Name:- Limbasiya Vishvash B.

Modual 1 :- Overview of IT Industry

Q 1 : Explain in your own words what a program is and how it functions.

* A computer program is a sequence or set of instructions in a programming language for a computer to execute.

Q 2: What is Programming?

* Programming is a technological process that involves creating instructions for a computer to follow in order to solve problems.

Q 3: What are the key steps involved in the programming proces?

1. Identify the problem

2. Find a solution

3.Test the code

Q.4 : Types of Programming Languages.

1. **Procedural programming languages**
2. **Functional programming languages**
3. **Object-oriented programming languages**
4. **Logic programming languages**
5. **Frontend languages**
6. **Backend programming languages**

**Q 5:** What are the main differences between high-level and low-level programming languages?

| It is programmer friendly language. | It is a machine friendly language. |
| --- | --- |

|  |  |
| --- | --- |
| [High level language](https://www.geeksforgeeks.org/difference-between-assembly-language-and-high-level-language/) is less memory efficient. | [Low level language](https://www.geeksforgeeks.org/programming-language-generations/#:~:text=Two%20low%2Dlevel%20languages%20are%20Machine%20Language%20and%20Assembly%20Language.) is high memory efficient. |
| It is easy to understand. | It is tough to understand. |
| Debugging is easy. | Debugging is complex comparatively. |
| It is portable. | It is non-portable. |
| It can run on any platform. | It is machine-dependent. |
| It needs compiler or interpreter for translation. | It needs assembler for translation. |

Q .6: World Wide Web & How Internet Works.

WWW:  The WWW, users can access web pages over the internet by following a series of HTTP links.

How Internet Works : The internet allows users to access information, send messages, and interact with others online.

Q.7: Describe the roles of the client and server in web communication.

Client: Requests resources or services (e.g., web pages) from a server and displays or processes the response.

Server: Hosts resources or services, processes client requests, and sends the appropriate response.

Q.8: Network Layers on Client and Server.

1. **Application layer**

**2.Transport layer**

**3.Network laye**

**4.Physical layer**

**5.Session layer**

**6.Presentation layer**

**7.Data link layer**

**Q.9:** : Explain Client Server Communication.

* The client: Initiates the communication by sending a request to the server.
* The server: Processes the request, performs the necessary actions, and sends a response back to the client.
* The protocol: The TCP/IP protocol suite is typically used for client-server communication. The TCP protocol maintains the connection until the message exchange is complete.

**Q.10:** Types of Internet Connections.

* 1. **Fiber-optic**
  2. **Cable**
  3. **Satellite**
  4. **Wi-Fi**

**Q.11:** How does broadband differ from fiber-optic internet? Protocols.

1. Broadband

* Definition: A general term for high-speed internet, including DSL, cable, satellite, and fiber-optic.
* Technology: Uses existing infrastructure like copper telephone lines (DSL) or coaxial cables (cable).
* Speed: Moderate to fast, but slower than fiber-optic.

2. Fiber-Optic Internet

* Definition: A type of broadband that uses thin glass fibers to transmit data as light signals.
* Technology: Delivers faster speeds and higher bandwidth compared to traditional broadband.
* Speed: Very fast (up to 1 Gbps or more).

Q.12: What are the differences between HTTP and HTTPS protocols? Application Security.

1. Definition:
   * HTTP: Hypertext Transfer Protocol; standard protocol for transferring data over the web.
   * HTTPS: HTTP Secure; an extension of HTTP with added encryption for security.
2. Security:
   * HTTP: Data is transmitted in plain text, making it vulnerable to interception (e.g., man-in-the-middle attacks).
   * HTTPS: Uses SSL/TLS encryption, ensuring secure, encrypted communication between the client and server.

Application Security

* HTTPS ensures data confidentiality, integrity, and authentication, significantly improving security over HTTP and protecting users from threats like eavesdropping or tampering.

Q.13: What is the role of encryption in securing applications? Software Applications and Its Types.

Confidentiality: Prevents unauthorized access to sensitive data.

Integrity: Ensures that data is not altered during transmission.

Authentication: Verifies the identity of users or systems to prevent impersonation.

Non-repudiation: Guarantees that the sender cannot deny sending the data.

\*Software Applications types:

1.System Software

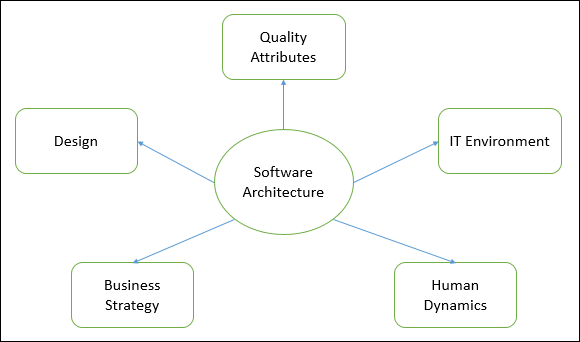
2.Application Software

3.Development Software

Q.14: What is the difference between system software and application software? Software Architecture.

|  |  |
| --- | --- |
| Low-level languages are used to write the system software. | While high-level languages are used to write the application software. |
| It is general-purpose software. | While it’s a specific purpose software. |
| Without system software, the system stops and can’t run. | While Without application software system always runs |

Software Architecture:



Q.15: What is the significance of modularity in software architecture? Layers in Software Architecture.

Modularity in Software Architecture refers to the practice of designing software as a set of separate, self-contained units or modules. This approach has several key benefits:

1. Maintainability: Modularity makes it easier to manage and modify individual components without affecting the entire system.
2. Reusability: Modules can be reused in different projects or parts of the system.
3. Scalability: New features or components can be added with minimal disruption to the existing architecture.
4. Flexibility: Modularity allows for easy updates or replacements of individual modules, fostering adaptability to changing requirements.

Layers in Software Architecture typically refer to organizing the system into different levels of abstraction, such as:

1. Presentation Layer: Handles user interaction (UI).
2. Business Logic Layer: Manages core functionalities and rules.
3. Data Layer: Deals with data storage and retrieval.

Q16:- Why are layers important in software architecture? Software Environments

Importance of Layers in Software Architecture:

1. Separation of Concerns: Layers help divide the system into distinct, manageable parts, each responsible for specific functions..
2. Modularity: By isolating functionalities in different layers, developers can update or modify one layer without affecting others, making the system more flexible and scalable.
3. Reusability: Layers allow components to be reused across different parts of the system or even in other systems, improving efficiency and reducing redundancy.
4. Testability: With clear separation between layers, each can be tested independently, ensuring better quality and easier debugging.

Software Environments:

A Software Environment refers to the configuration of hardware, software, and tools used to develop, test, and deploy applications. Key types include:

1. Development Environment: Where software is designed, coded, and tested during the development phase (e.g., IDEs, version control systems).
2. Testing Environment: A controlled environment where the software is tested to ensure it works as expected before deployment.
3. Production Environment: The live environment where the final version of the software is deployed for end users.

Q17:- Explain the importance of a development environment in software production. Source Code

Importance of a Development Environment in Software Production:

1. Efficient Coding: A well-configured development environment (IDE, text editors, etc.) provides tools that streamline coding, such as syntax highlighting, debugging, and auto-completion, which improve developer productivity.
2. Version Control: Development environments typically integrate with version control systems (e.g., Git), allowing teams to manage and track changes in source code, collaborate effectively, and avoid conflicts.
3. Testing and Debugging: Integrated testing and debugging tools help identify errors early, improving software quality and reducing time spent on bug fixes later.
4. Reproducibility: A consistent development environment ensures that all developers are working with the same setup, reducing the risk of "works on my machine" problems.
5. Collaboration: It allows multiple developers to work together on the same project seamlessly, with access to the same libraries, frameworks, and configurations.

Source Code:

Source code is the human-readable set of instructions written by developers using a programming language (like Python, Java, C++).

Q18:- What is the difference between source code and machine code? Github and Introductions

Difference Between Source Code and Machine Code:

* Source Code: This is the human-readable code written by programmers using a high-level programming language (e.g., Python, Java, C++).
* Machine Code: This is the low-level code, composed of binary (0s and 1s), that the computer's processor can directly understand and execute.

GitHub and Introductions:

* GitHub: GitHub is a web-based platform for version control and collaboration, using Git. It allows developers to host and share code repositories, track changes, and collaborate on projects.
* Introductions: In software development, "introductions" typically refer to initial explanations or overviews of concepts, technologies, or projects.

Q19:- : Why is version control important in software development? Student Account in Github.

Why Version Control is Important in Software Development:

1. Collaboration: Version control allows multiple developers to work on the same project simultaneously without overwriting each other's work.
2. History and Tracking: It keeps a history of all changes made to the code, allowing developers to track progress, identify bugs, and revert to previous versions if necessary.
3. Code Integrity: With version control, developers can manage and test changes in isolated branches, preventing incomplete or experimental changes from affecting the main codebase.
4. Backup and Recovery: It acts as a backup system, allowing recovery of lost or corrupted code from previous commits.
5. Collaboration with Remote Teams: Version control enables remote teams to collaborate on the same project, ensuring everyone works with the latest version.

Student Account in GitHub:

A Student Account on GitHub typically provides access to GitHub's paid features at no cost for educational purposes. GitHub offers GitHub Student Developer Pack, which includes free access to premium tools, cloud hosting, and resources for learning and collaboration. It's designed to help students with their coursework, projects, and learning to work with version control and collaborative coding environments.

Q20:- What are the benefits of using Github for students? Types of Software.

Benefits of Using GitHub for Students:

1. Collaboration: GitHub allows students to collaborate on coding projects with peers or instructors, making it easy to share code, review changes, and contribute to group assignments.
2. Version Control: It helps students track and manage changes to their code over time, ensuring they can revert to previous versions if needed and understand how their work evolves.
3. Free Access to Premium Tools: Through the GitHub Student Developer Pack, students gain free access to various tools and services that would otherwise be paid, like cloud hosting, IDEs, and project management tools.
4. Learning Industry Standards: GitHub is widely used in the software development industry. Using it in school prepares students for real-world coding practices, such as version control and project collaboration.
5. Showcase Projects: Students can use GitHub to host their personal portfolios, share projects, and demonstrate their skills to potential employers or professors.

Types of Software:

1. System Software: Software that manages hardware and provides a platform for running application software. Examples include operating systems (e.g., Windows, macOS) and device drivers.
2. Application Software: Programs designed to perform specific tasks for the user, such as word processors (e.g., Microsoft Word), web browsers (e.g., Google Chrome), and games.
3. Development Software: Tools used to create, test, and debug other software, such as programming languages (e.g., Python), Integrated Development Environments (IDEs like Visual Studio), and version control systems (e.g., GitHub).
4. Utility Software: Tools designed to perform maintenance tasks, such as antivirus programs, backup software, and disk management tools.

Q21:- What are the differences between open-source and proprietary software? GIT and GITHUB Training

Differences Between Open-Source and Proprietary Software:

1. Source Code Accessibility:
   * Open-Source Software: The source code is publicly available, allowing anyone to view, modify, and distribute it.
   * Proprietary Software: The source code is closed and owned by the developer or organization, and access to it is restricted.
2. Cost:
   * Open-Source Software: Typically free to use, though some may have paid support options.
   * Proprietary Software: Generally requires purchasing a license to use.
3. Customization:
   * Open-Source Software: Users can modify and customize the software to suit their needs.
   * Proprietary Software: Users are restricted to using the software as provided by the vendor, with limited customization options.

4.Support:

Open-Source Software: Support is often community-driven (forums, online resources), though paid support may be available.

* + Proprietary Software: Vendors typically offer professional support services, often included with the software license.

1. Security:
   * Open-Source Software: Security can be more transparent as anyone can audit the code, but it depends on community involvement for fixing vulnerabilities.
   * Proprietary Software: Security updates are managed by the vendor, though the code is closed, making it harder to audit independently.

GIT and GITHUB Training:

* GIT: GIT is a distributed version control system that helps track changes in code, manage different versions, and collaborate with others. GIT training typically covers topics like committing changes, branching, merging, and resolving conflicts.
* GITHUB: GitHub is a web-based platform that hosts Git repositories, allowing developers to share and collaborate on projects. GitHub training often includes setting up repositories, using pull requests, managing issues, and collaborating with teams.

Q22:- How does GIT improve collaboration in a software development team?

GIT improves collaboration in a software development team by:

1. Version Control: GIT tracks changes to code, allowing team members to work on the same project without overwriting each other's work.
2. Branching: Developers can create branches to work on features or fixes without affecting the main codebase.
3. Distributed System: GIT is decentralized, meaning every team member has a full copy of the repository.
4. Collaboration through Pull Requests: Team members can review each other's code by creating pull requests before merging changes, ensuring code quality and consistency.
5. Conflict Resolution: GIT helps identify and resolve code conflicts when merging changes from different developers, making collaboration smoother.

Q22:- What is the role of application software in businesses? Software Development Process

Role of Application Software in Businesses:

1. Increases Efficiency: Application software automates tasks, streamlining operations like data entry, inventory management, and accounting, helping businesses run more efficiently.
2. Improves Communication: Software tools like email clients, messaging apps, and project management systems enable better communication and collaboration among teams, even in remote settings.
3. Enhances Decision Making: Applications like data analytics and business intelligence software provide insights and help businesses make data-driven decisions.
4. Boosts Productivity: Business applications, such as word processors, spreadsheets, and CRM software, help employees work faster and more accurately.
5. Customer Relationship Management: Software like CRM (Customer Relationship Management) helps businesses manage and nurture customer relationships, leading to improved customer service and satisfaction.

Software Development Process:

The Software Development Process is a structured approach to creating software. Key phases include:

1. Planning: Defining the software's goals, requirements, and project scope.
2. Design: Creating the architecture and design of the software system.
3. Development: Writing the actual code and implementing the software.
4. Testing: Checking the software for bugs and ensuring it meets requirements.
5. Deployment: Installing the software in a live environment for users.
6. Maintenance: Ongoing updates and fixes after the software is deployed.

Q23:- What are the main stages of the software development process? Software Requirement.

Main Stages of the Software Development Process:

1. Requirement Gathering and Analysis: In this stage, the needs and requirements of the users or clients are collected and analyzed to define what the software should do.
2. System Design: Based on the requirements, the software architecture, components, and interfaces are designed to meet functional and non-functional requirements.
3. Implementation (Coding): The actual code is written according to the design specifications using appropriate programming languages and tools.
4. Testing: The software is tested for bugs, errors, and compliance with the requirements. This ensures the product works as intended.
5. Deployment: Once tested, the software is deployed in the live environment for users to access and use.
6. Maintenance: After deployment, the software undergoes regular updates, bug fixes, and improvements based on user feedback and evolving requirements.

Software Requirement:

Software requirements define the features, functionalities, and constraints of a software application. They are typically divided into:

1. Functional Requirements: Describe the specific behaviors or functions the system must perform (e.g., user login, data processing).
2. Non-Functional Requirements: Specify how the system performs its functions, such as security, performance, scalability, and usability.

Q24:- Why is the requirement analysis phase critical in software development? Software Analysis

Why the Requirement Analysis Phase is Critical in Software Development:

1. Defines Clear Goals: Requirement analysis helps clarify the purpose and goals of the software, ensuring that all stakeholders (clients, developers, users) have a shared understanding of what the software should do.
2. Prevents Misunderstandings: By gathering and documenting requirements early, it reduces the risk of misunderstandings or miscommunication between the development team and the client, leading to fewer changes later.
3. Sets Scope and Budget: It helps define the scope of the project, enabling better estimation of time, cost, and resources required for development.
4. Avoids Scope Creep: A well-conducted analysis prevents scope creep by setting clear boundaries for the project, ensuring that only necessary features are included.
5. Improves Software Quality: By addressing user needs and technical constraints from the start, the software is more likely to meet expectations, ensuring better functionality and quality.

Software Analysis:

Software analysis is the process of studying and understanding a software system's requirements, components, and behavior. It involves:

1. Gathering Requirements: Collecting detailed information about user needs, system requirements, and business goals.
2. System Design and Architecture: Analyzing the design and architecture of the system to ensure it meets the required specifications.
3. Problem Identification: Identifying any issues or gaps in the system that need to be addressed.

Q25:- What is the role of software analysis in the development process? System Design.

Role of Software Analysis in the Development Process:

1. Understanding Requirements: Software analysis ensures a deep understanding of user needs, business goals, and system requirements. It clarifies what the software should achieve and guides the entire development process.
2. Identifying Problems and Opportunities: It helps identify potential issues, inefficiencies, and opportunities for improvement early, reducing the risk of costly mistakes later in development.
3. Feasibility Study: Software analysis evaluates the feasibility of implementing requirements within given constraints (time, cost, technology), ensuring that the project is realistic and achievable.
4. Clear Communication: It creates a common understanding between stakeholders, including clients, developers, and users, ensuring everyone is aligned on goals and expectations.
5. Foundation for Design and Development: The insights from software analysis directly inform the design, architecture, and development phases, ensuring the right solutions are built from the start.

System Design:

System design is the process of defining the architecture, components, modules, and data flow of a software system. It translates the requirements and analysis into a blueprint for the system's construction. Key activities in system design include:

1. High-Level Design: Defining the overall architecture, including components, modules, and their interactions.
2. Detailed Design: Specifying the internal workings of each component, data structures, algorithms, and interfaces.
3. Design Specifications: Creating detailed documents or models that guide the development and implementation phases.

Q26:- What are the key elements of system design? Software Testing.

Key Elements of System Design:

1. Architecture Design: Defines the overall structure of the system, including its components, modules, and their interactions. It establishes how the system will function as a whole.
2. Component Design: Specifies the detailed design of each component or module, including data structures, algorithms, and interfaces.
3. Data Design: Involves defining how data will be stored, accessed, and processed within the system, including database design and data flow.
4. Interface Design: Defines the communication between different modules or external systems, ensuring smooth interaction and integration.
5. Security and Performance Considerations: Ensures that the system is secure, performs efficiently, and can handle expected loads and stress.
6. User Interface (UI) Design: Focuses on designing the front-end interface, ensuring that it is intuitive, user-friendly, and aligned with user requirements.

Software Testing:

Software testing is the process of evaluating and verifying that a software application works as expected and is free of bugs. It involves:

1. Unit Testing: Testing individual components or units of the software for correct functionality.
2. Integration Testing: Checking if different modules or components work together as expected.
3. System Testing: Testing the complete software system to ensure it meets all requirements and functions as intended.
4. User Acceptance Testing (UAT): Ensuring the software meets the end-user requirements and is ready for deployment.
5. Performance Testing: Evaluating the software's responsiveness, scalability, and stability under various conditions.

**Q27:-** Why is software testing important? Maintenance

Why Software Testing is Important:

1. Ensures Quality: Testing helps identify defects or bugs, ensuring the software meets quality standards and works as expected.
2. Reduces Costs: Catching issues early in the development process is less costly than fixing them after deployment. It prevents costly post-release fixes and damage to the product's reputation.
3. Improves Reliability: Regular testing ensures that the software functions correctly under different conditions, enhancing its reliability and performance.
4. Meets Requirements: Testing verifies that the software meets the original requirements and user expectations, leading to greater customer satisfaction.
5. Prevents Security Issues: Security testing helps identify vulnerabilities and weaknesses, preventing potential breaches and ensuring the software is secure.

Maintenance:

Software maintenance is the process of updating and improving software after it has been deployed. It involves:

1. Corrective Maintenance: Fixing bugs or issues discovered after the software is in use.
2. Adaptive Maintenance: Making updates to the software to adapt to new environments, operating systems, or technologies.
3. Perfective Maintenance: Enhancing the software's performance, features, or usability based on user feedback or evolving requirements.
4. Preventive Maintenance: Modifying the software to prevent future problems or to make it easier to maintain in the future.

Q28:- What types of software maintenance are there? Development

Types of Software Maintenance:

1. Corrective Maintenance: Involves fixing bugs or defects in the software that were discovered after deployment. This is aimed at resolving problems to restore the software’s correct functionality.
2. Adaptive Maintenance: Involves updating the software to ensure compatibility with new hardware, operating systems, or external systems. It helps the software remain functional as the environment changes.
3. Perfective Maintenance: Focuses on improving the software’s performance, usability, or adding new features based on user feedback or evolving business needs.
4. Preventive Maintenance: Aims at making changes to prevent future issues, such as refactoring code to improve maintainability or enhancing security to avoid potential risks.

Development:

Development refers to the process of creating software, which includes the stages of planning, designing, coding, testing, and deploying. It involves translating user requirements into functional code, ensuring the software is built to meet specific needs and is reliable, efficient, and scalable.

Q29:- What are the key differences between web and desktop applications?

Key Differences Between Web and Desktop Applications:

1. Platform:
   * Web Applications: Run in a web browser, making them platform-independent (can be accessed from any device with a browser).
   * Desktop Applications: Installed and run directly on a computer or device, typically designed for specific operating systems (e.g., Windows, macOS).
2. Access:
   * Web Applications: Accessible over the internet, requiring a network connection to function.
   * Desktop Applications: Do not require an internet connection (unless they involve cloud features); they work offline once installed.
3. Installation:
   * Web Applications: No installation required; users only need a web browser and internet access.
   * Desktop Applications: Must be installed on a local machine, taking up disk space.
4. Updates:
   * Web Applications: Updates are managed centrally by the server, and users always access the latest version.
   * Desktop Applications: Updates must be manually installed or triggered by the software itself.

Q30:- What are the advantages of using web applications over desktop applications?

Advantages of Using Web Applications Over Desktop Applications:

1. Accessibility: Web applications can be accessed from any device with an internet connection and a web browser, making them platform-independent.
2. No Installation Required: Users don’t need to install anything on their devices, reducing setup time and disk space usage.
3. Centralized Updates: Updates are handled centrally, meaning users always have access to the latest version without needing to manually update the software.
4. Cross-Platform Compatibility: Web apps work across different operating systems (Windows, macOS, Linux) without requiring separate versions.
5. Remote Collaboration: Web apps can support real-time collaboration and data sharing across teams, regardless of location.

Q31:- What role does UI/UX design play in application development?

Role of UI/UX Design in Application Development:

1. User Satisfaction: UI (User Interface) and UX (User Experience) design ensure the application is intuitive, easy to use, and visually appealing, leading to a better user experience and higher satisfaction.
2. First Impressions: A well-designed UI makes a positive first impression, encouraging users to engage with the app and continue using it.
3. Functionality and Usability: UI/UX design focuses on making the app functional and user-friendly, ensuring that users can easily navigate and perform tasks.

Q32:- What are the differences between native and hybrid mobile apps?

Differences Between Native and Hybrid Mobile Apps:

1. Platform Dependency:
   * Native Apps: Built specifically for a single platform (e.g., iOS or Android) using platform-specific programming languages (Swift for iOS, Kotlin/Java for Android).
   * Hybrid Apps: Developed using web technologies (HTML, CSS, JavaScript) and run inside a native container, allowing them to work across multiple platforms.
2. Performance:
   * Native Apps: Generally provide better performance since they are optimized for the specific platform and have direct access to device hardware.
   * Hybrid Apps: May have slower performance because they rely on a web view and are not fully optimized for each platform.
3. Development Cost and Time:
   * Native Apps: Require separate codebases for different platforms, which can increase development time and cost.
   * Hybrid Apps: Allow a single codebase for multiple platforms, reducing development time and cost.

Q33:- What is the significance of DFDs in system analysis?

Significance of DFDs in System Analysis:

1. Clear Visualization: DFDs help visualize how data moves through a system, making it easier to understand the system's processes, inputs, outputs, and data flows.
2. Requirement Clarification: They help identify and clarify system requirements by mapping out the current or proposed system, ensuring all stakeholders are aligned on its functionality.
3. Identifying System Boundaries: DFDs define the scope and boundaries of the system, showing what is inside and outside the system’s control, helping prevent scope creep.
4. Process Optimization: By breaking down the system into smaller components, DFDs highlight inefficiencies or redundancies in data flow, aiding in process optimization.
5. Documentation: DFDs provide a clear and standardized method of documenting the system, useful for future reference, maintenance, and enhancements.

Q34:- What are the pros and cons of desktop applications compared to web applications?

Pros and Cons of Desktop Applications Compared to Web Applications:

Pros of Desktop Applications:

1. Performance: Typically offer better performance as they run directly on the local machine, utilizing full hardware resources.
2. Offline Access: Can be used without an internet connection, providing flexibility in environments with limited or no connectivity.

Cons of Desktop Applications:

1. Platform Dependency: Need separate versions for different operating systems (Windows, macOS, Linux), increasing development and maintenance time.
2. Installation Required: Users must install and update the application on each device, which can be cumbersome.

Pros of Web Applications:

1. Cross-Platform: Work across multiple platforms (Windows, macOS, Linux) without the need for separate versions, reducing development time.
2. Easy Access: Accessible from any device with a web browser and an internet connection, offering greater flexibility.

Cons of Web Applications:

1. Dependent on Internet: Requires a stable internet connection for access and functionality.
2. Performance: May be slower compared to desktop apps due to reliance on the browser and server-side processing.

Q35:- How do flowcharts help in programming and system design?

How Flowcharts Help in Programming and System Design:

1. Visual Representation: Flowcharts provide a clear, visual representation of the system's processes and logic, making it easier to understand and design complex workflows.
2. Simplifying Logic: They break down complex algorithms or processes into simpler, step-by-step visual components, helping programmers understand and follow the flow of control.
3. Problem-Solving: Flowcharts help identify inefficiencies, bottlenecks, or potential issues in the system's logic early in the design phase.
4. Communication: Flowcharts make it easier to communicate ideas and logic between team members or stakeholders, ensuring everyone is aligned on the system's design.