Module 1 –Overview of the IT Industry

❖ What is a Program?

✓ A program is instructions for a computer to execute specific tasks.

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

❖ HTML:

```
<!DOCTYPE html>
<title>Hello World</title>
</head> <body>
 <h1>Hello, World</h1>
</body>
</html> C
language:
#include <stdio.h>
int main ()
{
print (“Hello, World! \ n”);
Return 0;
 }
```

HTML: HTML uses tags to define the structure of a webpage. The tags such as <html>, <head>, and <body> organize content, while <h1> is used to display a large heading.

C: C programs need a main() function as the entry point. The printf() function is used to print text to the console. C requires explicit instructions for how to handle output and execution flow, and the #include <stdio.h> preprocessor directive is necessary for input/output functionality.

❖ Explain in your own words what a program is and how it functions.

✓ A program is instructions for a computer to execute a specific task.

Programs are written in both low-level and high-level programming languages and are human-readable and writable.

Simple function:

1: Create a new project.
2: Design the web.
3: Coding the program.
4: Package the program.
5: Test and Run the program.

❖ What is Programming?

- ✓ Programming is creating a set of instructions that a computer can follow to perform specific tasks or solve particular problems.

❖ What are the key steps involved in the programming process?

- ✓ There are six types of programming process:

1. Planning
2. Analysis
3. Designing
4. Coding
5. Testing
6. maintenance

❖ Types of Programming Languages.

- ✓ Procedural programming language.

1. functional programming language.
2. object-oriented programming language.
3. Scripting programming language.
4. logic programming language.

❖ What are the main differences between high-level and low-level programming languages?

- ✓ High-level programming:

High-level languages are more abstracted from the computer's hardware and closer to human language.

High-level languages are often used for software development, web development, and database management.

Example: python, JavaScript, C, C++.

Low-level programming:

Low-level languages are typically used for system programming, device driver development, and embedded systems. Example: Machine language and Assembly language.

❖ World Wide Web & How the Internet Works.

- ✓ www: world wide web.

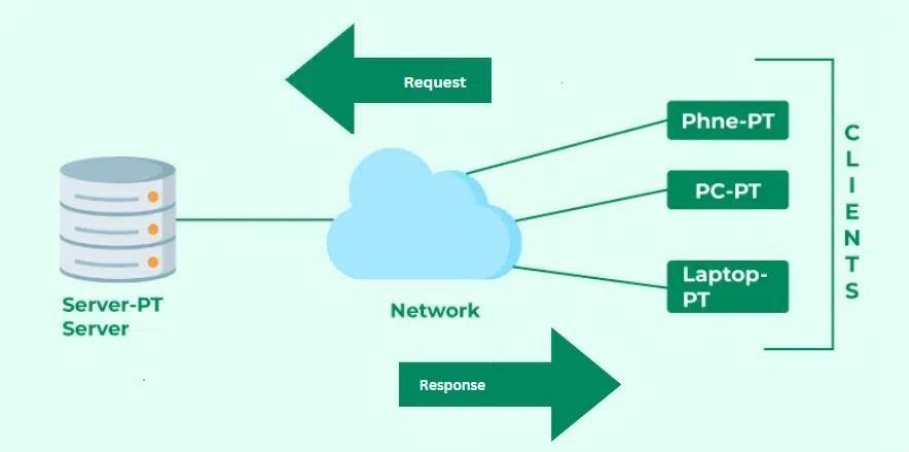
The World Wide Web is a network of interconnected websites and documents accessible via the Internet.

Internet works: The Internet is a worldwide network of interconnected computers and servers that share information, services, and resources.

1. Sending a Request
2. Routing Data
3. Server Response
4. Receiving Data

- LAB EXERCISE: Research and create a diagram of how data is transmitted from a client to a server over the Internet.
- The Client-server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters called clients.

In the client-server architecture, when the client computer sends a request for data to the server through the internet, Client-server model:



❖ Describe the roles of the client and server in web communication.

- ✓ The client is an application or device that requests services or resources from the server.

The server is a powerful computer or software application that stores and manages resources such as web pages, images, and databases, responding to requests made by clients.

❖ Network Layers on Client and Server.

- ✓ There are four layers.

Layer 1: Network Layer.
Layer 2: Internet Layer.
Layer 3: Transport Layer.
Layer 4: Application Layer.

- LAB EXERCISE: Design a simple HTTP client-server communication in any language.

- The server-side code will handle incoming HTTP requests and send back a response.

Client Code (HTML + JavaScript):

```
<!DOCTYPE html>
<head>
<title>HTTP Client</title>
```

```
</head>
 <body>
 <h1>HTTP Client-Server Communication</h1>
<button id="get Request Button">Send GET Request</button>
<p id="response"></p>
</body>
</html>
```

PHP Server (Backend):

Create a file called server. PHP on your web server (make sure your web server is running, e.g., using XAMPP, WAMP, or MAMP).

```
<?PHP
// Set the content type to plain text
Header ('Content-Type: text/plain');  //
Send a simple response message
echo "Hello from the server! This is a simple GET request response.";
 ?>
```

Header ('Content-Type: text/plain'): This sets the content type of the response to plain text.

echo "Hello from the server!": This outputs the response that will be sent to the client.

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

✓ TCP/IP (Transmission Control Protocol/Internet Protocol):

The main work of TCP/IP is to transfer the data of a computer from one device to another.

The TCP/IP model is a fundamental framework for computer networking.

There are four layers of the TCP IP model:

1. Application layer
2. Transport layer
3. Internet layer
4. Link layer

1. application layer:

- Provides network services directly to end-user applications.

Protocols: HTTP, FTP, SMTP, etc.
Example: Web browsers, Email clients.

2. transport layer:

- Ensures a dependable transfer of data between two devices.

Protocols: TCP, UDP.
Example: Email transfer, Video streaming.

3. internet layer:

- Routes data between devices located on different networks.

Protocols: IP, ICMP, IPv4, IPv6.
Example: Transmitting data between different networks.

4. Link layer:

- Handles the transmission of data through physical media.

Protocols: Ethernet, Wi-Fi, ARP.
Example: Ethernet cables, Wi-Fi.

❖ Client and Servers.

✓ The client-server model involves a client that requests services or resources from a server.

❖ Explain Client Server Communication.

✓ Clients and servers communicate by exchanging messages in a request response pattern.

The client sends a request, and the server returns a response.
This model is widely used in web applications, email systems, file sharing, and many other network-based services.
Types of client-server communication:

1. Request-Response Model:
   The client sends a request and waits for the server's response.
   This is used in web browsing and API calls.
2. Two-Way Communication:
   Both the client and server can send data to each other.

3. Peer-to-Peer (P2P) Communication:
   Although P2P networks do not follow a strict client-server model, they enable direct communication between devices.

❖ Types of Internet Connections.

- ✓ Shared network.
- ✓ DSL Internet connection.
- ✓ Fiber Internet.
- ✓ Cable Internet.
- ✓ Fixed Wireless.
- ✓ Satellite Networks.

- LAB EXERCISE: Research different types of internet connections (e.g., broadband, Fiber, satellite) and list their pros and cons.

1. Broadband Internet:

Broadband is a general term that refers to high-speed internet connections.

Pros:

Broadband offers high-speed internet, allowing for smooth streaming, gaming, and browsing.

Includes various technologies like DSL, fiber, and cable, giving customers multiple choices.

Cons:

Depending on your location, broadband might not be available or could be limited in terms of speed and reliability.

Broadband speeds can be affected during peak usage times, especially with shared technologies like cable.

2. Satellite Internet:

Pros:

Can be used in remote or rural areas where other types of internet (like fiber or cable) are not available.

Provides internet in most parts of the world, including places with poor infrastructure.

Cons:

Satellite internet is typically more expensive than DSL, cable, or fiber options.

❖ How does broadband differ from fiber-optic internet?

✓ Broadband:

- DSL, Cable, Satellite, Wireless (4G/5G).
- 1 Mbps to 1 Gbps (depending on tech).
- Moderate to High (depending on type).
- Widely available in urban areas.
- Generally lower.

Fiber optic internet:

- Fiber-optic cables (FTTH, FTTP).
- 100 Mbps to 10 Gbps.
- Low latency.
- High bandwidth, not affected by distance or usage.
- Generally higher (due to better performance).

❖ Protocols.

✓ Network protocols are rules that facilitate communication between devices in a network.

Types of protocols:

1. HTTP or HTTPS
2. FTP
3. Email Protocols
4. TCP

- LAB EXERCISE: Simulate HTTP and FTP requests using command line tools

(e.g., curl).

- Simulating HTTP and FTP requests using command-line tools such as curl is a great way to test.

Simulating HTTP Requests with curl:

The most common type of HTTP request is a GET request, used to retrieve data from a server. curl http://example.com.

A POST request is used to send data to the server, such as submitting a form or sending JSON data. curl -X POST -d "username=example & password=12345" http://example.com/login. Simulating FTP Requests with curl:
You can use curl to download a file from an FTP server using the -O (uppercase) option: curl -u username: password -O

❖ What are the differences between HTTP and HTTPS protocols?

✓ HTTP: hypertext transfer protocol.
In HTTP, the URL begins with "http://".
HTTP is considered to be insecure.
HTTP works at the Application Layer.
HTTP is faster than HTTPS.

HTTPS: hypertext transfer protocol secure.
In HTTPs, URL starts with "https://".
HTTPs are considered as secure.
HTTPS works at the Transport Layer.
HTTPS is slower than HTTP.

❖ Application Security.

✓ Application Security means designing, coding, and configuring your application to prevent and defend against cyber threats.

- LAB EXERCISE: Identify and explain three common application security vulnerabilities. Suggest possible solutions.

- SQL Injection
- Sensitive data exposure
- XML external entities (XXE)
- Cross-site scripting (XSS)
- Insecure deserialization
- Using components with known vulnerabilities • Insufficient logging and monitoring • Cryptographic failures.

1. **SQL Injection:**

SQL Injection occurs when an attacker can manipulate a web application's SQL queries by injecting malicious SQL code into user input fields.
This can lead to unauthorized access to or manipulation of the database, including data theft, data corruption, or even complete control over the system.

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

Cross-site scripting (XSS) occurs when an attacker can inject malicious scripts into web pages viewed by other users.
These scripts can be used to steal sensitive information redirect users **Cross-Site Request Forgery (CSRF):**
ross-site Request Forgery (CSRF) is an attack that tricks a user into unknowingly submitting a request to a web application where the user is authenticated, potentially performing unwanted actions on their behalf.

❖ What is the role of encryption in securing applications?

- ✓ Encryption is an effective and dependable method for protecting data by transforming it into a format that cannot be read.

Encryption protects sensitive information, like personal data, passwords, and financial transactions, by ensuring that only authorized users or systems have access to it.

❖ Software Applications and Its Types.

- ✓ Application software is a type of computer program that performs a specific personal, educational, and business function.

Types of software applications:

1. System Software
2. Application Software
3. Development Software 4. Security software
5. Driver software.

- LAB EXERCISE: Identify and classify 5 applications you use daily as either system software or application software.

1. Operating System (example: Windows, macOS, Linux):

- Classification: System Software
- Explanation: The operating system is responsible for managing hardware resources and providing essential services for computer programs.

2. Web Browser (example: Google Chrome, Mozilla Firefox):
   - Classification: Application Software
   - Explanation: Web browsers allow users to access and interact with content on the internet, such as websites, videos, and documents.
   - This is a program built to perform a specific task.
3. Text Editor (example: Microsoft Word, Google Docs):
   - Classification: Application Software
   - Explanation: Text editors are programs used to create and edit documents.
   - These applications are designed to help users perform specific tasks like word processing.

3. Antivirus Software (example: Norton, McAfee):
   - Classification: System Software
   - Explanation: Antivirus software is considered system software because it helps protect the system from malware, viruses, and other security threats.
   - It runs in the background and ensures the system remains secure.
4. File Management System (example: File Explorer, Finder):
   - Classification: System Software
   - Explanation: File management systems allow users to organize, manage, and navigate the files and folders on their devices.

❖ What is the difference between system software and application software?

✓ System software:

Manages hardware and provides a platform for running applications.

Low-level languages are used to write the system software.

Without system software, the system stops and can't run.

System software runs independently.

Application software:

Performs specific tasks for the user.

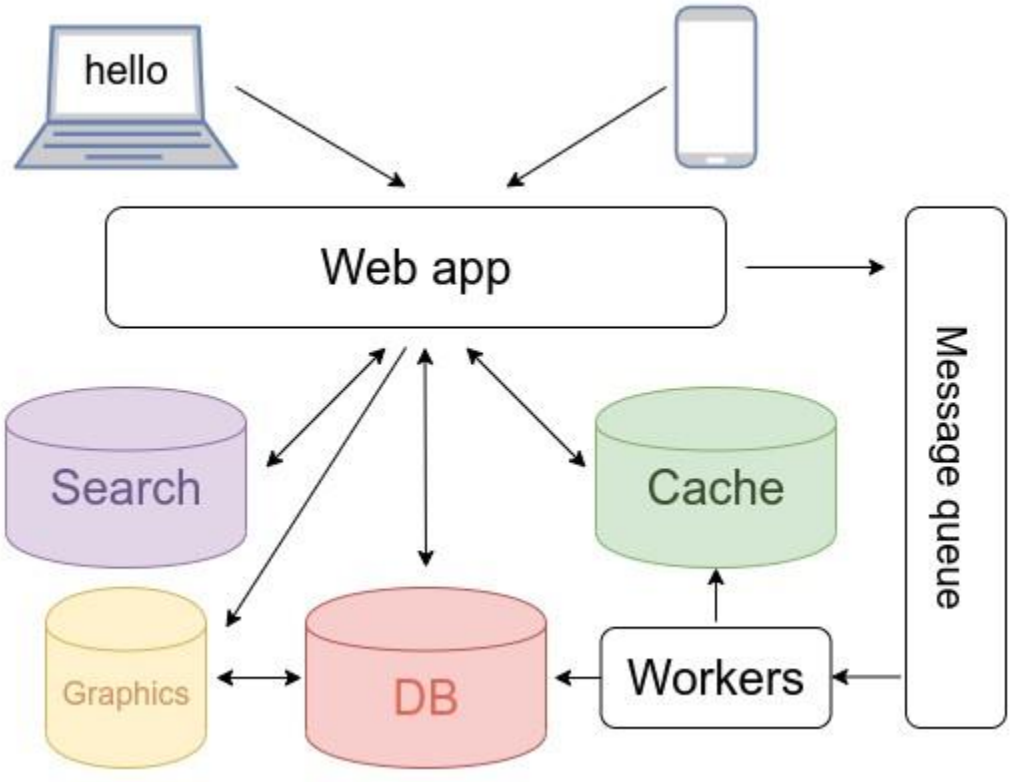
While high-level languages are used to write the application software.
While Without application software system always runs.

❖ Software Architecture.

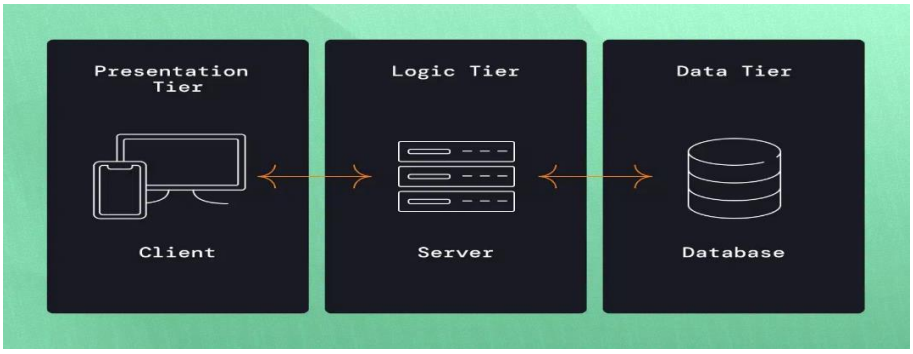
- ✓ Software architecture refers to the high-level structuring of software systems.

Software Architecture defines the fundamental organization of a system and more simply defines a structured solution.

Architecture:



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



1. Presentation Layer:
    - This is the topmost layer and interacts directly with the user.
    - It includes web pages, mobile interfaces, and other client-side elements that display data and accept user input.
    - Web browsers, HTML/CSS, JavaScript, React, Angular, or other frontend technologies.
2. Logic Layer:
    - This layer contains the business logic of the application.
    - The logic layer may also include API endpoints, which the client interacts with.
    - Web frameworks (e.g., Express, Django), application servers, business logic functions, RESTful APIs.
3. Data Layer:
    - his layer is responsible for data storage and retrieval.
    - It consists of the database system, which holds the data and allows for querying, updating, and deleting data.
    - Relational databases (MySQL, PostgreSQL), NoSQL databases (MongoDB), and data access objects (DAOs).

❖ What is the significance of modularity in software architecture?

✓ Modularity aims to improve software development by partitioning complex problems into more manageable sub-problems.

It allows for better code organization and readability.

By dividing a complex system into smaller modules, developers can create a more logical and structured codebase.

❖ Layers in Software Architecture.

✓ There are a total of five layers.

1. Presentation layer.
2. Application layer.
3. Business layer.
4. Persistence layer.
5. Database layer.

- LAB EXERCISE: Create a case study on the functionality of the presentation, business logic, and data access layers of a given software system.

1. Presentation Layer:

   Functions:
   - displays product listings with images, prices, and descriptions.
   - Provides search functionality to find products by name, category, or price range.
   - Allows users to view product details.
   - Provides access to the shopping cart, displaying selected items, quantity, and total price.
   - The technology used: HTML, CSS, JavaScript (React, Angular, or Vue.js).
2. Business Logic Layer:

   Functions:
   - Validates and processes data sent from the Presentation Layer, such as user registration or login.
   - Applies business rules to the user input.
   - Calculates prices, taxes, discounts, and shipping costs.
   - Technology used: Server-side languages and frameworks (example: Node.js, Django, Ruby).
3. Data Access Layer:

   Functions:
   - Updates inventory levels when products are purchased.

- Stores new orders in the database, including customer information, products ordered, and shipping details.
- Relational databases (MySQL, PostgreSQL) or NoSQL databases (MongoDB).

❖ Why are layers important in software architecture?

✓ Fundamental software architecture concepts help organize and structure complex systems by separating different concerns and responsibilities. Layers provide several advantages that enhance a system's maintainability, scalability, flexibility, and testability.

❖ Software Environments.

✓ . A software development environment is a collection of important tools and procedures that help developers to test and debug the software program. This environment supports software developers in designing, programming, and debugging software according to best practices.

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

1. Development Environment:

- The development environment is where developers write and initially test their code.
- It's used for building the software and conducting initial debugging.
- This environment often includes tools for code editing, version control, and local compilation.

2. Testing Environment:

- The testing environment is where the software is tested for bugs, performance, and security issues.
- This is where the software undergoes various tests like unit tests, integration tests, system tests, and user acceptance tests (UAT).

3. Production Environment:

- The production environment is where the software is deployed and accessed by end users.
- It must be stable, reliable, and efficient because any downtime or issues can directly affect users.

Set Up the Virtual Machine:

1. Download VirtualBox.
2. Download Ubuntu ISO.
3. Create a New Virtual Machine.
4. Install Ubuntu on the VM.

❖ Explain the importance of a development environment in software production.

- ✓ . A development environment is essential for software production, providing developers with the tools, resources, and settings to build, test, and debug software efficiently.

The software development environment must include all necessary hardware and software for creating, managing, and maintaining software throughout its lifecycle.

These environments:

- Play an integral role in software creation, management, and maintenance.
- Enable developers to do testing and verify that programs will function as expected.
- Help developers to make code changes in a controlled environment, without affecting users.

❖ Source Code.

- ✓ Source code is the set of instructions that a programmer writes to create software.

Instructions are written in various programming languages like Python, HTML, C++, or Java.

LAB EXERCISE: Write and upload your first source code file to GitHub.

.Upload Your First Source Code File to GitHub:

1. Create a GitHub Account:

- Go to GitHub and sign up for a free account.

2. Install Git on Your Local Machine:
   - Download Git from git-scm.com.
   - Follow the installation instructions.
3. Create a New Repository on GitHub:
   - Select New Repository.
   - Name your repository.
   - Choose to make it Public or Private.
   - add a description and initialize with a README file.
   - Click Create Repository.
4. Add Your Source Code File to Git:
   - git add index.js.
5. Link Your Local Repository to GitHub:
   - Go to your GitHub repository page.
   - Copy the repository URL.
6. Verify Your Code on GitHub:
   - Go to your repository page on GitHub.
   - You should see your index.js file listed in the repository with the commit message you provided.

❖ What is the difference between source code and machine code?

✓ . Source code:

Source code refers to high-level or assembly code that is created by programmers or humans.

Source code is easy to read and modify.

Source code is generated by humans or programmers.

Source code is high-level code.

Source code is written in plain text by using some high-level programming language.

It is written in a high-level language like C, C++, Java, Python, etc., or assembly language.

Example: print ("Hello, World!") (Python).

Machine code:

Machine code consists of binary instructions that are directly understandable by the CPU.

Machine Code is considered as the low-level code.

Represents actual executable instructions for the CPU.

Not understandable by humans.

Example: Binary instructions like 1101001110110001.

❖ GitHub and Introductions.

- ✓ Collaboration and version control are important for software development. GitHub has become a crucial platform for developers, facilitating teamwork and improving project management.

GitHub is an online platform that utilizes Git, a version control system, to assist developers in managing and tracking changes to their code.

GitHub is a web-based platform that helps Git host its repositories.

GitHub is a web-based graphical interface.

7. Create an Account:
   Go to GitHub and sign up for an account.
8. Create a New Repository:
   Once you are signed in, click the "New" button on your dashboard to create a new repository.
   You can choose to make it public or private.
9. Clone a Repository:
   In GitHub, each repository has a clone URL that you can use in Git to copy to your local machine.
   Example: git clone https://github.com/username/repositoryname.git.
10. Make Changes:
    You can modify the files. Use Git commands such as git add to stage changes and git commit to save them locally.
11. Push Changes:
    After committing your changes, use git push to send them to GitHub, allowing other contributors to view and merge them.

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

1. Create a GitHub Account:
    - Go to GitHub and sign up for a free account.
2. Install Git on Your Local Machine:
    - Download Git from git-scm.com.
    - Follow the installation instructions.
3. Create a New Repository on GitHub:
    - Select New Repository.
    - Name your repository.
    - Choose to make it Public or Private.
    - add a description and initialize with a README file.
    - Click Create Repository.
4. Add Your Source Code File to Git:
    - git add index.js.
5. Link Your Local Repository to GitHub:
    - Go to your GitHub repository page.
    - Copy the repository URL.
6. Verify Your Code on GitHub:
    - Go to your repository page on GitHub.
    - You should see your index.js file listed in the repository with the commit message you provided
7. Committing and Pushing Changes:
    - Whenever you make changes to your project, you can repeat the following steps to commit and push the changes.
    - Commit the Changes:
      git commit -m "Updated index.js with new feature".
    - Push the Changes to GitHub:
      git push.

Why is version control important in software development?

- ✓ Version control systems are a category of software tools that help in recording changes made to files by keeping track of modifications done in the code.

It allows multiple developers to collaborate efficiently, ensures that changes are well-documented, and helps maintain the integrity and history of a project.

Why version control is important in software development:

1. Collaboration Among Developers.
2. Tracking Changes and History.
3. Code Integrity and Backup.
4. Accountability and Traceability.
5. Branching and Feature Development.
6. Scalability for Large Projects.
7. Improved Communication and Documentation.

❖ Student Account in GitHub.

- ✓ GitHub offers a Student Developer Pack that provides students with free access to a variety of tools and resources that are essential for software development.

How to Get a GitHub Student Account:

1. Sign Up for GitHub.
2. Apply for the GitHub Student Developer Pack.
3. Verification Process.
4. Start Using the Pack.

- LAB EXERCISE: Create a student account on GitHub and collaborate on a small project with a classmate.

1) Create a Student Account on GitHub:

- Go to GitHub and Sign Up.
- Activate GitHub Student Benefits. 2) Create a Repository for Your Project:

- Create a New Repository.

 Invite your Classmates to Collaborate:

- Once the repository is created, go to the repository page.
- Click on the Settings tab at the top of the repository page.
- Scroll down to the Collaborators section on the left sidebar.
- type your classmate's GitHub username and click Add Collaborator.

3) Continue the Collaboration:

- Modify files locally.
- Stage, commit, and push changes to GitHub.
- Pull the latest changes from GitHub to stay up to date.
- Create pull requests to merge branches.

❖ What are the benefits of using GitHub for students?

✓ GitHub Education is a fantastic opportunity for students to build solid communities.

- Through the GitHub Student Developer Pack, you get a free GitHub Pro account as long as you remain a student.
- Free Domain Name and Hosting.
- Free Cloud Services.
- Learning Resources.
- Developer Tools and Services.

❖ Types of Software.

✓ System software.

1. application software.
2. development software.
3. middleware software.
4. network software.
5. programming software.

- LAB EXERCISE: Create a list of software you use regularly and classify them into the following categories: system, application, and utility software.

. 1. System Software:

System software is designed to manage and control the computer hardware so that application software can function.

Operating System (OS):

- Windows (Windows 10, 11)
- macOS
- Linux

Device Drivers:

- Graphics Driver • Printer Drivers

Virtualization Software:

- VMware Workstation (for running virtual machines)
- VirtualBox

2. Application Software:

Web Browsers:

- Google Chrome
- Mozilla Firefox • Safari

Productivity Software:

- Microsoft Office (Word, Excel, PowerPoint)
- Google Workspace (Docs, Sheets, Slides)
- Notion (Note-taking and project management)

Email Clients:

- Microsoft Outlook
- Mozilla Thunderbird

Media Players:

- VLC Media Player
- Spotify (for music streaming)
- Apple Music

3. Utility Software:

Antivirus Software:

- McAfee
- Norton Antivirus
- Windows Defender

File Compression Software:

- WinRAR • 7-Zip

Backup Software:

- Acronis True Image
- Macrium Reflect
- Time Machine

System Monitoring Tools:

- Task Manager (Windows)
- Activity Monitor (macOS)
- HWINFO (system information)

File Management Tools:

- Total Commander
- FileZilla (FTP client)

❖ What are the differences between open-source and proprietary software?

✓ Open-source software:

- Open-source software is computer software.
- whose source code is available openly on the internet and programmers can modify it to add new features and capabilities without any cost.
- In open-source software the source code is public.
- Open-source software can be installed on any computer.
- Users do not need to have any authenticated license to use this software.

Examples: Android, Linux, Firefox, Open Office, GIMP, VLC Media player, etc.

Proprietary software:

- Proprietary software is computer software.
- where the source codes are publicly not available only the company that has created them can modify them.
- In proprietary software, the source code is protected.
- Proprietary software cannot be installed into any computer without a valid license.

Examples: Windows, macOS, Internet Explorer, Google Earth, Microsoft Office, Adobe Flash Player, Skype, etc.

❖ GIT and GitHub.

✓ Git:

Git is a widely used version control system that was created by Linus Torvalds in 2005 and has been maintained by Junio Hamano since then.

It is used for:

- Tracking code changes.
- Tracking who made changes.
- Coding collaboration. GitHub:
- Git is not the same as GitHub.
- GitHub makes tools that use Git.
- GitHub is the largest host of source code in the world and has been owned by Microsoft since 2018.
- In this tutorial, we will focus on using Git with GitHub.

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

1. Create a Repository on GitHub:

- Log in to GitHub.
- Create a New Repository.

2. Create a New Branch:

Check the Current Branch:

- git branch.

Create a New Branch:

- git checkout -b feature-branch.

Verify the New Branch:

- git branch.
- The output will show a feature branch indicating that you're on the new branch.

3. Merge the Pull Request:

- On GitHub, go to your repository and click the Compare & Pull Request button that appears after pushing the branch.
- Provide a title and description for the pull request, then click Create pull request.
- This will allow you to propose merging the changes from the feature branch into the main.

- You can now create new branches for future features or fixes, make changes, commit, push, and merge as needed.

❖ How does GIT improve collaboration in a software development team?

✓ Git is used to track changes in source code, allowing multiple developers to collaborate on non-linear development.

1. Version control and tracking changes with Git. One of Git's core advantages is its ability to track changes to files and directories over time.
2. Branching and merging allow for collaborative workflow.
3. It enables remote collaboration.

❖ Application Software.

✓ . Application software refers to programs selected based on functionality and user accessibility.

Application software is a program or group of programs designed to help users perform specific tasks.

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

Types of Application Software:
Productivity Software.
Communication Software.
Project Management Software.
Database Management Software.
Cloud Storage Software.
How Application Software Improves Productivity:
Automation of Repetitive Tasks:

- Application software automates mundane, repetitive tasks, allowing individuals and organizations to focus on more critical, value-added activities.

Time Management:

- Many productivity applications have built-in tools that help users manage their time effectively.

Collaboration and Communication:

. This is especially important in remote and hybrid work environments. Data Organization and Accessibility:

- This improves decision-making by providing up-to-date information at the user's fingertips.

Cost and Resource Efficiency:

- This leads to overall cost savings and better allocation of resources.

❖ What is the role of application software in businesses?

✓ Business application software refers to specialized programs designed to optimize and support various operational processes within a business. Application software plays a key role in business by helping to improve efficiency, productivity, and communication.

By providing tools tailored to specific business needs, application software helps companies manage various aspects of their operations efficiently and effectively.

❖ Software Development Process.

✓ The software development process refers to the series of steps or stages that developers follow to design, develop, test, and deploy software.

1. Planning/Requirements Gathering
2. analysis
3. System Design
4. Implementation (Coding)
5. Testing
6. Maintenance

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

✓ . Flowchart for the Software Development Life Cycle:

1. Requirement Gathering:
   - Define system requirements.
   - Understand user needs and expectations.
   - Prepare the project scope.
2. System Design:
   - High-level system architecture.
   - Create detailed designs (e.g., database, interface).
   - Plan resources and technology stack.
3. Implementation (Coding):
   - Write the code based on design specifications.
   - Build the system components.
4. Testing:
   - Perform unit testing, integration testing, and system testing.
   - Identify bugs and issues, and fix them.
   - Ensure the software meets requirements.
5. Maintenance:
   - Ongoing bug fixes and updates.
   - Handle user feedback for improvements.
   - Ensure continued performance and security.

❖ What are the main stages of the software development process?

- There are main six stages of the software development process:

1. Planning/Requirements Gathering:
   - Understand and define the problem that the software aims to solve.
   - Define functional and non-functional requirements.
2. System Design:
   - Create the architecture and design that will fulfill the requirements.
   - High-Level Design: Outline the overall system architecture (for example: client-server, microservices).
   - Low-Level Design: Focus on individual components, databases, interfaces, and interactions.

- Select technologies, tools, and frameworks.

3. Implementation (Coding):
   - Translate design documents into working software through coding.
   - Developers write the code for the software, following the design documents.
   - Implement features, functions, and modules as specified in the requirements.
4. Testing:
   - Ensure the software works as intended and is free of bugs.
   - Unit Testing: Test individual components or functions.
   - Integration Testing: Ensure that different parts of the software work together.
   - System Testing: Test the entire software system as a whole to check for any errors.
5. Maintenance:
   - Keep the software functional and updated after deployment.
   - Bug Fixes: Correct any errors or issues reported by users.
   - Updates: Implement new features, improvements, or changes based on feedback or evolving requirements.

❖ Software Requirement.

- A requirement or ability needed by a user to address a challenge or meet a goal.

Software requirement is a document created by a system analyst after the requirements are collected from various stakeholders.

Types of Software Requirements:

1. Functional Requirements.
2. Non-Functional Requirements.

- LAB EXERCISE: Write a requirement specification for a simple library management system.

✓ The Library Management System (LMS) is designed to help libraries manage their collections of books, user memberships, and the borrowing process.

Requirement specification:

1. Ability to add and remove books from the library
2. Ability to search for books in the library by title, author, or ISBN
3. Ability to check out and return books
4. Ability to display a list of all books in the library
5. Ability to store and retrieve information about library patrons, including their name and ID number

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

✓ The requirement analysis phase is critical in software development because it sets the foundation for the entire project.

Requirement analysis and gathering are the critical first phases of the software development life cycle process.

The effectiveness of this phase often depends on the success or failure of a software project.

❖ Software Analysis.

✓ Software analysis is a requirement analysis that aims to determine the tasks needed to create fully functional software.

The purpose of software analysis is to ensure the software meets its intended requirements, operates efficiently, and is free from issues that could impact its performance, security, or user experience.

Types of Software Analysis:

6. Requirements Analysis
7. System Analysis
8. Static Analysis
9. Dynamic Analysis
10. Data Flow Analysis

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

✓ online shopping system:

Login and registration:

- Users can register to browse and purchase goods.

Search:

- Users can search for products using key and fuzzy words. Shopping cart:
- Users can add items to their shopping cart and choose to submit a purchase.

Message board:

- Users can submit orders and complete the purchase process.

❖ What is the role of software analysis in the development process?

✓ Software analysis plays a critical role in the development process, serving as the foundation for designing, building, and maintaining high-quality software systems.

It involves a thorough analysis of requirements, system capabilities, and design specifications.

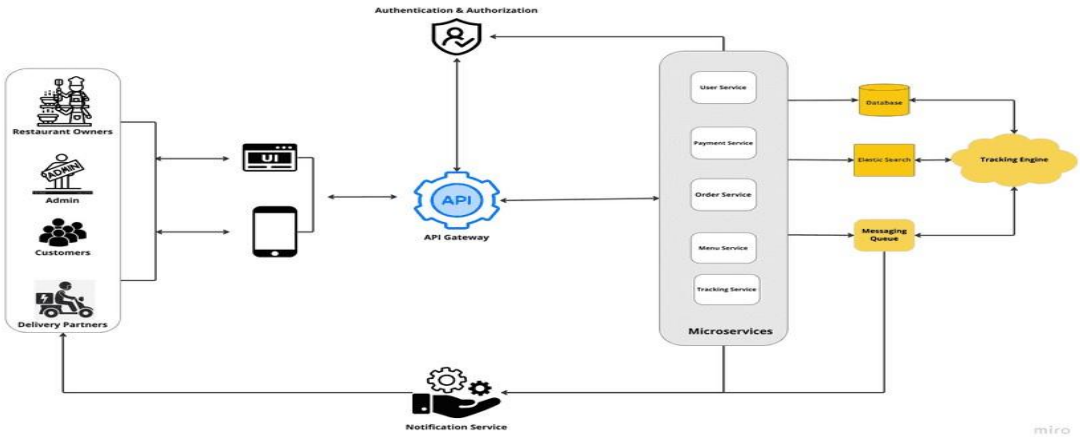
The Software Development Process includes Communication, requirement gathering, Feasibility Study, System Analysis, Software Design, Coding, Testing, Integration, Implementation, and Operation and Maintenance.

❖ System Design.

✓ System Design is a critical phase in the software development lifecycle where the architecture and components of a system are defined based on the requirements identified during the analysis phase. There are six types of system design:

1. Planning/Requirements Gathering
2. analysis
3. System Design
4. Implementation (Coding)
5. Testing
6. Maintenance

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



1. **User Interactions**: The mobile apps (Customer & Driver) send requests to the web server through the API Gateway.
2. **Order Management**: The Order Management API interacts with the databases to store or retrieve information related to the order.
3. **Payment Processing**: Once an order is confirmed, the Payment API connects to external payment services for transaction processing.
4. **Notifications**: The Notification Service sends real-time updates to users and drivers.
5. **Geolocation**: The system utilizes the Mapping API for calculating delivery routes and tracking the driver's location.

❖ What are the key elements of system design?

✓ System design is a structured process that involves creating the architecture and components of a system to meet specific requirements.

System design aims to build systems that are: reliable, effective, and maintainable.

Key concepts of system design and performance, such as:

- Latency
- Throughput
- Availability
- Redundancy
- Time
- CAP Theorem
- Lamport's Logical Clock Theorem.

❖ Software Testing.

✓ Software testing is the process of evaluating and verifying that a software application or system meets the specified requirements and functions as intended.

- Functional Testing: Selenium, QTP, Test Complete
- Performance Testing: Apache JMeter, LoadRunner
- Unit Testing: JUnit, N Unit, TestNG
- API Testing: Postman, SoapUI
- Security Testing: OWASP ZAP, Burp Suite.

- LAB EXERCISE: Develop test cases for a simple calculator program.

✓ Test Case 1: Addition Operation.

- Test Case Name: Test Addition of Two Positive Numbers • Test Description: Test the addition of two positive numbers.
- Test Input: 5 + 3
- Expected Output: 8
- Actual Output: (To be determined by execution)

Test Case 2: Multiplication Operation

- Test Case Name: Test Multiplication of Two Positive Numbers • Test Description: Test the multiplication of two positive numbers.
- Test Input: 4 * 3
- Expected Output: 12
- Actual Output: (To be determined by execution).

❖ Why is software testing important?

✓ Software testing is a crucial aspect of the software development process that ensures the reliability, functionality, and security of a software application.

Why the testing of the software is really important:

1. Helps in saving money
2. Security
3. Quality of the product
4. Customer Satisfaction

5. the development process
6. Easy while adding new features
7. Evaluating the software's performance.

❖ Maintenance.

✓ Software maintenance is the process of changing, modifying, and updating software to keep up with customer needs.

Maintenance Activities:

1. Monitoring and Issue Detection
2. Code Updates
3. Patch Management
4. Database Management 5. Documentation Updates.

- LAB EXERCISE: Document a real-world case where a software application requires critical maintenance.

Ans. Software maintenance is a continuous process that occurs throughout the entire life cycle of the software system.

Critical Maintenance of the Amazon Web Services:

1. Identifying and Rolling Back the Faulty Command:
    - AWS engineers worked to roll back the destructive command, restore lost metadata, and reconfigure the system to its normal state.
2. Restoring Service to Affected Customers:
    - The restoration process involved gradually bringing back functionality to affected customers.
    - AWS used its internal processes to ensure that metadata and data storage services were fully operational.
3. the S3 Metadata System:
    - The S3 metadata service was rebuilt from scratch in some parts, particularly the systems affected by the initial command error.
4. Improving Monitoring and Alerting:
    - much earlier in the future, providing engineers with quicker insights into system failures.
5. Communication and Transparency:

- hey kept users updated via their status page and social media channels throughout the process, which helped build trust despite the downtime.

❖ What types of software maintenance are there?

✓

1. **Corrective Maintenance**:
   Fixes defects or bugs after deployment.
   Identifying and resolving errors or issues that affect functionality.
2. **Adaptive Maintenance**:
   Modifying the software to remain compatible with new operating systems, hardware, or software platforms.
   Adjusting the system to integrate with updated external systems, services, or databases.
3. **Perfective Maintenance**:
   Adding new features or improving existing ones.
   Optimizing the software for better performance. Refining the user interface or user experience.
4. **Preventive Maintenance**:
   Proactively improves the software to avoid future issues and ensure longterm stability.
   Conducting routine health checks to identify and fix issues before they become major problems.

❖ Development.

✓ Software development refers to the entire process of designing, creating, testing, and maintaining software applications. There are six types of software development:

1. Planning/Requirements Gathering
2. analysis
3. System Design
4. Implementation (Coding)
5. Testing
6. Maintenance

❖ What are the key differences between web and desktop applications

✓ Web application:

- Runs inside a web browser.
- Hosted on web/application servers.
- Accessible from any internet-connected device.
- Uses a client-server model over the Internet.
- The programming languages used are JavaScript, PHP, Python, etc.
- Example applications include Gmail, Facebook, Twitter, etc.

Desktop application:

- It can only be accessed from the device it is installed on.
- Hosted/run directly from the user's device.
- Can only be accessed from the device it is installed on.
- Uses a standalone or client-server model within a local network.
- Programming languages used are C#, C++, Java, etc.
- Example applications include MS Word, Photoshop, Visual Studio, etc.

❖ Web Application.

✓ Web application is an application that runs on a web browser making use of a web server.

The web application is difficult to build as compared to the Windows application.

It can run on a variety of platforms including Mac, Linux, Solaris, Android, etc. Examples: Chrome, Internet Explorer, Firefox.

❖ What are the advantages of using web applications over desktop applications?

✓ Web applications have some advantages when compared to desktop applications, particularly in terms of accessibility, ease of maintenance, and deployment.

Advantages:

Cross-Platform Compatibility:
Works across all operating systems and devices with a browser.
No Installation Required:
Instant access via a web browser, no need for software installation.
Centralized Updates:
Automatic updates for all users, reducing maintenance efforts.
Accessibility:
Available from anywhere with an internet connection.
Scalability:
Easier to scale, particularly with cloud hosting.
Security:
Centralized security and encryption.
Lower System Requirements:
Less resource-intensive for users compared to desktop apps.

❖ Designing.

✓ Software Design is the process of transforming user requirements into a suitable form, which helps the programmer in software coding and implementation.

1. Interface Design:
   Interface design is the specification of the interaction between a system and its environment.
2. Architectural Design:
   The architectural design specifies the key components of a system, including their responsibilities, properties, interfaces, and the relationships and interactions among them.
3. Detailed Design:
   Detailed design is the specification of the internal elements of all major system components.

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

✓ UI (User Interface) and UX (User Experience) design play a crucial role in the success of any application, whether it's a web or mobile application.

UX designers specialize in conducting user research, creating wireframes, and developing user flows.
They design the user interface (UI), making sure it is intuitive and user-friendly.
Responsibilities of UI:

1. User Research
2. Information Architecture
3. Interaction Design
4. Wireframing and Prototyping
5. Collaborating.

Responsibilities of UX:

1. Typography
2. Colors
3. Styles
4. Branding 5. Spacing
6. Boldness
7. Number of items 8. Icons and Images.

❖ Mobile Application.

✓ A mobile app is a software application developed specifically for use on small, wireless computing devices, such as smartphones and tablets, rather than desktop or laptop computers.

There are 3 types of mobile apps based on technology:

1. native apps
2. web apps
3. hybrid apps.

Examples: Facebook, Instagram, WhatsApp, Amazon, etc.

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

✓ Native app:

- Native applications require installation.
- They require high maintenance.
- They have multiple codebases.

- They provide the best user experience.
- These applications are particularly developed for one platform.
- The languages used in native apps are Java, Swift, and Kotlin.

Hybrid app:

- These apps don't require installation.
- They require less maintenance.
- They have a single codebase.
- Hybrid apps don't have a good user experience.
- Hybrid apps can work on various platforms. It means that they can operate on both iOS and Android.
- The languages used in Hybrid apps are JavaScript, HTML, and CSS.

❖ DFD (Data Flow Diagram).

✓ A Data Flow Diagram (DFD) is a graphical representation of the flow of data through an information system.

The models enable software engineers, customers, and users to work together effectively during the analysis and specification of requirements.

It provides an overview of

- What data is the system processes?
- What transformations are performed?
- What data are stored?
- What results are produced, etc. There are three levels of DFD:

1. 0-level DFD:
   Represents the system as a whole and its interaction with external entities.
2. 1-level DFD:
   Breaks down the system into major sub-processes, each of which performs a specific function.
3. 2-level DFD:
   Break down Level 1 processes into more detailed processes.

Components of DFD:

- Process
- Data flow
- Datastore

- External Entity

- LAB EXERCISE: Create a DFD for a hospital management system.

✓

1. HOSPITAL SERVICES AUTOMATION BENEFITS.
2. FEATURES OF THE HOSPITAL MANAGEMENT SYSTEM.
3. PATIENT REGISTRATION AND ELECTRONIC HEALTH RECORDS (EHR).
4. APPOINTMENT AND RECEPTION MANAGEMENT.
5. LABORATORY AND TEST MANAGEMENT.
6. INVENTORY MANAGEMENT.

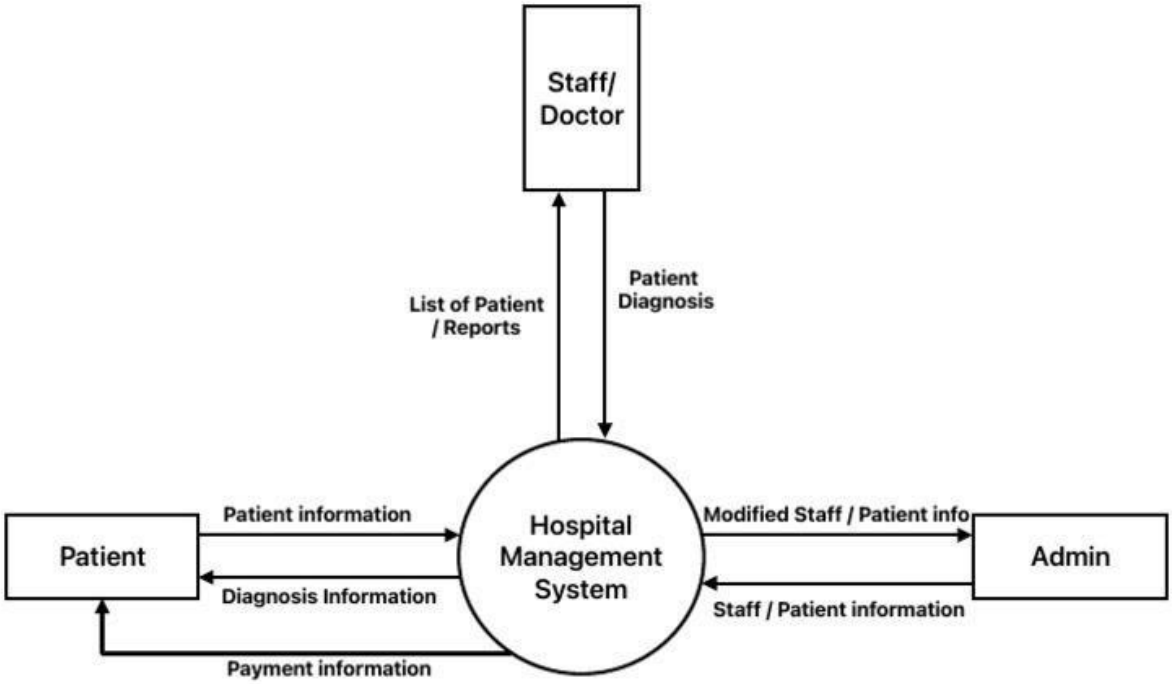


Fig. DFD level 0 of Hospital Management System

❖ What is the significance of DFDs in system analysis?

- ✓ Data Flow Diagrams (DFDs) are essential in system analysis as they visually depict the flow of data within an information system.

• Understand the system:

DFDs help to understand how a system operates and what its limitations are.

• Identify issues:

DFDs can help to identify potential problems or bottlenecks in the flow of data.

• Communicate:

DFDs can be used to communicate with both technical and non-technical staff.

• Document:

DFDs are often included in system documentation files.

• Plan:

DFDs can be used to plan the design of a new system and to determine what information is required for each process.

• Verify:

DFDs can be used to check that a completed system meets its intended design.

❖ Desktop Application.

Ans. A desktop application is a type of software that is designed to run on a personal computer or workstation.

- Uses a standalone or client-server model within a local network.
- Programming languages used are C#, C++, Java, etc.
- Example applications include MS Word, Photoshop, Visual Studio, etc.

How to develop a desktop application:

- Create a concept
- Design the application
- Select a programming language
- Select a development platform
- Install an IDE
- Create the application
- Test the application • Distribute the application.

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

✓ Let us create an application to get more familiar with PyQt5.

We will be developing a sample GUI application that has text written over it. Using a python:

```
import sys

# importing QtWidgets class modules
from PyQt5.QtWidgets import *

def window():
  app = QApplication(sys.argv)
wid = QWidget()    Lab =
QLabel(wid)

  # set the content to be displayed
  Lab.setText("Hello World! I can use PyQt!!")

  # set the geometry of application
wid.setGeometry(100, 200, 400, 100)

  # move application
  Lab.move(50, 20)

  # set the title
  wid.setWindowTitle("PyQt5")
  wid.show()

  # start the app
  sys.exit(app.exec_())

if __name__ == '__main__':
  window()
```

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

✓ Desktop application:

- Fully functional offline.
- Better performance, and resource-intensive tasks.
- Platform-specific (e.g., Windows, macOS).
- Manual installation and updates.
- Only accessible on the machine it's installed on.

Web application:

- Requires internet connection.
- May have slower performance, depending on internet and browser limitations.
- Works across platforms via browsers.
- Easy updates and no installation are required.
- Accessible anywhere with an internet connection.
- A lower, single version works across platforms.
- Centralized server-based storage.

❖ Flow Chart.

✓ A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

A flowchart is a diagram that shows the steps in a process. Flowcharts are often used for visualizing the sequence of actions or information needed for training, documenting, planning, and decision-making. They often use symbols, shapes, and arrows to illustrate how one step leads to another.

Key Components of a Flow Chart:

1. Oval (Start/End).
2. Rectangle (Process).
3. Diamond (Decision).
4. Parallelogram (Input/Output).
5. Arrow (Flow of Control).
6. Circle (Connector).

- LAB EXERCISE: Draw a flowchart representing the logic of a basic online registration system.

Ans. **Online Registration System:**

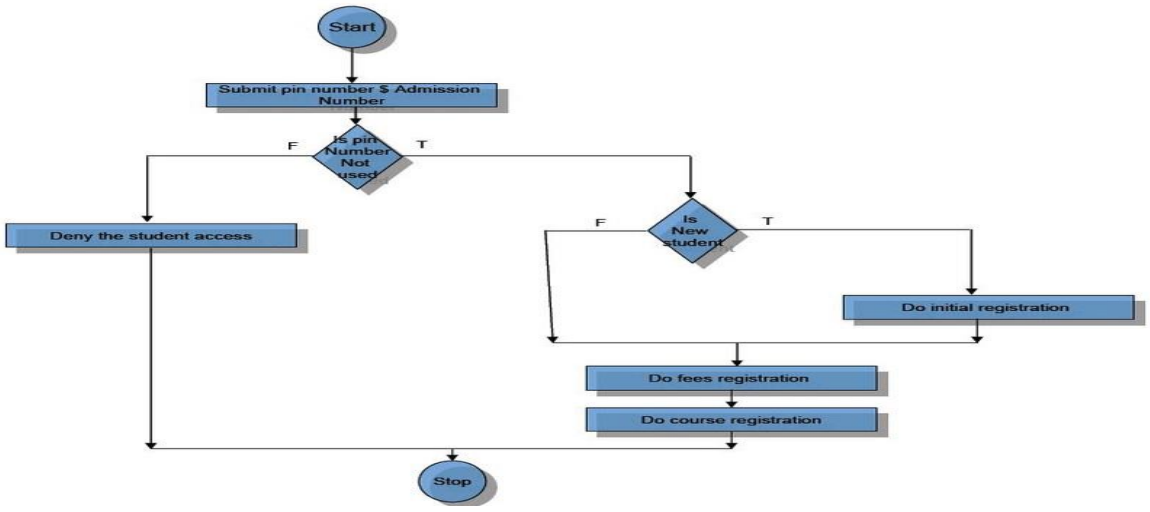
1. **Start**: The process starts when the user accesses the registration page.
2. **Enter Registration Information**: The user fills out the registration form with the required details (name, email, password).
3. **Validate Input**: The system checks if the entered details are valid.

- **Email Format Check**: Validates the email format (user@example.com).
- **Password Strength Check**: Ensures the password meets criteria length, special characters, numbers).
- **Field Completeness Check**: Verifies that all required fields are filled.

4. **Input Valid**: If the input is valid, proceed to the next step.

- If invalid, prompt the user to correct the mistakes and re-enter the information.

5. **Store User Data**: After validation, the system stores the user's registration details in a database.
6. **Registration Success**: If the data is successfully saved, display a confirmation message (Registration Successful).
7. **End**: The process ends here.



❖ How do flowcharts help in programming and system design?

- ✓ Flowcharts are a powerful tool in both programming and system design because they provide a clear, visual representation of the logic, processes, and decision-making flow within a program or system.

Flowcharts help with programming and system design:

- Identifying steps:
  Flowcharts help identify the essential steps in a process and organize them in chronological order.
- Communicating ideas:
  Flowcharts help communicate complex logic within a system and ideas quickly.
- Solving problems:
  Flowcharts help identify potential problem areas and solve problems.
- Debugging:
  Flowcharts help programmers debug code that is not working.
- Maintaining programs:
  Flowcharts simplify understanding the core structure of a program, making maintenance easier.
- Designing programs:
  Flowcharts can act as guides for creating the blueprint for designing a new program.