# FULL STACK DEVELOPER ASSIGNMENT MODULE – 1 OVERVIEW OF IT INDUSTRY

## 1. What is Program?

- Program it is a set of Instruction

## 2. What is Programming?

- It is a set of words and symbol use to write a programs

## 3. What are key steps involved in programing process?

- ➢ Identifying a problem – Understand a problem and list possible solution
- ➢ Selecting a solution – Choose the best solution and design it
- ➢ Writing an algorithm – Prepare an algorithm and pseudo code to solve the problem
- ➢ Writing the code – Translate the design into lines of code that a computer can understand
- ➢ Debugging – Looks for and fix any errors in the code
- ➢ Testing – Test the program to ensure it functions properly
- ➢ Documenting – Write a clear Documentation to help other developers understand and maintain the software.

## 4. Types of Programing languages?

- ➢ Procedural Programing Languages (C Languages)
- ➢ Object Oriented Languages
- ➢ Logical Languages
- ➢ Functional Languages ( Python)

## 5. What are the main difference in high level and low level programing language?

- High Level programing language: are user friendly, portable and efficient for general application development.

- Low Level Languages: provide direct hardware control, making them ideal for system programing and performance critical application

## 6. World Wide Web & How Internet work?

- The World Wide Web is a system of interlinked hypertext document and other resources are accessed the Via Internet

- User Can accessed the content of these sites from any part of world over the internet using such as a computer laptop cell phones etc.

## 7. Describe the role of client and server in web communication?

- Client sometimes on

- Initiate a request to the server when interested

- Ex. Web browser on your laptop or phone

- Doesn’t Communication directly with other clients. Needs to know the server address.

## 8. Network layers on client and server

- In Client server communication, network layer refers to the layer in the network stack responsible for routing data packets between the client and server using logical address(like IP Address), essentially determining the best path to send data across the network, and is primarily associated with the internet Protocol within the TCP/IP model

## 9. Explain the function of the TCP/IP model and its layers.

- The **TCP/IP model** (Transmission Control Protocol/Internet Protocol) is a conceptual framework used for understanding and implementing communication protocols in network systems, particularly for Internet-based communications. It provides a set of standards and protocols for devices to communicate over the Internet and other networks.

- There are four layers of the TCP/IP model: network access, Internet, Transport and application the TCP/IP model passes data through these layer in particular order when user send a information and then reverse order again when the data received.

## 10. Explain the client server Communication?

- Client Sometimes on

- Initiate request the server when interested

- Ex web browser on your laptop or phone

- Doesn’t Communication directly with clients. Needs to know the server address

## 11. Types of Internet Connection

- Digital Subscribers Lines

- Cable Internet

- Fiber Optic

- Satellite Internet

- Wireless

- Board Band Over power line

## 12. How does Broadband differ from fiber optic Internet?

- Broadband is a general team covering different types of internet connection (DSL, cable, Satellite, wireless)

- fibber optic internet is the fastest and most reliable types of broadband, but it has limited availability And higher costs.

## 13. Protocols

- Set of rules which are used in digital communication to connect network devices and exchange information between them

## 14. What are the difference between HTTP and HTTPS protocols?

- HTTP ( Hypertext transfer protocol ) and HTTPS ( Hypertext transfer protocol secure) are protocols used for transmitting data over the web.

### HTTP (Hyper Text Transfer Protocol ) :

- ❖ Data transmitted in plan text
- ❖ Uses Port 80
- ❖ No encryption, less secure
- ❖ Does not require a security certificate
- ❖ Suitable for non-sensitive Information

### HTTPS (Hyper Text Transfer Protocol Secure) :

- ❖ Data is Encrypted
- ❖ Uses Port 443
- ❖ Provides Secure Communication through SSL/TLS encryption
- ❖ Requires an SSL/TLS certificate
- ❖ Suitable for sensitive information and secure transaction

## 15. Application Security

- Application Security involves implementing security measures at the application level to protect against threats and vulnerabilities during development , deployment and usages

## 16. What is role of encryption in securing application?

- Encryption plays a crucial role in securing application by protecting the confidentiality, integrity, and authenticity of data

- Encryption is fundamental security mechanism that ensure sensitive data remains private, intact, and accessible only to authorized parties. It is an essential tool for securing applications and protecting user data in an increasingly digital world

## 17. Software applications and Its Types

- Software applications, often referred as app are program or software designed to perform specific task for user. These task can range from productivity and communication to entertainment to beyond

- ❖ Application software
- ❖ System Software
- ❖ Driver Software
- ❖ Middleware Software
- ❖ Programing Software

## 18. What is difference between system software and applications software?

- System software responsible for managing variety of Independent for hardware Components, so they that work together. It Purpose is unburden the application software programmer from often complex details of the particular computer being used, including such accessories as communication device, printers, device readers, and Keyboard and also partition the computer resources such as memory and processor time in a safe stable manner

- Application software is a developed to aid in any task that benefit from computation. It is a broad category and encompasses software of many kinds, including the internet browser being used displayed the page. This category includes ex, Business software, Computer aided design, database, Decision making software, Educational software, Image editing

## 19. Software Architecture?

- Software Architecture refers to high level structure of system software, encompassing the organization of its components, their relationship, and the principle guiding its design and evolution.

- It Serves blueprint for both the system and the project developing it. Ensuring that the software meets both functional and non- functional requirements.

## 20. What is significance of modularity in software architecture?

- Modularity is fundamental design principle in software architecture that emphasizes dividing a system into smaller, self-contained units or modules.

Each module encapsulate a specific functionality, making the overall system easier to develop, understand and maintain

## 21. Layer in Software Architecture?

- ❖ Presentation Layer
- ❖ Application Layer
- ❖ Business Layer
- ❖ Persistence Layer
- ❖ Database Layer

## 22. Why are layer in important in software Architecture?

- Layering Is Fundamental Concept in Software Architecture, where the software is organized into a series of layers, each with specific responsibilities

## 23. Software Environments

- A Software environments refers to the collection of software tools, libraries, framework and configuration that support the development, deployment, testing and operations of the software application.

- It provides the necessary of infrastructure and setup for software project to run efficiently

## 24. Explain the importance of development environment in software production?

- A Development environment is crucial in software production because it provides a dedicated space for developers to write, test, and debug code without impacting live system, allowing for rapid experimentation early bug detection ultimately, the creation of higher quality software by minimizing error before deployment real user.

## 25. Source Code

- Source code It is Source of computer program

- Source Code is human-readable set of instruction and statement written by programmer in programming language to create software applications.

- It is fundamental component of any software and serves as the blueprint for the software functionality

## 26. What is the difference between source code and machine code

- **Source Code**: This is the high-level, human-readable code written by programmers using programming languages such as Python, Java, C++, JavaScript, etc. It is written in a syntax that is understandable to humans but not directly executable by computers.

- **Source Code**: Easily readable and understandable by humans. It uses keywords, variables, functions, and logical structures that are meaningful to developers.

- **Source Code**: Needs to be translated into machine code for the computer to understand and execute it. This translation is done through compilers or interpreters.

- **Source Code**: Created and modified by programmers. It is written in a high-level programming language that allows for easier understanding, problem-solving, and communication.

- **Source Code**: High-level or low-level (in the case of assembly code). High-level programming languages are abstracted from the hardware, making them easier to learn and use.

- **Source Code**: Portable across different systems (with some modifications depending on the language and platform). The same source code can generally run on different types of computers after being recompiled or reinterpreted for that platform.

- **Source Code**: It is written during the software development process by programmers. It is subject to changes and iterations throughout the development lifecycle.

- **Machine Code**: This is the low-level, binary code that is directly understood and executed by a computer's central processing unit (CPU). It consists of instructions in binary or hexadecimal form (combinations of 0s and 1s).

- **Machine Code**: Not human-readable. It is a series of binary digits (bits) that represent specific operations for the CPU to execute.

- **Machine Code**: This is the final output after the source code is translated. It is directly executable by the computer.

- **Machine Code**: Not written directly by humans (except in very rare cases, such as low-level assembly programming). It is generated automatically by compilers or assemblers.

- **Machine Code**: The lowest level of code, as close to the hardware as possible. Each machine code is specific to a particular type of CPU architecture (e.g., x86, ARM).

- **Machine Code**: Not portable. It is specific to a particular machine or processor architecture. Machine code generated for one CPU type won't work on another unless the architectures are compatible.

## 27. GitHub and Introductions

**GitHub** is a web-based platform for version control and collaboration that allows developers to manage and share code. It is built around **Git**, a distributed version control system, and provides additional features like issue tracking, project management, and team collaboration. GitHub is widely used by both individual developers and large organizations to host code repositories, contribute to open-source projects, and manage software development processes.

## 28. Why is Version control in important in software development?

-Version control is crucial in software development for several reasons, as it ensures that code is effectively managed, tracked, and maintained over time. Here are the key reasons why version control is important:

## 29. What are the benefits of using Github for students

- Github offer a free tier that gives you to access ultimate the public and private repositories that you can as resource when you are learning to code

- GitHub is home to millions of open-source projects. Students can explore, learn from, and contribute to these projects, helping them gain hands-on experience while working with real-world codebases.

- GitHub provides the **GitHub Student Developer Pack**, which gives students access to free or discounted tools, resources, and services that are essential for learning to code and developing projects. This pack includes resources for cloud computing, code hosting, project management, design, and more

- Regularly committing code to GitHub repositories encourages students to code more often, helping them improve their coding and problem-solving skills.

## 30. Types of software

- ❖ Application software
- ❖ System Software
- ❖ Driver Software
- ❖ Middleware Software
- ❖ Programing Software

## 31. What are the differences between open-source and proprietary software?

| Aspect | Open-Source Software | Proprietary Software |
|---|---|---|
| Source Code Accessibility | Publicly available | Not available |
| Licensing | Open-source licenses | Restrictive licenses |
| Cost | Usually free | Often requires a purchase or subscription |
| Development | Community-driven | Controlled by a single organization |
| Flexibility | Highly customizable | Limited customization |
| Support | Community and optional professional | Professional support by vendor |
| Security | Peer-reviewed, quick patches | Vendor-managed, potential delays |

## 32. How does GIT improve collaboration in a software development team.

### 1. Distributed Version Control

- **Function:** Git is a distributed version control system, meaning each developer has a full copy of the repository, including its history, on their local machine.
- **Benefit:** This allows developers to work independently and offline, committing changes locally and synchronizing with the team when convenient.

### 2. Branching and Merging

- **Function:** Git supports lightweight branching and merging, enabling developers to create branches for new features, bug fixes, or experiments.
- **Benefit:** This allows parallel development efforts without disrupting the main codebase. Merging integrates changes back into the main branch seamlessly.

### 3. Pull Requests

- **Function:** Pull requests (also known as merge requests) are a way to propose changes to the codebase and request reviews from team members.
- **Benefit:** This facilitates code review, feedback, and collaboration, ensuring code quality and collective ownership of the codebase.

### 4. Commit History

- **Function:** Git maintains a detailed history of all changes made to the repository, including who made the changes, when, and why (with commit messages).

- **Benefit:** This provides transparency and accountability, making it easier to track progress, understand the rationale behind changes, and identify potential issues.

### 5. Conflict Resolution

- **Function:** Git helps manage and resolve conflicts that arise when multiple developers make changes to the same part of the codebase.
- **Benefit:** Conflict resolution tools and workflows ensure that changes are integrated smoothly, minimizing disruptions and maintaining a cohesive codebase.

### 6. Collaboration and Contribution

- **Function:** Git enables collaboration through shared repositories, forks, and cloning. Developers can contribute to projects, both internal and open-source, by submitting pull requests.
- **Benefit:** This fosters a collaborative environment, where team members can share knowledge, contribute to each other's work, and collectively improve the project.

### 7. Continuous Integration/Continuous Deployment (CI/CD)

- **Function:** Git integrates with CI/CD tools to automate the process of building, testing, and deploying code changes.

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

## INTRODUCTION:

Application software consists of programs designed to carry out specific tasks for the end-user. Unlike system software, which manages the hardware and system resources, application software is focused on user-centric activities.

## 1.Word Processing software :

- Word processing software is used for creating, editing, formatting, and printing documents.

- Automates formatting (fonts, margins, page numbers, etc.), reducing manual work.

- Provides templates for various document types (resumes, reports, letters), speeding up document creation.
- Allows real-time collaboration (e.g., Google Docs), making it easier for teams to work together.
- Offers advanced tools like grammar/spell check, version history, and commenting, ensuring high-quality output with less effort.

## 2. Spreadsheet software :

- Spreadsheet software allows users to organize, analyze, and store data in tabular form, often with advanced computation capabilities.

• Automates calculations with built-in formulas and functions (e.g., SUM, AVERAGE, VLOOKUP), reducing errors in financial and data analysis.

• Features like conditional formatting and data visualization (charts, graphs) make complex data easily understandable.

• Supports data sorting, filtering, and pivot tables to quickly extract meaningful insights from large datasets.

• Enables collaboration (e.g., Google Sheets), allowing multiple users to contribute to and edit data in real time.

## 3. Presentation software :

- Presentation software is used to create visually engaging slideshows for information dissemination, meetings, and instructional purposes.

• Offers design templates and themes to quickly create professional presentations without design expertise.

• Facilitates multimedia integration (images, videos, audio), making presentations more dynamic and informative.

• Provides tools like speaker notes, transitions, and animations to enhance audience engagement.

• Enables remote collaboration and easy sharing, allowing multiple team members to work on a single presentation.

## 4. Database Management :

- DBMS applications are used to create, manage, and manipulate databases, which store and organize large volumes of data.

• Provides a structured environment for storing and retrieving information, reducing redundancy and improving data accuracy.

• Automates data queries and reporting, saving time in generating insights.

• Supports data security features like encryption and user authentication, ensuring data integrity.

• Enhances collaboration by allowing multiple users to access and work with the data concurrently.

## 5. Project Management software :

- Project management software assists in planning, executing, and monitoring projects, offering tools to manage tasks, resources, and timelines.

• Centralizes project information, providing a single source of truth for team members.

• Tracks deadlines, milestones, and dependencies, ensuring that tasks are completed on time.

• Facilitates collaboration by allowing teams to assign tasks, track progress, and communicate within the software.

• Automates notifications, file sharing, and progress reports, reducing the need for time-consuming status meetings.

## 6. Graphic Design software :

- Graphic design software allows users to create, edit, and manipulate visual content, including images, illustrations, and layouts.

• Provides pre-built templates, fonts, and stock images to expedite the design process.

• Supports advanced editing tools that allow designers to fine-tune visuals with precision.

• Enables integration with other applications (e.g., exporting designs directly to web development platforms), streamlining the workflow.

• Collaboration features allow multiple users to work on a single design project and share feedback efficiently.

## 7. Communication and Collaboration software :

- Communication software enables real-time or asynchronous communication between individuals or groups, often across distances.

• Facilitates remote work through instant messaging, voice/video calls, and screen sharing, allowing teams to stay connected and collaborate.

• Reduces the need for lengthy email chains and in-person meetings, allowing faster decision-making.

• Supports file sharing, group chats, and integration with other productivity tools (e.g., project management software).

• Ensures continuity of communication across time zones, enabling global teams to function efficiently.

## 8. **Accounting software :**

- Accounting software automates financial management tasks like invoicing, bookkeeping, and payroll processing.

- Automates tasks like generating financial reports, tracking expenses, and reconciling bank statements, reducing manual data entry.
- Reduces the chance of human error in financial calculations, improving accuracy.
- Integrates with banking systems and tax preparation tools, simplifying workflows.
- Provides real-time financial insights that help businesses make informed decisions quickly.

## 34. What is the role of application software in business ?

• **Automation of Business Processes** – Reduces manual work by automating repetitive tasks like payroll processing, invoicing, and inventory management.

• **Improved Communication and Collaboration** – Enables teams to communicate effectively through email, chat applications, and collaboration tools like Microsoft Teams or Slack.

• **Data Management and Analysis** – Helps businesses collect, store, and analyze data using tools like databases (SQL, MongoDB) and business intelligence software (Power BI, Tableau).

• **Customer Relationship Management (CRM)** – Enhances customer interactions and sales tracking through CRM software like Salesforce and HubSpot.

• **Financial and Accounting Management** – Streamlines accounting, budgeting, and financial reporting with software like QuickBooks and SAP.

• **Marketing and Sales Support** – Assists in digital marketing campaigns, email marketing, and sales automation through platforms like Google Ads, Mailchimp, and Shopify.

• **E-commerce and Online Transactions** – Facilitates online sales, transactions, and inventory management through platforms like Shopify, Woo Commerce, and Magento.

• **Project Management and Productivity** – Helps in task tracking, time management, and project coordination using tools like Trello, Asana, and Monday.com.

• **Security and Compliance** – Protects business data and ensures regulatory compliance with cyber security software, antivirus programs, and data encryption tools.

• **Human Resource Management (HRM)** – Simplifies recruitment, payroll, and employee performance tracking using HR software like BambooHR and Workday.

## Software development process?

## Create Flow chart representing the software Development Life Cycle (SDLC)

- Planning

- Designing

- Implementation

- Testing

- **Maintenance**

## What are the main stage of the software development process?

### 1. Planning

- Define project goals and feasibility
- Estimate costs, time, and resources
- Identify risks and mitigation strategies

### 2. Requirements Analysis

- Gather and document user and system requirements
- Define functional and non-functional requirements
- Create Software Requirement Specification (SRS)

### 3. Design

- Define system architecture and components
- Create high-level and low-level design documents
- Plan database structures, user interfaces, and workflows

### 4. Implementation (Coding)

- Convert design into actual software by writing code
- Follow best coding practices and use version control
- Develop modules and integrate them

### 5. Testing

- Perform unit, integration, system, and acceptance testing
- Identify and fix bugs and vulnerabilities
- Ensure software meets requirements

### 6. Maintenance

- Fix bugs and apply updates
- Enhance software with new features
- Ensure system stability and security

## Software Requirement

## Write a requirement specification for a simple library management system?

### 1 Introduction

- The purpose of this document is to define the functional and non-functional requirements for a simple Library Management System (LMS). The system aims to automate library operations such as book management, member management, and lending transactions

- **LMS**: Library Management System
- **Admin**: Library administrator with full system privileges
- **Librarian**: Staff responsible for book management and lending
- **Member/User**: Registered library users who can borrow and return books

### 2 Functional requirement

- Users should be able to register with the system by providing personal details.
- Users should be able to log in and log out securely.
- Admin should be able to manage user roles and permissions.

### 3 Non-functional requirement

- The system should handle multiple concurrent users efficiently.
- Searches and updates should be processed within a reasonable time.

### 4 System Constraints

- The system will use a relational database for storing records.
- It will be a web-based application accessible through standard browsers.

### 5 Assumptions and Dependencies

- Users will have stable internet access for system usage.
- The system will integrate with email services for notifications.

### 6. Feature Enhancement

- Implement barcode scanning for book check-in/check-out.
- Introduce a digital library for e-books.

## Why is the requirement analysis phase critical in software development?

The **Requirement Analysis** phase is critical in software development because it lays the foundation for the entire project.

• Ensure that all stakeholders (clients, users, and developers) have a shared understanding of what the software must achieve.

• Help prevent misunderstandings that can lead to project scope creep or unmet expectations.

• Identifies potential issues and conflicts early, reducing the risk of costly changes in later phases.

• Allows developers to create a realistic project timeline and budget.

• Captures essential features (functional requirements) and system attributes like security, scalability, and performance (non-functional requirements).

- Ensure that the software meets user needs and industry standards.
- Bridges the gap between business needs and technical implementation.
- Encourages collaboration among developers, testers, designers, and clients.
- A well-defined requirement leads to a well-structured design, reducing bugs and rework.
- Help ensure that the final product aligns with user expectations and business goals.
- Provides clear criteria for testing to ensure the software functions as expected.
- Makes verification and validation easier during the later stages.

## Software Analysis

## Perform a functional analysis for an online shopping system?

- A **functional analysis** identifies the key functions and processes of a system to ensure it meets user needs effectively. Below is a structured breakdown of the functional requirements of an **online shopping system**.

### 1 User Role

- **Customer**
- **Admin**
- **Vendor (Optional)**
- **Guest User (Optional)**

### 2 Core Functional Requirement

- New users can register with email, phone number, or social media.
- Existing users can log in with credentials.
- Password reset and account recovery options

- View and edit personal details.
- Change password.
- View order history.
- Manage saved addresses and payment methods.
- Customers can browse and shop.
- Admins can manage products, users, and orders.
- Vendors (if applicable) can manage their own products and inventory.
- Browse product categories.
- Filter by price, brand, rating, etc.
- Search functionality with keyword suggestions.
- Display product images, descriptions, prices, and specifications.
- Show reviews and ratings.
- Availability status (in stock, out of stock).
- Users can add/remove products from their wish list.
- Option to share or move wish list items to the cart
- Add, update, or remove items from the cart.
- Show price breakdown (subtotal, taxes, discounts).
- Select shipping address.
- Choose payment method (Credit Card, PayPal, etc.).
- Apply coupons or promo codes.
- Secure payment gateway integration.
- Order summary with estimated delivery date.
- Generate an order receipt.

- View past and current orders.
- Order tracking (Processing, Shipped, Delivered).
- Request returns or exchanges.
- Refund processing options.
- Customers can rate and review products.
- Admin moderation for inappropriate content.
- Live chat, email, or phone support.
- Frequently Asked Questions (FAQs) section.
- Email/SMS notifications for order status updates.
- Promotional offers and personalized recommendations.
- Manage users, orders, and inventory.
- Generate sales reports and analytics.
- Add, update, or remove product listings.
- Track stock levels and set alerts for low inventory.
- Approve vendor registrations.
- Monitor vendor performance.

## 3 Non-functional requirement

- **Security:** Secure user authentication, data encryption, and fraud prevention.
- **Scalability:** System should handle high traffic during peak shopping seasons.
- **Performance:** Fast page loads and smooth transactions.
- **Usability:** Responsive design for mobile and desktop users.
- **Compliance:** Adherence to data protection laws (GDPR, PCI DSS).

## What is the role of software analysis in the development process

• Gathers functional and non-functional requirements from stakeholders.

• Helps translate business needs into clear, structured specifications.

• Prevents misunderstandings that can lead to costly revisions later.

• Determines the **technical, financial, and operational** feasibility of the project.

• Identifies system limitations and potential risks early.

• Helps in selecting the right technologies and architectural approach.

• Provides a blueprint for how the system will function.

• Guides database design, user interface design, and system interactions.

• Ensures that the software structure is **scalable, maintainable, and efficient**.

• Early detection of **inconsistencies, ambiguities, and missing requirements** reduces defects in later phases.

• Helps ensure that the software meets business and user expectations.

• Leads to a structured **testing strategy**, reducing debugging time

• Provides clarity on project scope, timeline, and resource requirements.

• Helps in creating a **realistic budget and delivery schedule**.

• Assists in assigning roles and responsibilities within the development team.

• Generates comprehensive documentation that supports development, testing, and future maintenance.

• Ensures compliance with industry standards, regulations, and security protocols.

• Helps new developers understand system requirements quickly.

• Acts as a reference point for developers, testers, and stakeholders.

• Reduces **misinterpretation** of project requirements

## <u>System Design</u>

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

### 1 Architecture Overview

- **Customers** (who order food)
- **Restaurants** (who prepare food)
- **Delivery Personnel** (who deliver food)
- **Frontend** (Mobile & Web Apps)
- **Backend** (APIs, Databases, Business Logic)
- **Third-party Services** (Payment, Maps, Notifications)

### 2. System components

#### A. Fronted User(User Interface)

- Browse restaurants & menus
- Place & track orders
- Make payments
- Rate & review
- Manage orders
- Update menu & availability
- Accept & track deliveries
- Navigation assistance

### B. Back-end ( server & Database )

- **API Gateway** (Handles requests from the frontend)
- **Authentication Service** (User login, OAuth, JWT)
- **Order Management Service** (Processes orders)
- **Payment Gateway Integration** (Stripe, PayPal)
- **Restaurant Management Service** (Menus, availability)
- **Delivery Assignment Service** (Assigns delivery personnel)
- **Notification Service** (SMS, Push, Email)
- **Geo-location & Routing Service** (Google Maps, Open Street Map)

### C. Database (Data Storage)

- **SQL Databases** (Orders, Users, Payments)
- **No SQL (Redis, MongoDB)** (Real-time order tracking)
- **Cloud Storage** (User profile pics, menu images)

## 3. Data Flow

- **Customer places an order → API processes request → Restaurant accepts → Payment processed**
- **Order is assigned to a delivery person → Real-time tracking updates**
- **Customer & Restaurant get notifications at each stage**
- **Delivery completed → Customer rates service**

## 4.Tech Stack

- **Frontend**: React Native / Flutter (Mobile), React.js / Next.js (Web)
- **Backend**: Node.js / Django / Spring Boot
- **Database**: PostgreSQL, MongoDB, Redis (Caching)

• **Cloud**: AWS / Firebase

• **Payments**: Stripe / Razorpay

• **Maps & Routing**: Google Maps API

## What are the key element of system design?

### 1 High-Level Design (HLD)

• **Monolithic vs. Micro services**: Decide whether the system will be a **single unified application** (monolith) or **multiple independent services** (micro services).

• **Client-Server Model**: Defines how different components communicate (e.g., web clients, mobile apps, backend servers).

• **Horizontal Scaling**: Adding more servers/machines to handle increased load.

• **Vertical Scaling**: Increasing server capacity (CPU, RAM).

• **Load Balancing**: Distributing traffic across multiple servers (e.g., Nginx, AWS ELB).

• **Databases**: Choosing between **SQL (relational)** and **No SQL (non-relational)**.

• **Replication & Sharding**: Ensuring availability and performance by distributing data.

• **Caching**: Using **Redis** or **Memcached** to reduce database load.

• **REST vs. GraphQL**: Choosing an API style for communication.

• **Rate Limiting**: Preventing API abuse (e.g., using API Gateway).

• **Authentication**: Using JWT, OAuth, or API keys.

• **Redundancy**: Having backup servers to prevent downtime.

• **Failover Mechanisms**: Switching to backup systems when failures occur.

• **Circuit Breaker Patterns**: Preventing cascading failures in distributed systems.

## 2 Low-Level Design

- T**ables, relationships, indexes** for efficient data retrieval.
- **Normalization vs. Denormalization** based on read vs. write-heavy needs.
- **Data Encryption**: Protecting sensitive information (e.g., AES, TLS).
- **Access Control**: Role-based (RBAC) or attribute-based (ABAC) permissions.
- **Logging & Monitoring**: Detecting and preventing attacks

## Software Testing

### 1. Functional Test Cases

| Test Case ID | Test Description | Input | Expected Output |
|---|---|---|---|
| TC_01 | Addition of two positive numbers | `5 + 3` | `8` |
| TC_02 | Addition of positive and negative number | `5 + (-3)` | `2` |
| TC_03 | Addition of two negative numbers | `-5 + (-3)` | `-8` |
| TC_04 | Subtraction of two positive numbers | `10 - 5` | `5` |
| TC_05 | Subtraction of a larger number from a smaller number | `5 - 10` | `-5` |
| TC_06 | Multiplication of two positive numbers | `4 * 2` | `8` |
| TC_07 | Multiplication involving zero | `4 * 0` | `0` |
| TC_08 | Multiplication of two negative numbers | `-4 * -2` | `8` |
| TC_09 | Division of two positive numbers | `10 / 2` | `5` |
| TC_10 | Division resulting in a decimal | `10 / 3` | `3.33` |
| TC_11 | Division by zero | `5 / 0` | Error / Exception |
| TC_12 | Division of zero by a number | `0 / 5` | `0` |

## Why is software testing important?

- **Reliability** – The software consistently works as expected.
- **Performance** – The system runs efficiently under various conditions.
- **Security** – Prevents vulnerabilities like data leaks or cyber threats.
- Bugs and errors can cause **system crashes, security breaches, or financial losses**.
- Testing helps catch issues **before deployment**, reducing the cost of fixing them later.
- Early bug detection leads to **faster development cycles** and better product stability.
- Fixing bugs **after release** is **10x more expensive** than fixing them during development.
- Automated testing **reduces manual effort** and speeds up the testing process.
- Reducing downtime due to failures **saves businesses money** and maintains reputation
- Software should be **user-friendly and responsive**.
- Poor performance or crashes lead to **customer dissatisfaction** and negative reviews.
- Thorough testing ensures **smooth navigation, quick load times, and minimal errors**.
- Cyber security threats are increasing—testing helps prevent data breaches.
- Security testing identifies **vulnerabilities like SQL injection, XSS, or authentication flaws**.
- Many industries (finance, healthcare) **require compliance** with regulations
- Load testing ensures **the system can handle high user traffic**.
- Stress testing checks how the system behaves **under extreme conditions**.
- Optimized performance helps businesses scale without failures.
- Continuous testing supports **Agile development** by ensuring frequent updates are bug-free.
- In **DevOps**, automated tests are integrated into CI/CD pipelines for **seamless deployments**.
- This reduces the risk of introducing new defects with each release.

# Maintenance

## Document a real-world case where a software application required critical maintenance.

### Problem: MCAS Software Malfunction

- The MCAS was designed to prevent the aircraft from stalling by automatically adjusting the nose downward if it detected a high **angle of attack (AoA)**.

- Due to a **faulty sensor reading**, MCAS repeatedly forced the aircraft into a **nose-down position**, making it difficult for pilots to regain control.

• **Software bugs in critical systems (e.g., aviation, healthcare) can be life-threatening**.

• **Relying on a single-point failure (like one sensor) is risky**—redundancy is crucial.

• **Thorough testing, pilot training, and transparency could have prevented the disaster**.

• **Multiple sensor input**: Instead of relying on a **single faulty AoA sensor**, the system was updated to **use data from both sensors**.

• **Limited MCAS Activation**: The system was modified so that MCAS would **activate only once per event**, preventing repeated nose-down commands.

• **Pilot Override Capability**: Pilots were given **full manual control** over MCAS, ensuring they could counteract any unwanted movements.

## What types of software maintenance are there?

• Focuses on **fixing errors, bugs, and defects** found after the software is deployed.

• Issues may include **logic errors, crashes, incorrect calculations, or security vulnerabilities**.

• Example: A bank's mobile app has a bug that **miscalculates interest rates**—developers release a patch to correct it.

• Involves modifying software to **adapt to changes in the operating environment** (e.g., new OS versions, hardware updates, API changes).

• Necessary when external factors like **third-party integrations, cloud services, or regulatory requirements** change.

• Example: A web application needs updates to remain compatible with **a new version of Google Chrome**.

• Focuses on **proactively identifying and resolving potential future issues** to improve system reliability.

• Includes **code refactoring, security updates, database optimization, and removing obsolete dependencies**.

• Example: A cyber security update is applied to a financial app to **protect against a newly discovered security vulnerability**.

## Development

## What are the key differences between web and desktop applications

• Hosted on remote servers and accessed through a web browser. No installation is required on the user's device.

• Installed and run on a local computer. Users must download and install the software.

• Can be accessed from any device with an internet connection and a compatible browser.

• Limited to the device on which they are installed.

• May experience slower performance due to reliance on internet speed and server load.

• Typically faster since they use the device's local resources directly

• Require an internet connection to function, although some may have offline capabilities.

• Do not require an internet connection to operate once installed.

• More vulnerable to cyber threats, such as hacking and data breaches, as data is stored in the cloud.

• Offer better security for sensitive data since information is stored locally and less exposed to external threats.

• Updates are automatic and handled by the provider.

• Users must manually install updates, which can be inconvenient.

• Generally platform-independent, as they run on browsers.

• Often platform-specific (e.g., Windows, macOS, Linux), requiring separate versions for different OS.

• Have limited access to system resources and may not be as feature-rich as desktop apps.

• Can fully utilize system resources, offering better graphics, processing power, and offline capabilities.

## Web Application

## What are the advantage of using web applications over desktop applications?

• **Web Applications:** Hosted on remote servers and accessed through a web browser. No installation is required on the user's device.

• **Desktop Applications:** Installed and run on a local computer. Users must download and install the software.

• **Web Applications:** Can be accessed from any device with an internet connection and a compatible browser.

• **Desktop Applications:** Limited to the device on which they are installed.

• When accessibility and collaboration are priorities (e.g., cloud-based productivity tools).

• When reducing maintenance and update efforts is essential.

• When users need a lightweight, cross-platform solution.

• Data is usually stored in the cloud, reducing the risk of data loss due to hardware failure.

• Security updates and patches are applied by the provider, minimizing vulnerabilities.

## Designing

### What role does UI/UX design play in application development?

• Visual design (colors, typography, icons, animations).

• Layout and placement of buttons, forms, and menus.

• Aesthetic appeal and branding.

• User journey mapping (navigation flow).

• Interaction design (how users interact with elements).

• Usability testing and user feedback integration.

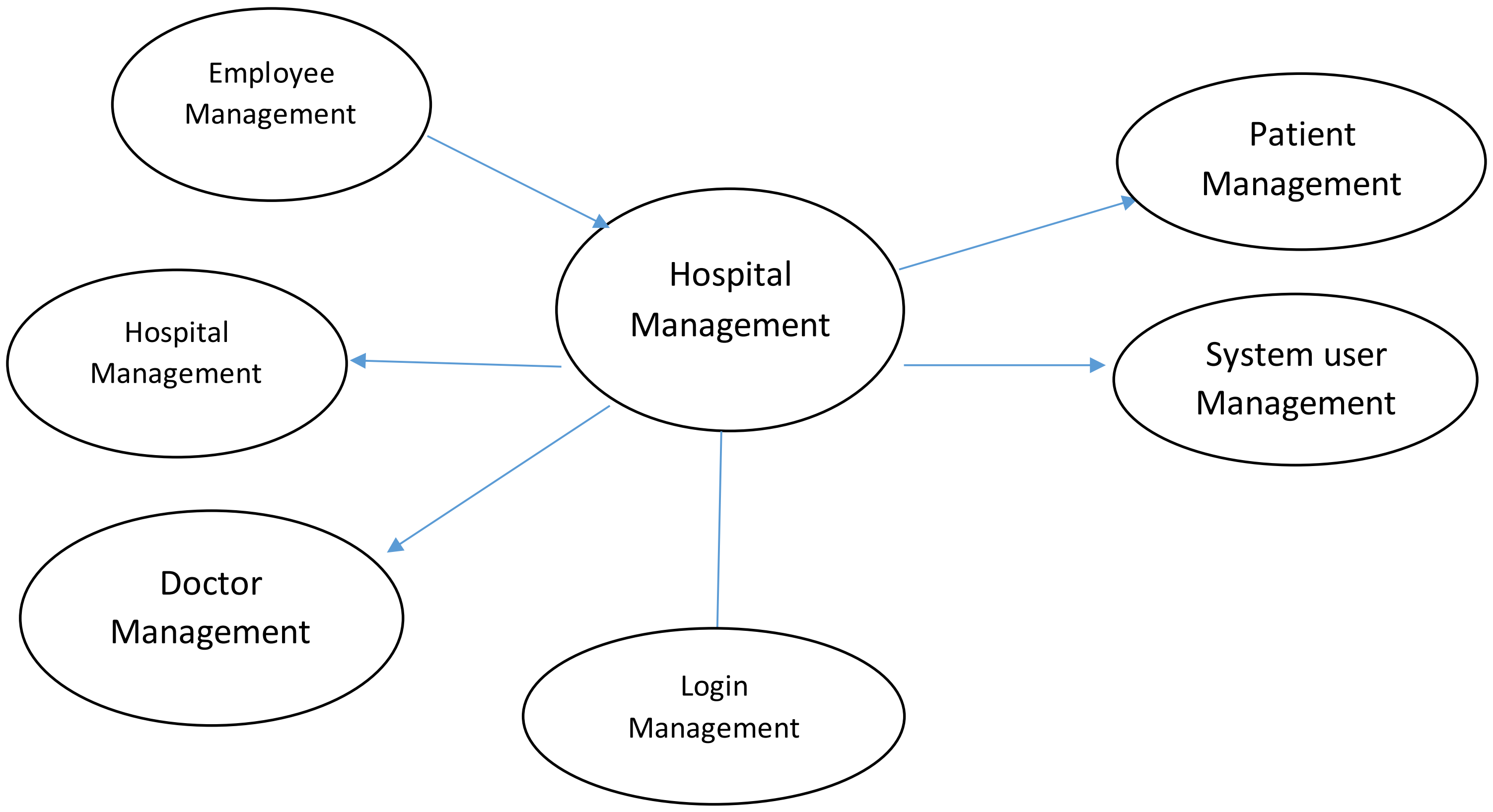
## Mobile Applications

### What are the differences between native and hybrid mobile apps?

| Factor | Native Apps | Hybrid Apps |
|---|---|---|
| **Performance** | ✓ Faster | ✕ Slightly slower |
| **Development Cost** | ✕ Higher | ✓ Lower |
| **Development Time** | ✕ Longer | ✓ Shorter |
| **User Experience** | ✓ Best | ⚠ Can be less fluid |
| **Access to Device Features** | ✓ Full access | ⚠ Requires plugins |
| **Cross-Platform Compatibility** | ✕ No | ✓ Yes |
| **Maintenance** | ✕ More effort | ✓ Easier |

# DFD (Data Flow Diagram)

## Create a DFD for a hospital management system



## What is the significance of DFD in system analysis

- **Level 0 (Context Diagram)** – Shows an **overview** of the system with external entities and a single process.

- **Level 1** – Breaks down the main process into **sub-processes** with data flows.

- **Level 2+** – Provides a **detailed breakdown** of each sub-process for deeper analysis.

A **Data Flow Diagram (DFD)** is a crucial tool in **system analysis** as it visually represents the flow of data within a system. It helps analysts, developers, and stakeholders understand how data moves through different processes, entities, and storage components.

## What are the pros and cons of desktop application campared to web applications

Desktop applications can leverage the full power of the local machine, providing faster performance and better access to hardware resources (e.g., GPU, CPU, RAM). This is crucial for resource-intensive applications like video editing, 3D rendering, or high-end gaming.

Desktop applications are typically bound to the machine they're installed on, limiting access. Users may need to carry their devices to access their apps and data.

Desktop applications are less prone to disruptions caused by internet connectivity issues or server downtimes, providing more reliable performance in some use cases.

Users must download, install, and update desktop applications manually or via third-party update systems. This can be cumbersome compared to web apps that are always up to date.

## **Flow chart**

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

```
+-------+
| Start |
+-------+
   |
   v
```

```
+--------------------------+
| User Enters Registration |
|       Details            |
+--------------------------+
            |
            v
    +-----------------+
    | Validate Inputs |
    +-----------------+
            |
        +---+---+
        |       |
        v       v
  [Invalid]   [Valid]
        |       |
        v       v
+---------------------+         +----------------------------+
| Display Error Message|         | Check if User Already Exists|
+---------------------+         +----------------------------+
        |                     |              |
        |                     v              v
        |             [User Exists?]    [User Does Not Exist?]
        |                     |              |
        |               Yes |            | No
        |                  v             v
        |         +---------------------+   +-------------------+
        |         | Display Error Message|   | Save User to       |
        |         +---------------------+   | Database          |
        |                  |            +-------------------+
        |                  |                   |
        |                  v                   v
        |            (Loop Back to)         +---------------------+
        |           User Enters Registration    | Display Success      |
        |                Details             | Message             |
        |                                 +---------------------+
        |                                       |
        |                                       v
        |                                   +-------+
        |                                   | End   |
        |                                   +-------+
        +-------------------------------------------------+
```

## How do flowcharts help in programming and system design?

Flowcharts break down complex algorithms or systems into simple, visual steps. This makes it easier to understand the logical flow and sequence of operations.

Flowcharts provide a common language for developers, designers, and stakeholders. They can quickly communicate the overall process and the flow of data, making it easier for teams to collaborate.

- Provide a clear, visual representation of complex processes.
- Improve communication and documentation among team members.
- Aid in debugging, testing, and decision-making.
- Support efficient planning and scalable design.