**Q 1 : Write a simple "Hello World" program in two different programming languages of your choice. Compare the structure and syntax.**

A : **1. Python Program**

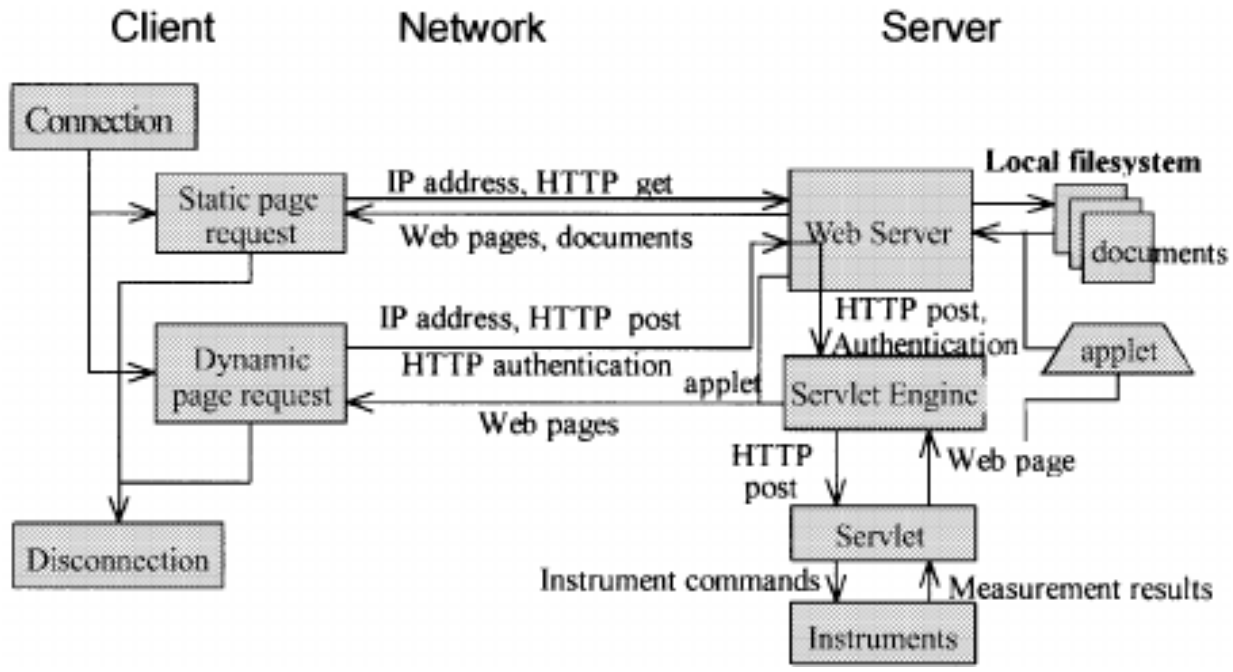
```
# This is a simple Python program
print("Hello, World!")
```

## 2. Java Program

```
// This is a simple Java program
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

**Q 2 : Research and create a diagram of how data is transmitted from a client to a server over the internet.**

**A :**



**Q 3 : Design a simple HTTP client-server communication in any language.**

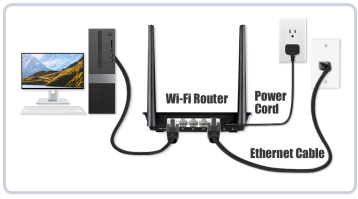
**A : Server Code (Python)**

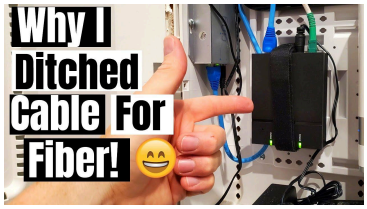
```
# simple_http_server.py

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 HTML message
        self.wfile.write(b"<html><body><h1>Hello,
Client!</h1></body></html>")

# Define server address and port
server_address = ('localhost', 8080)
httpd = HTTPServer(server_address, SimpleHandler)

print("✅ Server running on http://localhost:8080")
httpd.serve_forever()
```

**Q 4 : Research different types of internet connections (e.g., broadband, fiber, satellite)and list their pros and cons.**

**A :**

## 1. Fiber-Optic Internet





**Pros:**

- **Extremely high speeds for both downloads *and* uploads — up to gigabits per second. Mercury Fiber High Speed Internet+1**
- **Very low latency (delay) and high reliability — ideal for online gaming, video conferencing, heavy uploads. Excitel+1**
- **Less susceptible to electromagnetic interference and signal degradation over distance. VSOL+1**
- **Future-ready infrastructure (scalable for increasing demand). VSOL**

**Cons:**

- **Limited availability in many areas (especially rural or less-dense regions) due to infrastructure cost. weunion.com.cn+1**
- **Higher installation and monthly cost compared to some older technologies. Mercury Fiber High Speed Internet**
- **Installation may be more complex (drilling, wiring) in buildings not pre-wired. VSOL**

## 2. Cable Internet



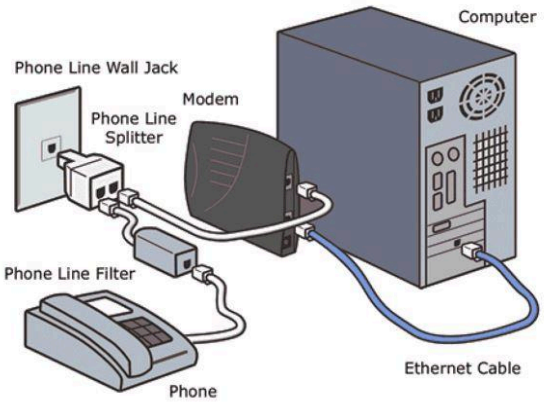
**Pros:**

- **Widely available in many urban and suburban areas. [Internet Speed Test | TestMySpeed.com+1]**
- **Good download speeds — sufficient for many households (streaming, multiple users). [fatbeamfiber.com]**
- **Usually less expensive than the highest-end fiber plans.**

**Cons:**

- **Bandwidth is often shared among users in a neighbourhood; during peak hours speeds may drop. [weunion.com.cn]**
- **Latency and upload speeds are often worse than fiber. [Excitel]**
- **Performance may degrade if network infrastructure is older or overloaded.**

## 3. DSL (Digital Subscriber Line)



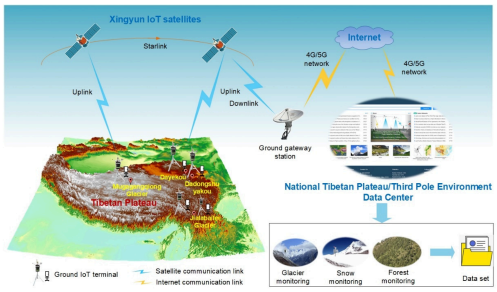

**Pros:**

- **Uses existing telephone lines — good for areas where newer infrastructure isn't available. [fatbeamfiber.com+1]**
- **Can be relatively affordable.**

**Cons:**

- **Speeds are significantly lower than fiber or cable in many cases. [swifttalk.net+1]**
- **Speed and quality degrade with distance from the provider's central office. [save on your broadband]**
- **Upload speeds may be very limited; not ideal for heavy usage (large uploads, streaming).**

## 4. Satellite Internet




**Pros:**

- **Coverage almost everywhere (especially in remote, rural or off-grid locations) as long as you have a clear view of the sky. [telecomssupermarket.in]**
- **Does not require extensive ground-cable infrastructure in the immediate area.**

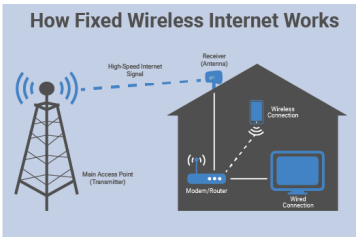
**Cons:**

- **High latency (signal has to travel up to satellites and back) — not optimal for real-time activities like gaming or live video calls. [telecomssupermarket.in+1]**

- **Often more expensive per unit of speed, may have data caps or throttling. [swifttalk.net]**

- **Performance can be affected by bad weather, dish alignment or obstructions. [telecomssupermarket.in]**

## 5. Fixed Wireless Internet




**Pros:**

- **Can serve areas where cables/fiber are not practical — uses radio or microwave signals. [Internet Speed Test | TestMySpeed.com]**
- **Quick to deploy compared to laying new cables.**

**Cons:**

- **Requires a clear line-of-sight to the tower; obstacles reduce reliability. [swifttalk.net]**
- **Performance may be variable; weather or physical obstructions can affect signal.**
- **Possibly more costly or less robust than wired alternatives for heavy use.**

**Q 5. Simulate HTTP and FTP requests using command line tools**

## 1. Simulating an HTTP Request

🧩 **Tool:** `curl`

`curl` **(Client URL) is the most common command-line tool for making HTTP requests.**

✅ **Example 1: Simple HTTP GET request**

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

**What it does:**

- **Sends a GET request to the server.**
- **Retrieves the HTML page from** `example.com`**.**
- **Displays the response directly in the terminal.**

✅ **Example 2: View headers only**

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

**Explanation:**

- **The** `-I` **flag sends a** `HEAD` **request.**
  **C. Shows HTTP response headers such as:**
  `HTTP/1.1 200 OK`
  **D.** `Content-Type: text/html; charset=UTF-8`
  **E.** `Content-Length: 1256`

✅ **Example 3: Sending POST data**

**F.**
```
curl -X POST -d "username=sonali&password=1234"
http://example.com/login
```

**Explanation:**

- **Sends an HTTP POST request to `/login`.**
- **Includes form data in the body of the request.**
- **You can use this to test APIs or login forms.**

✅ **Example 4: Downloading a file**

G. `curl -O https://example.com/file.zip`

**Explanation:**

- **Downloads `file.zip` from the server and saves it locally.**

## 💻 2. Simulating an FTP Request

🧩 **Tool: `ftp` or `curl` (both can be used)**

✅ **Example 1: Connect to an FTP server (interactive mode)**

H. `ftp ftp.example.com`

**Then enter:**

I. `Name: username`
J. `Password: ********`
K. `ftp> ls`
L. `ftp> get myfile.txt`
M. `ftp> bye`

**Explanation:**

- **Connects to the FTP server.**
- **Lists available files (`ls`).**
- **Downloads a file (`get`).**

- **`bye` disconnects from the server.**

✅ **Example 2: Downloading a file with `curl` (non-interactive)**

N. `curl -O ftp://ftp.example.com/pub/file.txt --user username:password`

**Explanation:**

- **Connects to the FTP server.**
- **Authenticates using the provided username and password.**
- **Downloads `file.txt` from the `/pub/` directory.**

✅ **Example 3: Uploading a file via FTP**

O. `curl -T upload.txt ftp://ftp.example.com/uploads/ --user username:password`

**Explanation:**

- **Uploads `upload.txt` to the server's `/uploads/` folder.**
- **Uses `-T` to specify the file to transfer.**

**Q 6. Identify and explain three common application security vulnerabilities. Suggestpossible solutions**

## 🧨 1. SQL Injection (SQLi)

🔍 **What it is:**

**An attacker injects malicious SQL commands into an input field (like a login form or search box) to access, modify, or delete database information.**

**Example:**

A. `SELECT * FROM users WHERE username = 'admin' AND password = '';`

**If the attacker enters `' OR '1'='1`, the query becomes:**

B. `SELECT * FROM users WHERE username = 'admin' AND password = '' OR '1'='1';`

**This always returns true, granting unauthorized access**

## 🐞 2. Cross-Site Scripting (XSS)

**🔍 What it is:**

**An attacker injects malicious scripts (usually JavaScript) into web pages viewed by other users.**
**For example, in a comment box:**

C. `<script>alert('Hacked!');</script>`

## 🔒 3. Cross-Site Request Forgery (CSRF)

**🔍 What it is:**

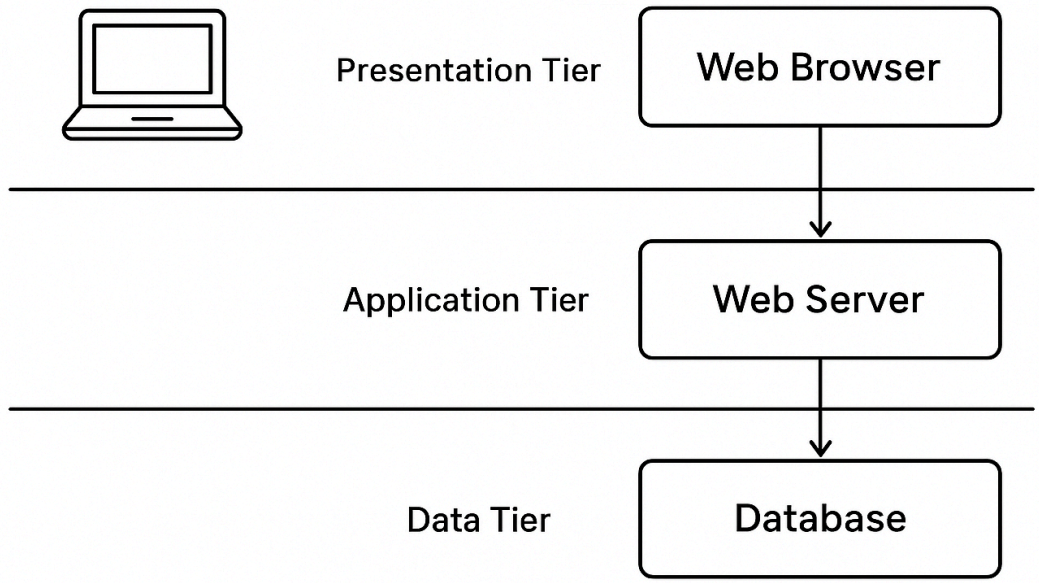
**An attacker tricks a logged-in user into performing unwanted actions on a web application — such as transferring money or changing a password — without their knowledge.**

**Example:**
**If a banking site allows this URL:**

D. https://bank.com/transfer?amount=1000&to=attacker

**Q 7. Design a basic three-tiersoftware architecture diagram for a web application.**

**A.**




**Q 8. Create a case study on the functionality of the presentation, business logic, and dataaccess layers of a given software system.**

**A.**

## 1. Presentation Layer (User Interface Layer)

🎯 **Purpose:**

This layer interacts directly with users. It displays information, accepts input, and sends user requests to the business logic layer.

🖥️ **Components:**

- Web pages / Mobile app screens
- HTML, CSS, JavaScript, React (frontend technologies)
- Forms for login, book search, shopping cart, and payment.

🧠 **Functionality:**

- Displays book catalog (title, author, price, image).
- Validates input (e.g., ensuring the email and password fields aren't empty).
- Collects data from users (e.g., selected book IDs) and sends it to the business logic layer using HTTP requests or APIs.
- Shows responses from the business logic layer (like "Order Confirmed" or "Payment Successful").

🔐 **Example:**

User fills out a form to search for a book → The request (/search?title=Python) is sent to the business logic layer.

## 🧮 2. Business Logic Layer (Application Layer)

🎯 **Purpose:**

This is the "brain" of the application – it processes data, applies business rules, and determines how data is created, displayed, and stored.

⚙️ **Components:**

- Web server (e.g., Node.js, Java Spring Boot, or Python Flask)
- Application logic classes and controllers
- Business rules and validation logic

🧠 **Functionality:**

- **Processes search requests: Filters books based on user queries.**
- **Handles orders: Calculates total price, applies discounts, checks stock availability.**
- **Manages authentication: Validates user credentials and sessions.**
- **Coordinates between Presentation and Data layers: Retrieves data from the database and returns it to the user interface in a readable format (like JSON).**

🧩 **Example:**

- **Request from Presentation Layer → /search?title=Python**
- **Logic Layer processes query → "Find all books with 'Python' in title"**
- **Calls Data Access Layer to fetch matching results.**

## 💾 3. Data Access Layer (Database Layer)

🎯 **Purpose:**

**This layer is responsible for storing, retrieving, and managing data in the database.**
**It hides database complexity from the upper layers.**

🗃️ **Components:**

- **Database server (e.g., MySQL, MongoDB, PostgreSQL)**
- **Data Access Objects (DAOs)**
- **SQL queries or ORM (Object Relational Mapping) tools like Hibernate or SQLAlchemy**

🧠 **Functionality:**

- **Executes CRUD operations (Create, Read, Update, Delete).**
- **Ensures data integrity and prevents SQL injection through prepared statements.**
- **Manages database connections and transactions.**
- **Returns data in structured form to the business logic layer.**

🧩 **Example:**

**When the business logic layer requests book data:**

```
SELECT * FROM books WHERE title LIKE '%Python%';
```

**The Data Layer executes this query and returns the results to the Business Logic Layer, which formats and sends it back to the user interface.**

## 🔁 Data Flow Example

**User → Presentation Layer → Business Logic Layer → Data Access Layer → Database**

**← Presentation Layer ← Business Logic Layer ← Data Access Layer ← Database**

## Q 8. Explore different types of software environments (development, testing, production).Set up a basic environment in a virtual machine.

**A.**

1. Development Environment (Dev):

- **This is the initial workspace for developers to write, test, and debug code.**
- **It often includes Integrated Development Environments (IDEs) and tools specific to the programming language or framework.**
- **The focus is on rapid iteration and individual component development.**

2. Testing Environment (QA/Test):

- **This environment is used for thorough testing of the application after development.**
- **It aims to replicate the production environment as closely as possible to identify bugs and performance issues.**
- **Various types of testing, such as unit, integration, system, and user acceptance testing (UAT), occur here.**

3. Production Environment (Prod):

- **This is the live environment where the software is deployed and accessed by end-users.**
- **It prioritizes stability, security, scalability, and high availability.**
- **Monitoring and disaster recovery mechanisms are crucial in this environment.**

**Q 9.Create a list of software you use regularly and classify them into the following categories: system, application, and utility software**

**A:**

- **Visual Studio Code**
- **Microsoft Excel,Word,Powerpoint**
- **Google**
- **Zoom**
- **Whatsapp**
- **Youtube**
- **Google Drive**
- **Google Docs**
- **Telegram**
- **Google Meet**
- **Gmail**
- **Notepad**

# Q 10. Write a report on the various types of application software and how they improveproductivity

## A.

## 1. Introduction

Application software refers to programs designed to help users perform specific tasks. Unlike system software, which manages hardware and system operations, application software focuses on enabling users to create documents, browse the internet, manage data, communicate, and solve daily problems. These applications play a crucial role in improving personal, academic, and business productivity by automating tasks, reducing manual effort, and enhancing efficiency.

## 2. Types of Application Software

### 2.1 Productivity Software

Productivity software includes programs that help users create documents, analyze data, and develop presentations.
Examples:

- Word processors (Microsoft Word, Google Docs)
- Spreadsheets (Excel, Google Sheets)
- Presentation tools (PowerPoint, Keynote)

How it improves productivity:

- Automates complex calculations
- Speeds up document creation
- Helps present ideas clearly
- Enhances collaboration through cloud sharing

### 2.2 Communication Software

**These applications enable individuals and teams to communicate instantly.**

**Examples:**

- **Email clients (Gmail, Outlook)**
- **Messaging apps (WhatsApp, Slack)**
- **Video conferencing tools (Zoom, Google Meet)**

**Productivity Benefits:**

- **Enables fast communication**
- **Supports remote work**
- **Helps teams collaborate in real time**
- **Reduces travel costs and delays**

### 2.3 Database Management Software

**Used to store, organize, and manage large amounts of data efficiently.**

**Examples:**

- **MySQL**
- **Oracle Database**
- **Microsoft Access**

**Productivity Benefits:**

- **Makes data retrieval fast and accurate**
- **Reduces errors in data handling**
- **Improves decision-making through structured storage**
- **Helps businesses keep track of inventory, customers, and transactions**

### 2.4 Multimedia Software

**Software used for creating, editing, and viewing images, videos, and audio.**

**Examples:**

- **Photoshop**
- **VLC Media Player**

- Canva
- Audacity

**Productivity Benefits:**

- Enhances creativity and marketing efforts
- Simplifies content creation
- Supports online education through video production
- Helps businesses design visual materials quickly

### 2.5 Web Browsers

**Browsers allow users to access and interact with websites and online resources.**

**Examples:**

- Google Chrome
- Mozilla Firefox
- Safari

**Productivity Benefits:**

- Provides quick access to information
- Supports cloud-based applications
- Enables online learning and research
- Integrates with extensions for improved workflow

### 2.6 Educational Software

**Designed to support learning and training.**

**Examples:**

- Duolingo
- Moodle
- Khan Academy

**Productivity Benefits:**

- Helps students learn at their own pace
- Provides interactive lessons

- Enhances knowledge retention
- Assists teachers with digital assessments

### 2.7 Business Software

These applications automate business operations such as accounting, project management, and customer service.
Examples:

- Tally
- SAP
- QuickBooks
- Trello / Asana

Productivity Benefits:

- Reduces manual paperwork
- Improves accuracy in financial calculations
- Enhances project tracking
- Speeds up decision-making

### 2.8 Graphics and Design Software

Used for drawing, modeling, and designing visuals.
Examples:

- AutoCAD
- Adobe Illustrator
- Blender

Productivity Benefits:

- Speeds up design processes
- Allows accurate visualization
- Helps create high-quality digital content

### 2.9 Entertainment Software

Applications used for games, streaming, and media playback.
Examples:

- Netflix

- Spotify
- Games and media apps

Productivity Benefits:

- Supports relaxation and stress relief
- Enhances creativity
- Keeps users entertained during breaks

## 3. How Application Software Improves Productivity

Application software improves productivity by:

### 3.1 Automating Tasks

Software reduces the need for manual work.
Example: Excel formulas can calculate thousands of rows instantly.

### 3.2 Enhancing Speed and Efficiency

Tasks that once took hours—like photo editing or data analysis—can now be done in minutes.

### 3.3 Supporting Collaboration

Cloud tools like Google Workspace allow teams to work on the same file at the same time.

### 3.4 Improving Accuracy

Software minimizes human errors, especially in accounting, data entry, and calculations.

### 3.5 Increasing Accessibility

Mobile and web apps enable users to work from anywhere.

### 3.6 Enabling Better Decision-Making

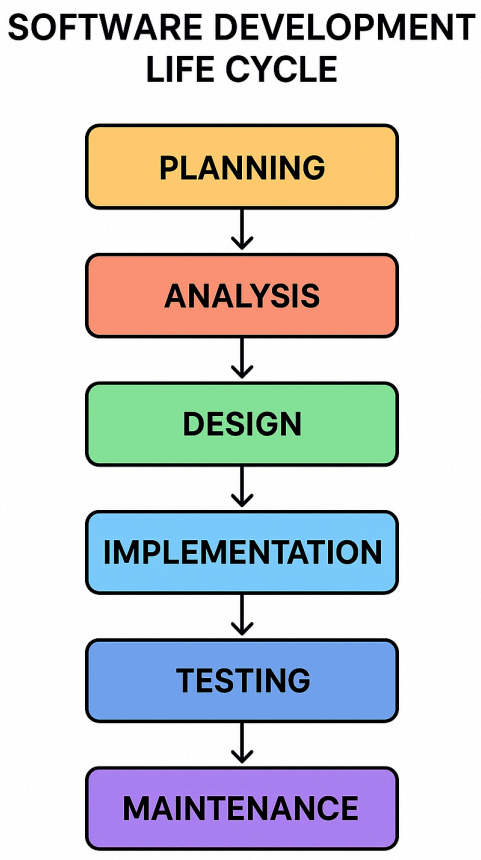
Data visualization and reporting tools help businesses make faster, informed decisions.

## 4. Conclusion

Application software plays a major role in improving productivity in all areas—education, business, communication, management, and creative work. By simplifying complex tasks, enhancing collaboration, and automating processes, application software enables individuals and organizations to work smarter and achieve more in less time. As technology continues to grow, the importance of application software in daily life and professional environments will continue to increase.

## Q 11. Create a flowchart representing the Software Development Life Cycle (SDLC).

## A.



## Q 12. Write a requirementspecification for a simple library management system

## A.

# 1. Introduction

### 1.1 Purpose

The purpose of this document is to outline the functional and non-functional requirements of a Simple Library Management System. The system is intended to help libraries manage books, users, borrowing, and returning operations efficiently.

### 1.2 Scope

The system will:

- Maintain a record of books
- Track library members
- Record borrowing and returning of books
- Provide search and reporting capabilities
- Support basic user authentication

The system will be used by librarians, library staff, and members.

### 1.3 Definitions

- Librarian: User with full administrative access.
- Member: User allowed to borrow/return books.
- Book: A physical or digital item available in the library.

# 2. Overall Description

### 2.1 System Perspective

The Library Management System is a standalone application that stores data in a central database. It interacts with users through a graphical or web-based interface.

### 2.2 Users of the System

- Librarian / Admin
- Library Members

### 2.3 Assumptions and Dependencies

- Users have basic computer knowledge.
- The system requires a stable database connection.
- All data stored is accurate and maintained by authorized users.

# 3. Functional Requirements

## 3.1 User Authentication

FR1: The system shall allow librarians and members to log in using username and password.
FR2: The system shall restrict access based on user role (admin/member).
FR3: The system shall allow librarians to reset member passwords.

## 3.2 Book Management

FR4: The system shall allow librarians to add new books with details (title, author, ISBN, category, quantity).
FR5: The system shall allow librarians to edit book information.
FR6: The system shall allow librarians to delete a book record.
FR7: The system shall allow the system to update the number of available copies automatically after a borrow/return action.

## 3.3 Member Management

FR8: The system shall allow librarians to add new members with personal details (name, ID, contact).
FR9: The system shall allow librarians to update member information.
FR10: The system shall allow librarians to deactivate or delete member accounts.

### 3.4 Borrowing and Returning

FR11: The system shall allow members to request book borrowing.
FR12: The system shall allow librarians to issue books to members.
FR13: The system shall record the borrow date and due date.
FR14: The system shall allow members or librarians to return books.
FR15: The system shall calculate overdue fines (if applicable).
FR16: The system shall prevent borrowing if no copies are available.

### 3.5 Search and Catalog

FR17: The system shall allow users to search books by title, author, category, or ISBN.
FR18: The system shall display book availability status.
FR19: The system shall allow filtering by category or author.

### 3.6 Reports

FR20: The system shall generate reports for:

- Books borrowed
- Books overdue
- Active members
- Available/low stock books

FR21: The system shall allow reports to be exported (PDF or CSV).

## 4. Non-Functional Requirements (NFR)

### 4.1 Performance Requirements

NFR1: The system shall handle at least 100 simultaneous users.
NFR2: Search results shall be displayed within 3 seconds.

### 4.2 Security Requirements

NFR3: Passwords shall be encrypted.
NFR4: Only librarians shall perform administrative functions.
NFR5: User sessions shall automatically log out after inactivity.

### 4.3 Usability Requirements

NFR6: The system shall have a user-friendly interface.
NFR7: Users shall be able to perform common tasks within three clicks.

### 4.4 Reliability Requirements

NFR8: The system shall provide 99% uptime.
NFR9: The system shall create automatic backups daily.

### 4.5 Compatibility Requirements

NFR10: The system shall run on Windows, Linux, or a web browser.
NFR11: The system shall support common screen resolutions.

## 5. System Constraints

- Must use a relational database (MySQL, PostgreSQL, or similar).
- Must comply with standard data protection rules.
- Works in offline or online mode depending on the version.

## 6. Conclusion

The Library Management System will streamline library operations by simplifying book tracking, user management, borrowing/returning processes, and reporting. It ensures better control, reduces manual paperwork, and improves overall library productivity and efficiency.

# Q 13. Perform a functional analysis for an online shopping system

## A.

## 1. Introduction

A functional analysis explains how an online shopping system works by identifying the main functions, user interactions, inputs, outputs, and processes involved. The system allows customers to browse items, manage a shopping cart, make payments, and track orders, while administrators manage products, inventory, and users.

# 2. System Actors (Users)

### 2.1 Customer

- Browses products
- Adds items to cart
- Places orders and makes payments
- Tracks delivery status

### 2.2 Administrator

- Manages product catalog
- Manages orders and customer accounts
- Views reports

### 2.3 Delivery Personnel

- Updates order delivery statuses

### 2.4 Payment Gateway

- Processes online payments
- Validates transactions

# 3. Major Functional Areas

**The functions of the system can be grouped into the following categories:**

## 3.1 User Management

**Functions:**

- **User registration**
- **User login/authentication**
- **Profile management**
- **Password reset**

**Inputs:**

- **User details (name, email, password, address)**

**Outputs:**

- **User account created**
- **Login confirmation**
- **Profile update confirmation**

## 3.2 Product Browsing & Catalog Management

**Customer Functions:**

- **View product categories**
- **Search products**
- **Filter products by price, ratings, brand**
- **View product details (price, description, reviews)**

### Admin Functions:

- Add new products
- Edit product information
- Delete/disable products
- Manage product categories

### Inputs:

- Search keywords, filters, new product details

### Outputs:

- Product listings
- Product details page
- Updated catalog

## 3.3 Shopping Cart Management

### Functions:

- Add items to cart
- Remove items from cart
- Update quantity
- Display total price
- Save cart for logged-in users

### Inputs:

- Product ID, quantity

### Outputs:

- Updated cart view
- Cart value calculation

## 3.4 Order Processing

**Functions:**

- Checkout
- Select delivery address
- Apply discounts/coupons
- Choose payment method
- Order confirmation

**Inputs:**

- User address, payment details, selected items

**Outputs:**

- Order summary
- Order confirmation message
- Order ID

## 3.5 Payment Processing

**Functions:**

- Secure payment submission
- Payment validation
- Transaction success/failure notification

**Inputs:**

- Payment details (UPI, credit card, COD selection)

**Outputs:**

- Payment success
- Payment failure

- Invoice generation

## 3.6 Order Tracking & Management

### Customer Functions:

- View order history
- Track delivery status
- Request order cancellation or return

### Admin/Delivery Functions:

- Update delivery status
- Approve cancellation or returns

### Inputs:

- Order ID, status updates

### Outputs:

- Tracking updates (Processing → Packed → Shipped → Delivered)

## 3.7 Review and Rating System

### Customer Functions:

- Rate purchased items
- Submit product reviews

### Inputs:

- Rating stars, review text

### Outputs:

- Displayed reviews on product page

- Updated average rating

## 3.8 Notification/Communication System

### Functions:

- Send emails/SMS notifications
    - Order confirmation
    - Shipping updates
    - Delivery confirmation
    - Offers and promotions

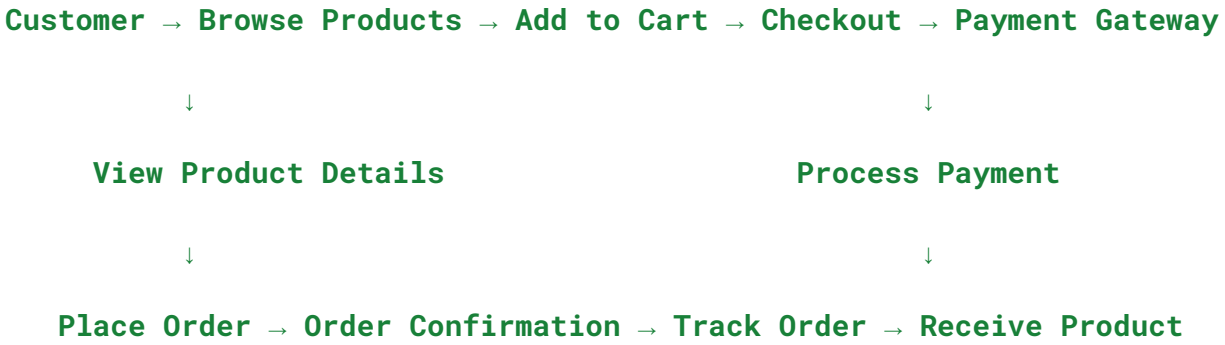
### Inputs:

- User contact information

### Outputs:

- Notifications delivered

# 4. Functional Diagram (Text Format)



# 5. Summary of Inputs and Outputs

| Function Area | Inputs | Outputs |
|---|---|---|
| User Management | User details, credentials | Account creation, login success |
| Product Browsing | Search keywords, filters | Product list, product details |
| Cart Management | Product ID, quantity | Updated cart, total price |
| Order Processing | Address, items, payment type | Order confirmation |
| Payment Handling | Payment details | Success/failure message |
| Order Tracking | Order ID | Order status |
| Review System | Rating, review text | Displayed reviews |
| Notifications | Email/phone | Alerts & confirmations |

# 6. Conclusion

The functional analysis of the Online Shopping System shows how the system interacts with customers, administrators, delivery personnel, and the payment gateway to deliver a complete e-commerce experience. By breaking the system into functional areas, it becomes easier to design, implement, and test each part effectively, ensuring a smooth and efficient shopping process.

## Q 14. Design a basic system architecture for a food delivery app

### A.

#### 1. Presentation Layer (Front-End)

Interfaces used by different users.

a. Customer App (Mobile/Web)

- Browse restaurants and menus
- Place orders
- Track delivery
- Make payments
- Manage profile & address

b. Restaurant App / Dashboard

- Receive and manage customer orders
- Update menu items
- Set availability status
- Track earnings

c. Delivery Agent App

- Accept delivery requests
- Navigate to restaurant and customer
- Update order status

## 2. Application Layer (Back-End Services)

Core logic and communication between all components.

### a. API Gateway

- Handles all client requests
- Routes to appropriate services

### b. Authentication Service

- Login/signup
- User identity management
- Token generation

### c. Restaurant Management Service

- Restaurant details
- Menus
- Prices, availability

### d. Order Management Service

- Create/process/update orders
- Order history
- Tracks order state (placed → accepted → prepared → picked → delivered)

### e. Delivery Management Service

- Assign delivery agents
- Track agent location
- Optimize delivery routes

### f. Payment Servics

- Manage online payments
- Refunds
- Transaction confirmation

### g. Notification Service

- Push notifications
- SMS/email alerts
- Order updates

## 3. Data Layer (Database & Storage)

### a. Databases

- User DB: customers, restaurant owners, delivery agents
- Menu DB: items, categories, prices
- Order DB: order details, status, timestamps
- Delivery DB: delivery status, routes
- Payment DB: transactions, invoices

### b. Storage

- Images (food, restaurant logos)
- Digital receipts

## 4. External Integrations

- Maps API → navigation, distance calculation
- Payment Gateways (Razorpay, Stripe, PayPal)
- SMS/Email Services (Twilio, AWS SES)

## ✔ Summary

A basic food delivery app architecture includes:

### Frontend

- Customer app
- Restaurant dashboard
- Delivery agent app

### Backend

- Authentication

- Order management
- Restaurant/menu service
- Delivery logistics
- Payment and notifications

Data Layer

- User, menu, order, delivery databases
- Cloud storage

This layered architecture ensures scalability, security, and efficient communication between all components.

## Q 15. Develop test cases for a simple calculator program.

A.

# 1. Addition Test Cases

| Test Case I | Input (A, B | Operatio | Expected Output | Type |
|---|---|---|---|---|
| ADD_01 | 5, 3 | + | 8 | Positive numbers |
| ADD_02 | -4, 6 | + | 2 | Negative + Positive |
| ADD_03 | -5, -5 | + | -10 | Two negatives |
| ADD_04 | 0, 10 | + | 10 | Zero addition |
| ADD_05 | 999999, 1 | + | 1000000 | Large number |

| | | | | |
|---|---|---|---|---|
| ADD_06 | "a", 3 | + | Error | Invalid input |

## 2. Subtraction Test Cases

| Test Case I | Input (A, B | Operatio | Expected Output | Type |
|---|---|---|---|---|
| SUB_01 | 10, 5 | - | 5 | Positive numbers |
| SUB_02 | 5, 10 | - | -5 | Negative result |
| SUB_03 | -5, -5 | - | 0 | Two negatives |
| SUB_04 | 0, 7 | - | -7 | Zero minus positive |
| SUB_05 | 1000000, 1 | - | 999999 | Large number |
| SUB_06 | "x", 2 | - | Error | Invalid input |

---

## 3. Multiplication Test Cases

| Test Case I | Input (A, B | Operatio | Expected Output | Type |
|---|---|---|---|---|
| MUL_01 | 4, 3 | * | 12 | Positive × Positive |

| | | | | |
|---|---|---|---|---|
| MUL_02 | -4, 3 | * | -12 | Negative × Positive |
| MUL_03 | -3, -3 | * | 9 | Negative × Negative |
| MUL_04 | 0, 8 | * | 0 | Zero multiplication |
| MUL_05 | 99999, 99999 | * | 9999800001 | Large numbers |
| MUL_06 | 2, "b" | * | Error | Invalid input |

---

# 4. Division Test Cases

| Test Case I | Input (A, B | Operatio | Expected Output | Type |
|---|---|---|---|---|
| DIV_01 | 10, 2 | / | 5 | Normal division |
| DIV_02 | -10, 2 | / | -5 | Negative ÷ Positive |
| DIV_03 | -10, -2 | / | 5 | Negative ÷ Negative |
| DIV_04 | 0, 5 | / | 0 | Zero divided |
| DIV_05 | 5, 0 | / | Error (Divide by Zero) | Division error |

| | | | | |
|---|---|---|---|---|
| DIV_06 | "x", 3 | / | Error | Invalid input |

---

## 5. General Functional Test Cases

| Test Case I | Input | Description | Expected Output |
|---|---|---|---|
| GEN_01 | 5 + 3 * 2 | Order of operations | Should follow BODMAS |
| GEN_02 | 2.5 + 4.1 | Decimal support | 6.6 |
| GEN_03 | -3.5 * 2 | Decimal + negative | -7 |
| GEN_04 | Blank input | No numbers provided | Error message |
| GEN_05 | Special characters (#, @ | Invalid | Error |

---

## 6. User Interface / Usability Test Cases (If GUI)

| Test Case I | Test | Expected Outcome |
|---|---|---|
| UI_01 | Buttons clickable | Each button performs correct function |

| | | |
|---|---|---|
| UI_02 | Enter long numbers | Display handles overflow or scrolls |
| UI_03 | Press "=" repeatedly | Should repeat last operation or do nothing |
| UI_04 | Clear button | Resets to 0 |
| UI_05 | Backspace button | Deletes last entered character |

---

# 7. Performance Test Cases

| Test Case I | Description | Expected Result |
|---|---|---|
| PERF_01 | Perform 1000 operations quickly | No crashes |
| PERF_02 | Large integer input (100 digits) | Should handle or show error graceful |

## Q 16. Document a real-world case where a software application required criticalmaintenance

## A.

## 1. Overview

In May 2017, the WannaCry ransomware attack affected over 230,000 computers across 150 countries, crippling operations in hospitals, telecom companies, transportation systems, and government agencies.
One of the most severely impacted organizations was the UK National Health

Service (NHS), where outdated and unpatched systems directly contributed to widespread disruptions.

This incident is a strong real-world example of how critical software maintenance—specifically patch management—can prevent catastrophic failures.

# 2. Background

### Software Involved

- Microsoft Windows operating systems, especially older versions like Windows XP and Windows 7.
- The vulnerability exploited was EternalBlue, a security flaw in the SMBv1 protocol.

### Patch Availability

- Microsoft had released a critical security patch (MS17-010) in March 2017, two months before the attack.
- Many affected organizations failed to apply the patch, leaving systems exposed.

# 3. What Went Wrong

### Key Maintenance Failures

1. Unpatched Operating Systems
   Large numbers of systems were running outdated and unsupported OS versions without security updates.
2. Lack of Centralized Patch Management
   Many organizations, including the NHS, did not have a reliable, centralized method to ensure all systems were updated.
3. Dependency on Legacy Applications
   Some critical hospital software was tied to older operating systems, preventing timely upgrades.
4. Insufficient Backup and Recovery Plans
   When ransomware encrypted system data, many organizations lacked clean, recent backups.

# 4. Impact of the Failure

### Organizational Impact

- NHS hospitals cancelled thousands of medical appointments and emergency patients were diverted.
- Staff lost access to:
    - Patient records
    - Diagnostic equipment
    - Internal communication systems

### Global Impact

- Transport systems (e.g., FedEx, Deutsche Bahn) halted services.
- Some countries declared national emergencies.

### Financial Impact

- Estimated global losses reached $4–8 billion.
- NHS alone incurred £92 million in damage and recovery cost.

# 5. Critical Maintenance Actions Taken

After the attack began, several urgent maintenance measures were implemented:

### 1. Emergency Patch Deployment

Microsoft released:

- Emergency patches even for unsupported systems like Windows XP, Windows Server 2003, etc.
- Organizations scrambled to deploy MS17-010 across all devices.

### 2. System Isolation

Infected machines were unplugged from networks to prevent further spread.

### 3. Data Restoration & Recovery

- Backup systems were used to restore services.
- Systems where backups were unavailable required full reinstallations.

### 4. Infrastructure Overhaul

Long-term maintenance upgrades included:

- Migrating away from outdated operating systems
- Strengthening cyber hygiene practices
- Implementing centralized patch management systems
- Mandatory software upgrade cycles

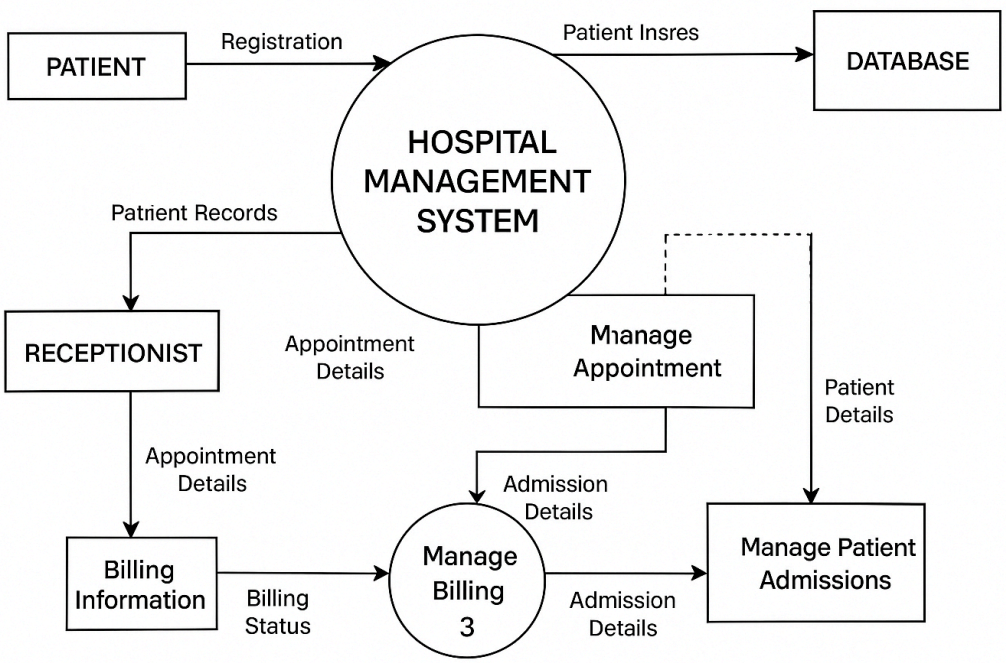
# 6. Lessons Learned

### Why Critical Maintenance Is Essential

1. Security patches must be applied promptly
   Delayed patching exposes systems to preventable attacks.
2. Legacy systems must be modernized
   Unsupported systems create vulnerabilities.
3. Centralized update and asset management is required
   Manual updates are unreliable in large organizations.
4. Routine backups protect against data loss
   Without backups, recovery becomes costly and time-consuming.
5. Proactive maintenance is cheaper than reactive recovery
   The cost of prevention is far lower than the cost of incident response.

## Q 17. Create a DFD for a hospital managementsystem

### A.



**Q 18. Draw a flowchart representing the logic of a basic online registration system.**

**A.**