Assigment-1

**Module 1 – Overview of IT Industry**

**Q1. : Write a simple "Hello World" program in two different programming languages of your choice. Compare the structure and syntax.**

**Ans:-**

**C:- #include <stdio.h>**

**int main() {**

**printf("Hello World\n");**

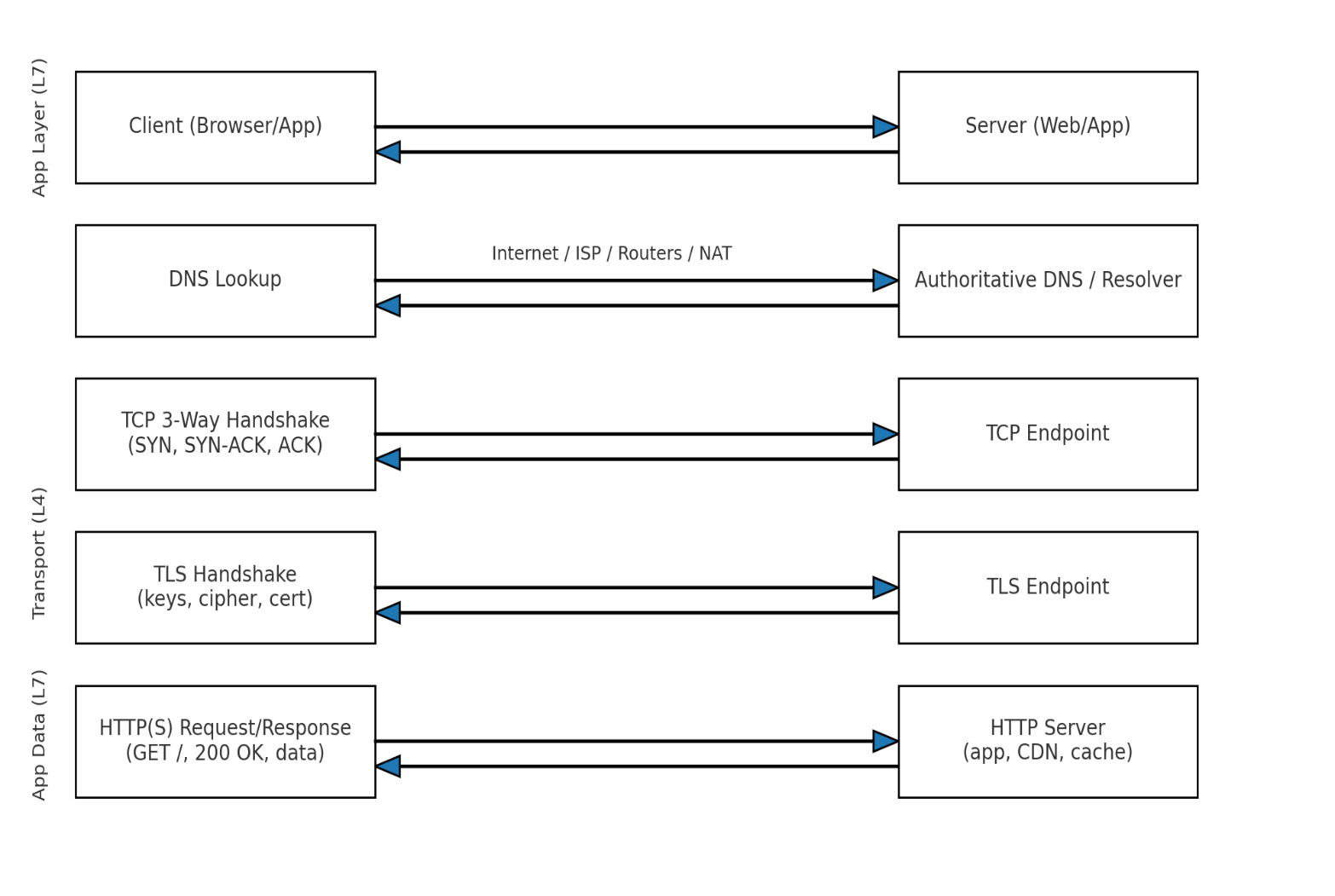
**return 0;**

**}**

**Python:- print("Hello World")**

**Q2.:Research and create a diagram of how data is transmitted from a client to a server over the internet**

**Ans:-**

****

**Q3.:Design a simple HTTP client-server communication in any language.**

**Ans:-**

A Server (using Python’s built-in http.server)

A Client (using the requests library)

**HTTP Server (server.py)**

from http.server import BaseHTTPRequestHandler, HTTPServer

class SimpleHandler(BaseHTTPRequestHandler):

def do\_GET(self):

# Send response status code

self.send\_response(200)

# Send headers

self.send\_header('Content-type', 'text/html')

self.end\_headers()

# Send the body of the response

self.wfile.write(b"Hello from the Server!")

# Server settings

host = "localhost"

port = 8080

server = HTTPServer((host, port), SimpleHandler)

print(f"Serving on http://{host}:{port}")

server.serve\_forever()

**HTTP Client (client.py)**

import requests

# Make a GET request to the server

response = requests.get("http://localhost:8080")

# Print the response text

print("Server says:", response.text)

**How to Run**

Save the server code as server.py and run:

python server.py

(It will start listening at <http://localhost:8080>.)

In a separate terminal, save the **client code** as client.py and run:

python client.py

(It will send a GET request to the server and print the response.)

**What Happens**

**Client** → Sends GET / request to localhost:8080.

**Server** → Receives request, sends back “Hello from the Server!”.

**Client** → Prints the server’s response

**Q4. Research different types of internet connections (e.g., broadband, fiber, satellite)and list their pros and cons**.

**Ans:-**

| Connection Type | Speed | Availability | Cost | Best For |
| --- | --- | --- | --- | --- |
| Broadband (DSL/Cable) | Medium | High | Medium | General home/office use |
| Fiber Optic | Very High | Low–Medium | Medium–High | Streaming, gaming, large downloads |
| Satellite | Medium | Very High | High | Remote/rural areas |
| Mobile Data (4G/5G) | Medium–High | High | Medium–High | On-the-go internet |
| Dial-Up | Very Low | Medium | Low | Very basic access |

* That’s all are pros and cons different types of internet connections.

**Q5.: Simulate HTTP and FTP requests using command line tools (e.g., curl)**

**Ans:-**

**HTTP Request (using curl)**

curl <http://example.com>

Sends an **HTTP GET** request to a website.

Shows the page’s HTML code in the terminal.

**FTP Download**

curl [ftp://speedtest.tele2.net/1MB.zip -o file.zip](ftp://speedtest.tele2.net/1MB.zip%20-o%20file.zip)

Downloads a file from an FTP server and saves it as file.zip.

**FTP with Login**

curl -u user:pass [ftp://ftp.example.com/myfile.txt -o local.txt](ftp://ftp.example.com/myfile.txt%20-o%20local.txt)

Connects to FTP with a username/password and downloads a file.

**In short:**

* **HTTP** → Get web pages or send data.
* **FTP** → Upload or download files from servers.

**Q6. Identify and explain three common application security vulnerabilities. Suggestpossible solutions.**

**Ans:-**

1. **SQL Injection (SQLi)**

**What it is:**  
When an attacker inserts malicious SQL code into a query, allowing them to view, modify, or delete database data.

**Example.**

SELECT \* FROM users WHERE username = 'admin' --' AND password = 'pass';

**Solution:**

* Use **prepared statements** or **parameterized queries**.
* Validate and sanitize all user inputs.

**2.Cross-Site Scripting (XSS)**

**What it is:**  
When attackers inject malicious JavaScript into web pages viewed by others, often to steal cookies or session data**.**

**Example:**

<script>alert('Hacked!');</script>

**Solution:**

* Escape special characters in user input.
* Use **Content Security Policy (CSP)**.
* Validate and sanitize all input/output.

1. **Cross-Site Request Forgery (CSRF)**

**What it is:**  
Tricks a logged-in user into performing an unwanted action on a website without their knowledge.

**Example:**

A hidden form on another site submits a request to change your account password.

**Solution:**

* Use **anti-CSRF tokens** in forms.
* Require re-authentication for sensitive actions.

| **Vulnerability** | **Risk** | **Solution** |
| --- | --- | --- |
| **SQL Injection** | Data theft or manipulation | Parameterized queries, input validation |
| **XSS** | Data theft, defacement | Input/output sanitization, CSP |
| **CSRF** | Unauthorized actions | Anti-CSRF tokens, re-authentication |

**Q7.:dentify and classify 5 applications you use daily as either system software orapplication software.**

**Ans:-**

Google Chrome (Web Browser)

* Type: Application Software
* Reason: Designed for users to browse the internet and interact with websites.

2. Microsoft Word (Word Processor)

* Type: Application Software
* Reason: Helps users create, edit, and format documents.

3. WhatsApp (Messaging App)

* Type: Application Software
* Reason: Enables users to send messages, make calls, and share media.

4. Windows Operating System

* Type: System Software
* Reason: Manages computer hardware, runs applications, and provides the user interface.

5. Antivirus Program (e.g., Windows Defender)

* Type: System Software
* Reason: Protects the computer from viruses and malware, running at a system level.

**Q8.:Design a basic three-tiersoftware architecture diagram for a web application**.

**Ans:-**

Three-Tier Structure

1. Presentation Layer (Client/User Interface)
2. Application Layer (Business Logic / Server)
3. Data Layer (Database)

+---------------------+

| Presentation | <-- Browser / Mobile App

| (HTML, CSS, JS) |

+---------------------+

|

v

+---------------------+

| Application | <-- Server-side Logic

| (Node.js, Java) |

+---------------------+

|

v

+---------------------+

| Data | <-- Database

| (MySQL, MongoDB) |

+---------------------+

**Q9.:Create a case study on the functionality of the presentation, business logic, and dataaccess layers of a given software system.**

**Ans:-**

Case Study: Three-Tier Architecture in an Online Food Ordering System

1. Presentation Layer (User Interface Layer)

Functionality:

* Displays menus, offers, and order status to customers.
* Allows users to browse restaurants, select food items, and place orders.
* Handles input from users (e.g., quantity, delivery address, payment details).
* Technologies: HTML, CSS, JavaScript, React, Bootstrap.

Example in Action:  
A user opens the website/app, sees a list of restaurants, clicks on one, selects items, and adds them to the cart. The interface updates in real time to show total price and offers.

2. Business Logic Layer (Application Layer)

Functionality:

* Validates user inputs (e.g., ensuring the delivery address is complete).
* Processes the order (calculates total cost, applies discounts, checks item availability).
* Coordinates communication between the presentation and data layers.
* Technologies: Node.js, Java, Python (Django/Flask), PHP.

Example in Action:  
When the user clicks "Place Order," this layer checks if the restaurant is open, verifies payment details, applies promotional discounts, and creates the order record in the database.

3. Data Access Layer (Database Layer)

Functionality:

* Stores and retrieves data related to users, menus, restaurants, and orders.
* Ensures data integrity and security.
* Handles queries, updates, and data relationships.
* Technologies: MySQL, PostgreSQL, MongoDB.

Example in Action:  
When the business layer requests the menu for a specific restaurant, this layer fetches the data from the database and sends it back. When the order is placed, it stores order details and updates stock availability.

**Q10.:Explore different types of software environments (development, testing, production).Set up a basic environment in a virtual machine.**

**Ans:-**  Types of Software Environments

| Environment | Purpose | Key Characteristics | Example Use Case |
| --- | --- | --- | --- |
| Development | Where developers write, debug, and build the software. | Flexible, frequent changes, includes IDEs, debuggers, local servers. | Writing and testing new features on a local machine. |
| Testing / Staging | Used to test software before releasing to users. | Controlled environment, mirrors production, used for QA and bug fixing. | QA team tests an e-commerce site before launch. |
| Production | The live environment where end-users interact. | Stable, secure, high-performance, minimal changes. | A live banking application accessed by customers. |

**Q11.: : Write and upload your firstsource code file to Github**.

**Ans:-**

Step 1 – Install Git

If Git isn’t already installed:

* Windows: [Download here](https://git-scm.com/download/win)
* Linux:
* sudo apt install git
* Mac:
* brew install git

Step 2 – Create a GitHub Account

1. Go to <https://github.com> and sign up (if you don’t have an account).
2. Once logged in, click New Repository.
3. Name it something like first-project.
4. Keep it Public for now and don’t add a README yet.
5. Click Create Repository.

Step 3 – Create Your First Source Code File

Example: a simple hello.py file.

print("Hello, GitHub!")

Save this file in a folder on your computer, e.g., first-project.

Step 4 – Push to GitHub

Open a terminal (or Git Bash) inside your project folder and run:

git init

git add hello.py

git commit -m "First commit: Hello GitHub"

git branch -M main

git remote add origin https://github.com/YOUR-USERNAME/first-project.git

git push -u origin main

*(Replace YOUR-USERNAME with your GitHub username.)*

Step 5 – Verify Upload

1. Go to your GitHub repository page.
2. You should see your hello.py file with the commit message.

**Q12.: Create a Github repository and document how to commit and push code changes.**

**Ans:-**

**GitHub Repository Creation & Code Commit Guide**

**1. Create a New GitHub Repository**

1. Log in to your GitHub account: <https://github.com>
2. Click the **+** icon (top right) → **New repository**.
3. Enter:
   * **Repository Name:** e.g., my-first-repo
   * **Description:** (optional)
   * **Visibility:** Public or Private
4. Leave "Initialize this repository with a README" **unchecked** (if you already have local code).
5. Click **Create repository**.

**2. Set Up Git Locally**

If you haven’t installed Git:

* **Windows:** [Download Git](https://git-scm.com/download/win)
* **Linux:**
* sudo apt install git
* **Mac:**
* brew install git

Configure your Git identity:

git config --global user.name "Your Name"

git config --global user.email "youremail@example.com"

**3. Initialize a Local Project**

Create a folder for your project and navigate into it:

mkdir my-first-repo

cd my-first-repo

Create a sample file:

echo "print('Hello, GitHub!')" > hello.py

Initialize Git:

git init

**4. Connect to GitHub Repository**

Copy the repository URL from GitHub (HTTPS recommended).

git remote add origin https://github.com/YOUR-USERNAME/my-first-repo.git

**5. Commit Your First Code**

Add the file(s) to staging:

git add hello.py

Commit with a message:

git commit -m "First commit: added hello.py"

**6. Push Code to GitHub**

Push to the main branch:

git branch -M main

git push -u origin main

**7. Verify on GitHub**

* Go to your repository’s GitHub page.
* You should see your committed file(s) and commit message.

**Workflow for Future Changes**

When you make edits to your files:

git add filename.py # Stage changes

git commit -m "Describe changes"

git push # Push changes to GitHub

**Q13.: Create a student account on Github and collaborate on a small project with a classmate.**

**Ans:-**

**Create a GitHub Student Account**

1. Go to **GitHub Education**.
2. Click **"Get Student Benefits"**.
3. Sign in with your existing GitHub account or create a new one.
4. Fill out the form:
   * **School-issued email address** (if you have one)  
     OR upload a **photo of your student ID**.
   * Proof of enrollment (student ID card, transcript, or acceptance letter).
5. Wait for approval (usually within a few days).
6. Once approved, you’ll get free access to:
   * Private repositories
   * GitHub Pro features
   * Developer tools (e.g., free hosting, CI/CD tools).

**2. Create a Project Repository**

1. Log in to GitHub → Click the **+** → **New Repository**.
2. Give it a name, e.g., class-project.
3. Choose **Public** (or Private if you want only teammates to see it).
4. Initialize with:
   * **README file** (project description)
   * **.gitignore** (optional, for ignoring certain files)
5. Click **Create Repository**.

**Q14.: Create a list of software you use regularly and classify them into the followingcategories: system, application, and utility software.**

**Ans:-**

**System Software**

*(Manages hardware and provides a platform for running applications)*

* **Windows 11** – Operating System
* **Linux Ubuntu** – Operating System
* **macOS** – Operating System
* **Android OS** – Mobile Operating System

**2. Application Software**

*(Performs specific user-oriented tasks)*

* **Google Chrome** – Web Browser
* **Microsoft Word** – Word Processing
* **WhatsApp** – Messaging
* **Zoom** – Video Conferencing
* **Spotify** – Music Streaming

**3. Utility Software**

*(Helps maintain, manage, and protect the computer system)*

* **Windows Defender** – Antivirus Protection
* **WinRAR** – File Compression/Extraction
* **CCleaner** – System Cleaning and Optimization
* **Disk Cleanup** – Storage Management Tool
* **Backup and Restore** – Data Backup Tool

**Q15.: Follow a GIT tutorial to practice cloning, branching, and merging repositories.**

**Ans:-**

**. Clone a Repository**

We’ll use a public GitHub repository as an example.

# Clone a repository

git clone https://github.com/octocat/Spoon-Knife.git

cd Spoon-Knife

✅ **You now have a local copy** of the project.

**2. Create and Switch to a New Branch**

# Create a new branch called 'feature-update'

git branch feature-update

# Switch to that branch

git checkout feature-update

(or do both in one step)

git checkout -b feature-update

✅ You’re now working on a separate branch without affecting main.

**3. Make Changes in the Branch**

Edit a file (e.g., index.html) and save it.

# Stage the changes

git add index.html

# Commit the changes

git commit -m "Updated index.html with new content"

**4. Merge Changes into Main Branch**

# Switch to the main branch

git checkout main

# Merge the feature branch

git merge feature-update

✅ Your changes are now part of main.

**5. Push Changes to GitHub**

git push origin main

**6. (Optional) Delete the Branch**

git branch -d feature-update

**Flow Summary**

1. **Clone** → Get a copy of a repository from GitHub.
2. **Branch** → Create a new branch for your changes.
3. **Commit** → Save your changes in that branch.
4. **Merge** → Bring the branch changes into the main code.
5. **Push** → Send changes back to GitHub.

**Q16.:Write a report on the various types of application software and how they improveproductivity.**

**Ans:-**

**Report on Types of Application Software and Their Role in Improving Productivity**

**▪ Graphics software**

**▪ Multimedia**

**▪ Education software**

**▪ Mange the data of a software.**

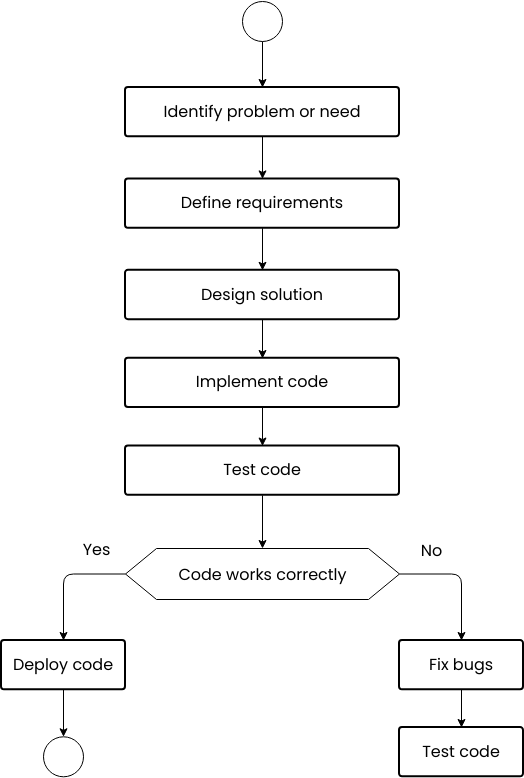
**▪ Make to easy to use.**

**▪ Make changes to work fast.**

**▪ Spreadsheet ▪ Web browser**

**Q17.: Create a flowchart representing the Software Development Life Cycle (SDLC).**

**Ans:-**

****

**Q18.: Write a requirementspecification for a simple library management system.**

**Ans :-**

**Requirement Specification for Library Management System**

**• User registration system , Simple search option for book by it’s title or author , Issue & return function , Book details management system**

**Q19.: Perform a functional analysis for an online shopping system.**

**Ans:-**

**Functional Analysis of an Online Shopping System**

**1. User Functions Register/Login View Product Details Add to Cart / Remove from Cart Place Orders Make Payments Track Orders Leave Reviews / Ratings**

**2. System Functions Shopping Cart Management Order Management Payment Gateway Integration Email Notifications Search and Filter Products**

**3. Functional Flow Example User logs in Searches for a productAdds product to cart Proceeds to checkout.**

**Q20.:** **Design a basic system architecture for a food delivery app.**

**Ans:-**

[ Customer App ] [ Restaurant App ] [ Delivery Partner App ]

│ │ │

└──────► [ Backend (APIs & Services) ] ◄────────────┘

│

┌─────────────────────────────────────────────────────┐

│ User Service | Order Service | Payment Service │

│ Delivery Service | Notification Service | Review │

└─────────────────────────────────────────────────────┘

│

┌─────────────── Databases & Storage ────────────────┐

│ SQL DB | NoSQL DB | Cloud Storage (Images, Logs) │

└────────────────────────────────────────────────────┘

│

┌──────────────── External Integrations ──────────────┐

│ Payment Gateways | Maps API | SMS/Email Services │

└─────────────────────────────────────────────────────┘

**Q21.: Develop test cases for a simple calculator program.**

**Ans:- # Simple Calculator**

def add(a, b):

    return a + b

def subtract(a, b):

    return a - b

def multiply(a, b):

    return a \* b

def divide(a, b):

    if b == 0:

        return "Error: Division by zero"

    return a / b

print("Addition Tests")

print(add(5, 3))

print(add(-5, -3))

print("\nSubtraction Tests")

print(subtract(10, 5))

print(subtract(5, 10))

print("\nMultiplication Tests")

print(multiply(5, 3))

print(multiply(-5, -3))

print("\nDivision Tests")

print(divide(10, 2))

print(divide(7, 2))

print(divide(5, 0))

**Bottom of FormQ22.; Document a real-world case where a software application required criticalmaintenance.**

**Ans:-**

-----------------+ | Receptionist | +---------+----------+ | v +---------------+ | Hospital Mgmt | | System | +---------------+ ^ ^ ^ | | | +--------+ +---+---+ +----------+ | Patient | | Doctor | | Pharmacy | +--------+ +-------+ +----------+

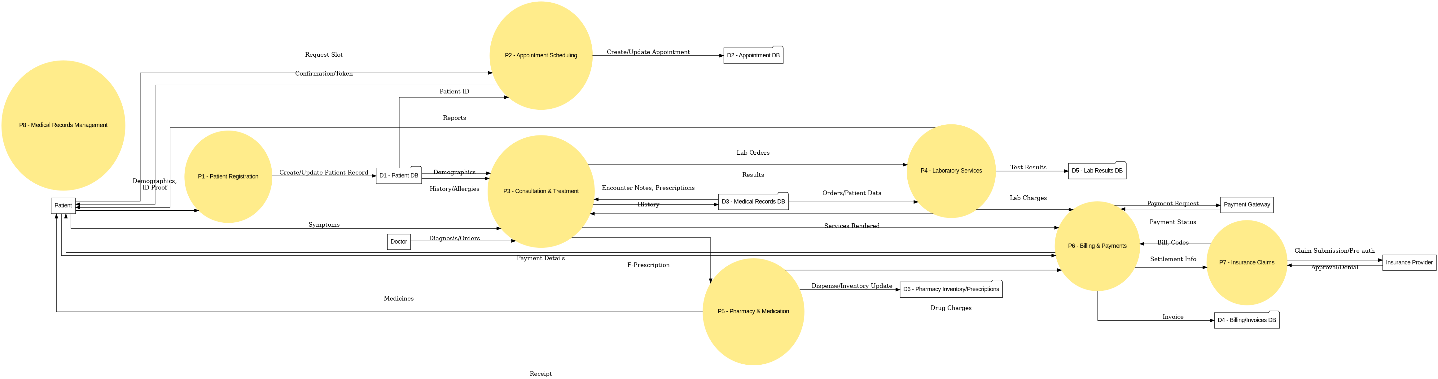
**Q23.: : Create a DFD for a hospital managementsystem.**

**Ans:-**

Context-Level DFD (Level 0)

* Shows the HMS as a single process interacting with external entities (Patient, Doctor, Insurance Provider, Payment Gateway, Admin).

**Level-1 DFD**

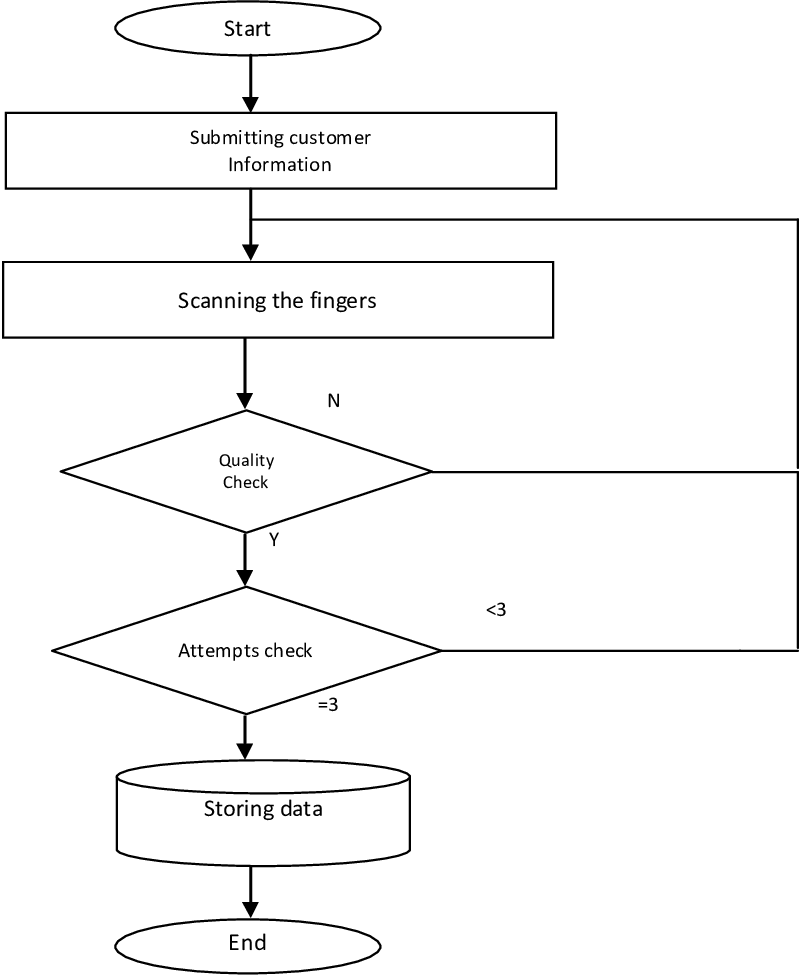
* Decomposes the system into main processes (Registration, Appointments, Consultation, Lab, Pharmacy, Billing, Insurance Claims, Medical Records) with data stores (Patient DB, Appointment DB, Medical Records DB, Billing DB, Lab Results DB, Pharmacy Inventory).

**Q24.: Build a simple desktop calculator application using a GUI library.**

**Ans:-** ---------------------------don’t know---------------------------

**Q25.: Draw a flowchart representing the logic of a basic online registration system.**

**Ans :-**

****