tools like code editors and debuggers.
   * It’s flexible and changes often.
2. **Testing Environment**
   * After writing the software, it is moved here to check if everything works well.
   * Bugs are found and fixed in this environment.
   * It tries to copy the real-world setup but isn’t live.
3. **Production Environment**
   * This is the real, live environment where the software is used by customers.
   * It is stable and reliable.
   * Changes here are made carefully to avoid breaking things.

* **How to Set Up a Basic Environment in a Virtual Machine (VM)**

1. **Choose a Virtual Machine software:**  
   Examples: VirtualBox, VMware, or Hyper-V.
2. **Download an Operating System (OS):**  
   For example, get an ISO file of Ubuntu (Linux) or Windows.
3. **Create a New Virtual Machine:**
   * Open the VM software.
   * Create a new VM and assign memory (RAM), disk space, and CPU cores.
   * Attach the OS ISO file to the VM.
4. **Start the Virtual Machine:**
   * Boot up the VM with the OS installer.
   * Follow the steps to install the OS inside the VM.
5. **Set Up Your Environment:**
   * Install software like code editors (e.g., VS Code), programming languages, or tools needed.
   * This can be your **development environment** inside the VM.
6. **Use Snapshots:**
   * Save the VM state so you can return to it if needed (useful for testing).

* **[LAB EXERCISE]: Create a GitHub repository and document how to commit and push code changes.**

✅ 1. Create a Repository on GitHub

1. Go to <https://github.com> and log in.
2. Click the + icon → New repository.
3. Enter a repository name (e.g., my-project), choose Public or Private.
4. *(Optional)* Check "Initialize with a README".
5. Click Create repository.

**EXAMPLE :**

git status

git add .

git commit -m "Initial project files"

git push origin main

* **[LAB EXERCISE]: Create a student account on GitHub and collaborate on a small project with a classmate.**
* Create GitHub Student Account & Collaborate
* Sign up at github.com using your student email.
* Apply for Student Pack: education.github.com/pack
* Create a repo: Click "+" → "New repository"
* Go to Settings → Collaborators, add your classmate’s GitHub username.
* **LAB EXERCISE: Create a list of software you use regularly and classify them into the**

**Following categories: system, application, and utility software.**

Sure! Here’s an example list of software classified into **system**, **application**, and **utility software**:

**System Software**

* **Windows 11** (Operating System)

**Application Software**

* **Microsoft Word** (Word processor)
* **Google Chrome** (Web browser)
* **Adobe Photoshop** (Image editor)

**Utility Software**

* **Disk Cleanup** (Tool to free up space)

 **CCleaner** – System Cleanup

 **Backup Software** – Data Backup

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

|  |  |
| --- | --- |
| Task | Command Example |
| Clone | **git clone URL** |
| branch | **git checkout -b feature-name** |
| Merge | **git checkout main → git merge branch** |

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

• Types of Application Software and Their Productivity Benefits

1. **Word Processing Software**

o Example: Microsoft Word, Google Docs

o Use: Writing documents, letters, reports

o Productivity: Speeds up writing with tools like auto-correct, templates, and collaboration.

2. Spreadsheet Software

o Example: Microsoft Excel, Google Sheets

o Use: Data analysis, financial planning, reports

o Productivity: Automates calculations and data organization; useful in business and accounting.

3. **Presentation Software**

o Example: Microsoft PowerPoint, Prezi

o Use: Creating visual slides for meetings or lectures

o Productivity: Enhances communication of ideas clearly and quickly.

4. **Database Management Software**

o Example: Oracle, MS Access, MySQL

o Use: Storing and managing large datasets

o Productivity: Ensures fast data access and organization for decision-making.

5. **Multimedia Software**

o Example: VLC Media Player, Adobe Premiere Pro

o Use: Creating and editing audio, video, and images

o Productivity: Supports creative tasks and professional media production.

6. **Communication Software**

o Example: Zoom, Microsoft Teams, Slack

o Use: Messaging, video conferencing, file sharing

o Productivity: Improves team collaboration and reduces time spent on meetings.

7. **Web Browsers**

o Example: Chrome, Firefox

o Use: Internet access for research and online tools

o Productivity: Quick access to information, tools, and cloud services.

* **LAB EXERCISE: Perform a functional analysis for an online shopping system.**

**• Functional Analysis – Online Shopping System**

**1. User Roles:**

**• Customer/User**

**• Admin**

**•** (Optional: Vendor/Seller if multi-vendor platform**)**

**2. Functional Requirements:**

**Customer Functions:**

**• User Registration/Login –** Create and access user account

**• Browse Products –** View items by category, brand, etc.

**• Search Products –** Search by name, price, rating, etc.

**• Add to Cart –** Add selected items to shopping cart

**• Place Order –** Checkout and complete purchase

**• Payment –** Online payment via card, UPI, or COD

**• Order History –** View past orders and status

**• Product Review/Rating –** Leave feedback

**Admin Functions:**

**• Manage Users –** Add/edit/remove user accounts

**• Manage Products –** Add/edit/delete product listings

**• Manage Categories –** Create or update product categories

**• View Orders –** See customer orders and statuses

**• Process Refunds/Returns –** Handle customer complaints

**• Report Generation –** Sales, revenue, and customer data

**3. System Functional Modules:**

**•** User Module

• Product Catalog Module

• Shopping Cart Module

• Order Management Module

• Payment Gateway Module

• Admin Dashboard Module

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

**1. Requirement Gathering**

**2. Planning**

**3. Design**

**4. Development**

**5. Testing**

**6. Deployment**

**7. Maintenance**

* **[LAB EXERCISE]: Write a requirement specification for a simple library management system.**

=> Library Management System – Requirement Specification

1. Purpose:

To automate book management, issue, return, and member handling in a library.

2. Users:

• Admin/Librarian: Manage books and members

• Students/Members: Search and view books

3. Main Features:

• Login system for admin

• Add/edit/delete books

• Register/edit/remove members

• Issue and return books

• Search books by title/author

• Generate reports

4. System Requirements:

• Simple user interface

• Fast access and search

• Role-based access for security

5. Assumptions:

• Unique ID for each book

• Max 3 books issued per user

• 15-day return period

* **LAB EXERCISE: Perform a functional analysis for an online shopping system.**

**• Functional Analysis – Online Shopping System**

**1. User Roles:**

**• Customer/User**

**• Admin**

**•** (Optional: Vendor/Seller if multi-vendor platform**)**

**2. Functional Requirements:**

**Customer Functions:**

**• User Registration/Login –** Create and access user account

**• Browse Products –** View items by category, brand, etc.

**• Search Products –** Search by name, price, rating, etc.

**• Add to Cart –** Add selected items to shopping cart

**• Place Order –** Checkout and complete purchase

**• Payment –** Online payment via card, UPI, or COD

**• Order History –** View past orders and status

**• Product Review/Rating –** Leave feedback

**Admin Functions:**

**• Manage Users –** Add/edit/remove user accounts

**• Manage Products –** Add/edit/delete product listings

**• Manage Categories –** Create or update product categories

**• View Orders –** See customer orders and statuses

**• Process Refunds/Returns –** Handle customer complaints

**• Report Generation –** Sales, revenue, and customer data

**3. System Functional Modules:**

**•** User Module

• Product Catalog Module

• Shopping Cart Module

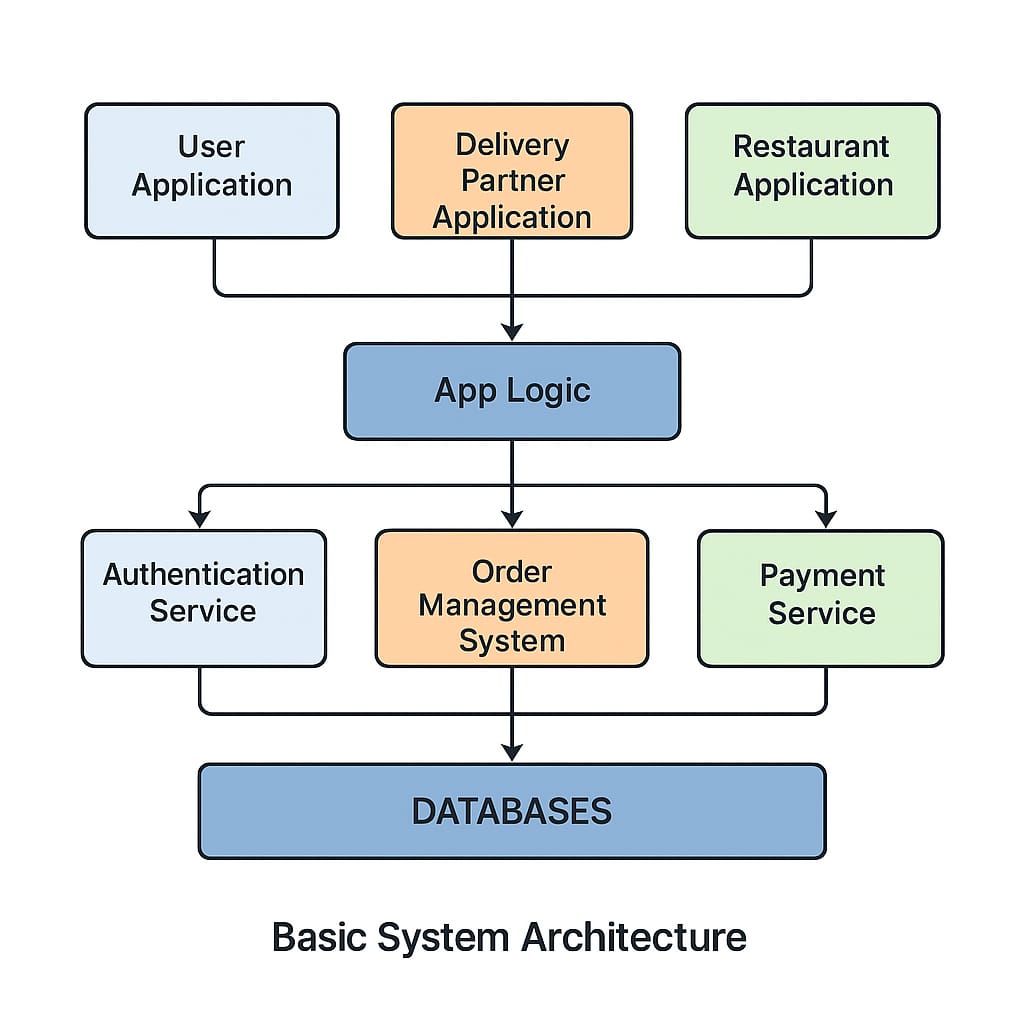
• Order Management Module

• Payment Gateway Module

• Admin Dashboard Module

# System Design

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



**1.** **Applications (Top Layer):**

• User Application – For customers to place orders.

• Delivery Partner Application – For delivery agents to view and deliver orders.

• Restaurant Application – For restaurants to receive and manage orders.

**2. App Logic (Middle Layer):**

• The main control unit of the system.

• Connects all the applications to the backend services.

**3. Services (Supporting Systems):**

• Authentication Service – Manages user login and security.

• Order Management System – Handles order processing, updates, and tracking.

• Payment Service – Processes online payments safely.

1. **Databases (Bottom Layer):**

• Stores all data like user profiles, menu items, orders, payments, and history.

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

**1. Functional Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output |
| TC001 | Addition of two positive numbers | 5 + 3 | 8 |
| TC002 | Addition of positive and negative number | 7 + (-2) | 5 |
| TC003 | Subtraction of two numbers | 10 - 4 | 6 |
| TC004 | Multiplication of two numbers | 6 \* 3 | 18 |
| TC005 | Division of two numbers | 12 / 4 | 3 |
| TC006 | Division resulting in a float | 7 / 2 | 3.5 |
| TC007 | Multiplying by zero | 15 \* 0 | 0 |
| TC08 | Division by zero | 5 / 0 | Error or "Cannot divide by zero" |

**2. Boundary Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| TestCase ID | Description | Input | Expected Output |
| TC011 | Maximum integer addition | 2147483647 + 1 | Overflow or Error |
| TC012 | Minimum integer subtraction | -2147483648 - 1 | Underflow or Error |
| TC013 | Very small float multiplication | 0.0000001 \* 0.0000001 | 1e-14 |
| TC014 | Large float division | 1e10 / 2 | 5e9 |

**3. Invalid Input Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output |
| TC015 | Non-numeric input | "a" + 5 | Error: Invalid input |
| TC016 | Empty input | "" | Error: Input required |
| TC017 | Symbol instead of number | $# + 10 | Error: Invalid input |
| TC018 | Division by string | 10 / "two" | Error: Invalid input |
| TC019 | Multiple operators | 2 ++ 3 | Error: Syntax error |

1. **Chained Operations (if supported):**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output |
| TC020 | Chain addition and multiplication | 2 + 3 \* 4 | 14 (if operator precedence) |
| TC021 | Chain with parentheses | (2 + 3) \* 4 | 20 |
| TC022 | Mix of operations | 10 - 2 + 5 | 13 |

* **[LAB EXERCISE]: Document a real-world case where a software application required.**
* **Critical maintenance.**

**Case Study: Facebook Outage – October 2021**

**Background:**  
On October 4, 2021, Facebook, WhatsApp, and Instagram experienced a global outage lasting over 6 hours. The disruption was caused by a router misconfiguration during routine maintenance.

**Issue:**  
The misconfiguration disrupted network traffic, disconnecting data centers . Even Facebook’s internal tools failed, significantly delaying recovery efforts.

**Impact:**

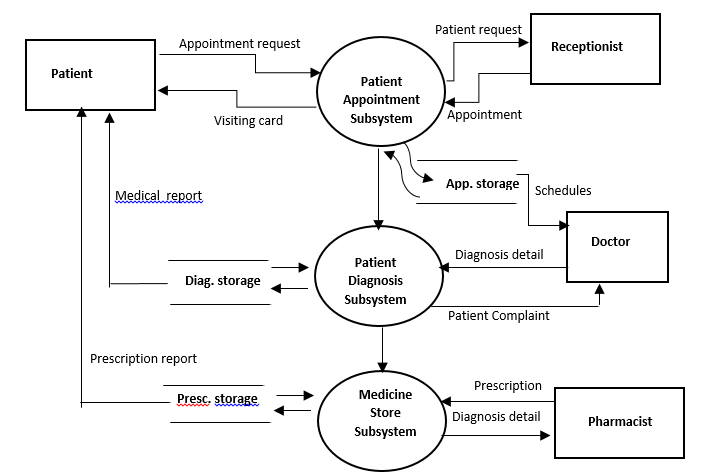
* Worldwide service outage
* Over $60 million in lost ad revenue
* Facebook’s stock dropped by 5%

**Action Taken:**

* Engineers manually accessed data centers to regain control
* Systems were gradually restarted
* Recovery tools and protocols were updated for future resilience

**Lessons Learned:**

* Backup access methods are critical
* Automated rollback mechanisms for configuration changes improve safety
* Regular disaster recovery drills are essential
* **[LAB EXERCISE]: Create a DFD for a hospital management system.**



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

#include <stdio.h>

int main() {

double num1, num2, result;

char op;

printf("Simple Calculator\n");

printf("Enter expression (e.g., 3 + 4): ");

scanf("%lf %c %lf", &num1, &op, &num2);

switch (op) {

case '+':

result = num1 + num2;

printf("Result: %.2lf\n", result);

break;

case '-':

result = num1 - num2;

printf("Result: %.2lf\n", result);

break;

case '\*':

result = num1 \* num2;

printf("Result: %.2lf\n", result);

break;

case '/':

if (num2 == 0) {

printf("Error: Division by zero\n");

} else {

result = num1 / num2;

printf("Result: %.2lf\n", result);

}

break;

default:

printf("Error: Invalid operator\n");

}

return 0;

}

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

