1.What is a Program?

→ A *program* is a piece of code or set of instructions that tells a how to perform a task.

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

→ PYTHON:

```
Print(“Hello World”)
```

→ C:

```
#include<stdio.h>
main()
{
printf("Hello World");
}
```

THEORY EXERCISE: Explain in your own words what a program is and how it functions.

→ The enter inputs and processes to give a output.

2. What is Programming?

→ Programming is the process of set of instructions that a system can follow to perform specific tasks.

THEORY EXERCISE: What are the key steps involved in the programming process?

→ Planning, Analysis, Designing, Implementation, Testing, Maintenance.

3. Types of Programming Languages?

THEORY EXERCISE: What are the main differences between high-level and low-level programming languages?

→ The high-level languages is easy to understand and low level language is tough to understand.

4. World Wide Web & How Internet Works

THEORY EXERCISE: Describe the roles of the client and server in web communication

→ **Client**:

Initiates the request.

Displays content to the user.

Can interact with the user directly.

→ **Server**:

Responds to the client's requests.

processes data.

Handles security, authorization and management.

5. Network Layers on Client and Server

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

→ Uses HTTP

6. Client and Servers

THEORY EXERCISE: Explain Client Server Communication

→ The process through which a client interacts with a server to request or send data.

→Client makes requests for data or services.

→Server processes these requests and sends back responses.

→Communication occurs over the internet using protocols like http/https.

7. Types of Internet Connections

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

Type	Pros	Cons
Broadband	Widely available, reliable, relatively affordable	Slower than fiber, may slow down during peak times
Fiber Optic	Very high speeds, low latency, future-proof	Expensive, limited availability
Satellite	Available in remote areas,	High latency, weather

Type	Pros	Cons
	no need for physical cables	sensitivity, slower speeds
Fixed Wireless	Faster than DSL, good for rural areas	Limited range, affected by weather

8. Protocols

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

→HTTP requests are simulated using curl to interact with web servers.

→FTP requests are simulated to interact with FTP servers using curl.

→These commands help test web and FTP communication directly from the command line.

THEORY EXERCISE: What are the differences between HTTP and HTTPS protocols?

→ Hypertext Transfer Protocol is a protocol using which hypertext is transferred over the Web.

→The full form of HTTPS is Hypertext Transfer Protocol Secure.

→The HTTP protocol does not provide the security of the data, while HTTPS ensures the security of the data.

9. Application Security

LAB EXERCISE: Identify and explain three common application security vulnerabilities.

→ SQL Injection: Use prepared statements, input validation, and least privilege.

→ XS: Escape output, implement Content Security Policy (CSP), sanitize input.

→ CSRF: Use anti-CSRF tokens, Same Site cookie attribute, double-submit cookies

Suggest possible solutions.

THEORY EXERCISE: What is the role of encryption in securing applications?

→Protects confidentiality by making data unreadable to unauthorized users.

→Ensures data integrity by detecting unauthorized modifications.

→Verifies authenticity by confirming the identities of the parties communicating.

→Prevents unauthorized access to sensitive information during storage or transmission.

→Supports regulatory compliance by meeting data protection standards.

10. Software Applications and Its Types

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

Operating System	Android, iOS
Web Browser	Google Chrome, Microsoft Edge
Media Player	VLC Media Player
Gaming	Candy Crush, Subway Surf

THEORY EXERCISE: What is the difference between system software and application software?

→ Application Software: Programs designed to help users complete specific tasks or activities on a computer or mobile device.

(Microsoft Word, Google Docs, Google Chrome, VLC Media Player)

→ System Software: The foundational software that manages the computer's

hardware and provides a platform for application software to run.

(Windows, macOS, backup software)

11. Software Architecture

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

→Presentation Laye: This is the user interface of the application, which could be a web browser or a mobile app.

→Business Logic Laye: The Business Logic Layer processes requests from the user interface and makes logical decisions.

→Database: The Data Layer is responsible for data storage and management.

THEORY EXERCISE: What is the significance of modularity in software architecture?

→Maintainability, Scalability, Reusability, Flexibility, Parallel Development, Testability, Reduced Complexity.

12. Layers in Software Architecture

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:

--User Registration and Login, Book Browsing, Checkout Process.

→2. Business Logic Layer

--User Authentication, Product Management, Order Processing.

→3. Database Layer

--User Data Management, Product Information, Order Records.

THEORY EXERCISE: Why are layers important in software architecture?

→Layers in software architecture are important because they provide a structured approach to organizing complex systems, ensuring that different responsibilities are handled separately.

13. Software Environments

LAB EXERCISE: Explore different types of software environments (development, testing,

production). Setup a basic environment in a virtual machine.

→1. Development Environment: DEs (Integrated Development Environments), code editors, version control systems (like Git), and local servers or databases. Tools such as Docker, Vagrant, or virtual environments may also be used.

→2. Testing Environment: Test automation frameworks, testing libraries, and test management tools.

→3. Production Environment: Cloud platforms (AWS, Azure), container orchestration, and monitoring tools. The production environment is highly optimized for performance, security, and availability.

THEORY EXERCISE: Explain the importance of a development environment in software production.

→Isolated Testing: It provides a space where developers can write and test code without affecting the live system. This isolation ensures that any bugs or issues are caught early in the development process.

→Consistency: It ensures that all developers work in a consistent setup, minimizing "it works on my machine" issues. With a standardized environment, the software behaves consistently, regardless of where it's developed.

→Tool Integration: The development environment can be configured with necessary tools such as Integrated Development Environments (IDEs), version control systems, debuggers, and libraries. This integration streamlines the development process.

→Dependency Management: It allows developers to manage dependencies efficiently, ensuring that the required software libraries and tools are available and configured correctly.

→Automated Builds and Testing: Many development environments support continuous integration (CI) and continuous deployment (CD) pipelines, which automate the building, testing, and deployment of code. This automation helps in maintaining high-quality software.

→Collaboration: A well-configured development environment supports

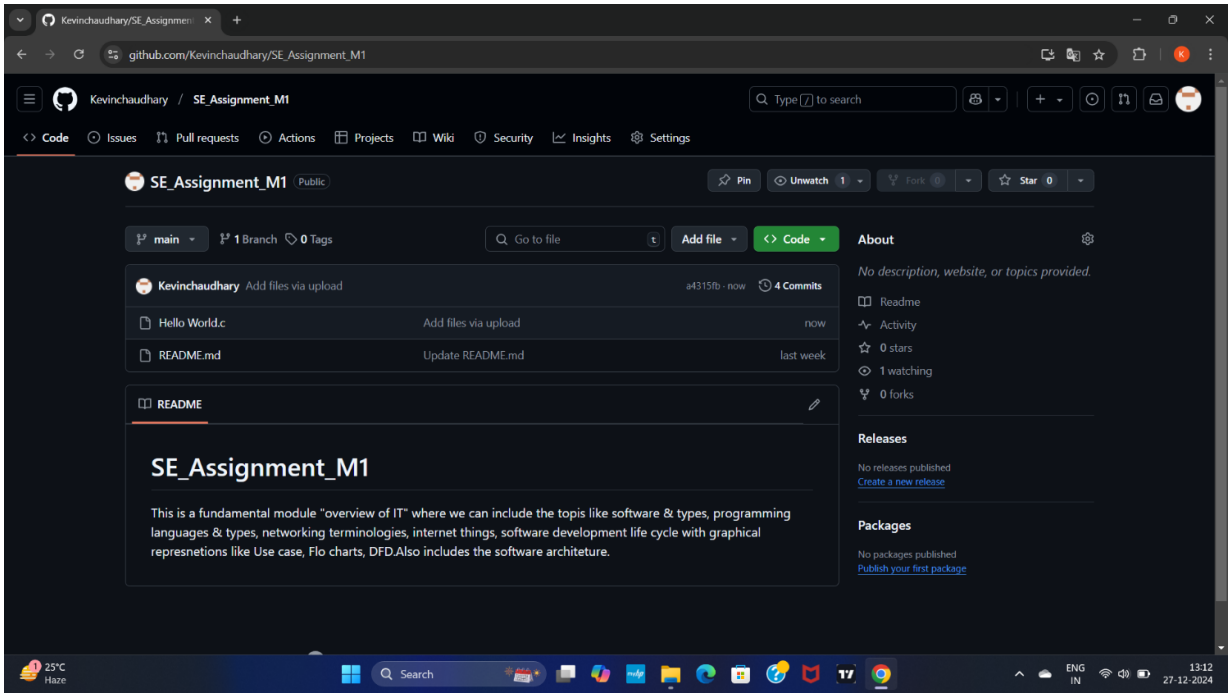
collaboration among team members. Version control systems like Git allow multiple developers to work on the same codebase simultaneously, track changes, and resolve conflicts.

→Experimentation and Learning: Developers can experiment with new technologies, frameworks, or solutions without the risk of affecting the production system. This flexibility fosters innovation and continuous learning.

14. Source Code

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

→



THEORY EXERCISE: What is the difference between source code and machine code?

→ Source code:

Source code is the human-readable set of instructions written by a programmer in a high-level programming language (e.g., Python)

→ Machine code:

Machine code is the low-level set of instructions that a computers cpu understands and can directly execute.

15. GitHub and Introductions

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

→ Create a GitHub repository on the GitHub website.

→ The repository to your local machine.

→ Them using git add, commit them with git commit, and push them to GitHub using git push.

THEORY EXERCISE: Why is version control important in software development?

→ Collaboration: Allows multiple developers to work on the same project simultaneously.

→ Tracking changes: Keeps a detailed history of code changes, making it easy to identify who made specific changes.

→ Backup and recovery: Acts as a backup, allowing developers to revert to previous versions if needed.

→ Branching and Merging: Facilitates creating separate branches for new features or fixes, which can be merged back into the main codebase.

→ Conflict Resolution: Helps manage and resolve conflicts when multiple changes affect the same part of the code.

→ Legal and Compliance: Provides an audit trail of changes for legal and compliance purposes.

16. Student Account in GitHub

THEORY EXERCISE: What are the benefits of using GitHub for students?

→Collaboration: Learn to work effectively with others on projects.

→Portfolio Building: Showcase projects in public repositories.

→Learning Resources: Access open-source projects and extensive documentation.

→Networking: Engage with the developer community and build connections.

→Skill Development: Improve coding skills through code reviews and exposure to best practices.

→Tools and Integrations: Enhance productivity with various integrations and automation features.

→Free Educational Resources: Utilize the GitHub Student Developer Pack for access to developer tools and software.

→GitHub equips students with essential skills, helps them create a professional portfolio, and connects them with a global community of developers.

17. Types of 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:

Windows, macOS, Linux, Android, iOS

Device Drivers (Graphics, Printer)

BIOS

2. Application Software:

Microsoft Office, Google Workspace

Google Chrome, Mozilla Firefox, Safari

Microsoft Outlook, Mozilla Thunderbird

VLC Media Player, Spotify

3. Utility Software:

Windows Defender, Norton Antivirus, McAfee

CCleaner, Disk Cleanup

Acronis True Image, Google Drive

THEORY EXERCISE: 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.

1. The software can be used for any purpose.

2. Allows to study how the software works.

3. Freedom to modify and improve the program.

4. No restrictions on redistribution.

→Proprietary software: Proprietary software is computer software where the source codes are publicly not available only the company that has created them can modify it.

1. Number of installations of this software into computers.

2. Restrictions on sharing of software illegally.

3. Time period up to which software will operate.

4. Number of features allowed to use.

18. GIT and GITHUB Training

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

→ Clone a Repository: Use git clone <repository-url> to copy a repository to your local machine.

→ Create a New Branch: Create and switch to a new branch using git checkout -b <new-branch-name>.

→ Make Changes and Commit: Make changes to files in your new branch.

→ Merge the Branch

THEORY EXERCISE: How does GIT improve collaboration in a software development team?

→Version Control: Tracks changes to code, enabling team members to work on the same project without conflicts.

→Branching and Merging: Allows multiple developers to work on features or fixes in isolated branches, which can be merged back into the main codebase.

→Centralized Repository: Facilitates easy sharing and updating of code among team members.

→Commit History: Keeps a detailed record of changes, helping in tracking progress and identifying issues.

→Conflict Resolution: Provides tools to handle merge conflicts when multiple changes are made to the same part of the code.

→Backup and Recovery: Ensures code is safely stored and can be recovered in case of accidental loss or errors.

19. Application Software

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

→Word Processing Software, Spreadsheet Software, Presentation Software, Database Management Software, Accounting Software.

→Efficiency: Automate repetitive tasks and processes, saving time and reducing errors.

→Collaboration: Enable multiple users to work together in real-time, improving teamwork and communication.

→Organization: Help in organizing data, tasks, and projects, leading to better management and tracking.

→Accessibility: Provide access to information and tools from anywhere, enhancing flexibility and remote work capabilities.

THEORY EXERCISE: What is the role of application software in businesses?

→1.Automation: Streamlines repetitive tasks, reduces manual effort, and minimizes errors, leading to increased productivity.

→2.Data Management: Facilitates the storage, retrieval, and analysis of large volumes of data, helping businesses make informed decisions.

→3.Communication: Improves internal and external communication through emails, instant messaging, and video conferencing tools.

→4.Collaboration: Supports teamwork with tools for project management, document sharing, and real-time collaboration.

→5.Financial Management: Simplifies accounting processes, invoicing, expense tracking, and financial reporting.

→6.Customer Relationship Management: Helps manage customer interactions, sales

processes, and customer service, enhancing customer satisfaction and loyalty.

→7.Supply Chain Management: Optimizes inventory management, procurement, and logistics, ensuring smooth operations.

20. Software Development Process

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

→

Planning→Analysis→Design→Implementation→Testing→Maintenance.

21. Software Requirement

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

→Ability to add and remove books from the library.

→Ability to search for books in the library by title, author, or ISBN.

→Ability to check out and return books

Ability to display a list of all books in the library.

→Ability to store and retrieve information about library patrons, including their name and ID number.

→Ability to track which books are currently checked out and when they are due to be returned.

→Ability to generate reports on library usage and checkouts.

THEORY EXERCISE: Why is the requirement analysis phase critical in software development?

→Requirement analysis lays the groundwork for a successful project by providing clarity, direction, and a strong foundation for all subsequent phases of development.

22. Software Analysis

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

→User Registration, Profile, Search and Filter, Add to Cart, Shipping Information, Payment Processing.

THEORY EXERCISE What is the role of software analysis in the development process?

→Software analysis ensures a clear understanding of requirements, guides the design and development phases, identifies risks, and helps in planning resources effectively. This ultimately leads to the successful delivery of a high-quality software product.

23. System Design

THEORY EXERCISE: What are the key elements of system design?

→Architecture, Database Design, APIs and communication, Caching, Load Balancing, Security, Logging, Usar Experience.

24. Software Testing

THEORY EXERCISE: Why is software testing important?

→1. Quality Assurance: Ensures the software meets specified requirements and is free of defects, resulting in a high-quality product.

2. Cost Efficiency: Identifies and resolves issues early, reducing the cost of fixing defects later in the development process.

3. User Satisfaction: Provides a smooth and reliable user experience, leading to increased user trust and satisfaction.

4. Compliance and Security: Verifies adherence to industry standards, regulations, and security protocols, protecting sensitive data.

5. Continuous Improvement: Facilitates feedback and iterative improvement, supporting agile development and rapid deployment of new features.

25. Maintenance

THEORY EXERCISE: What types of software maintenance are there?

→Corrective Maintenance: Fixes bugs and defects found in the software.

→Adaptive Maintenance: Adapts the software to changes in the environment or technology.

→Perfective Maintenance: Enhances the performance or functionality of the software based on user feedback.

→Preventive Maintenance: Prevents future issues by making proactive changes and improvements.

26. Development

THEORY EXERCISE: What are the key differences between web and desktop applications?

→Web Applications:

1. Accessibility: Can be accessed through web browsers from any device with an internet connection.
2. Installation: No installation required; users access the application via a URL.
3. Performance: Dependent on the internet connection and server performance.
4. Updates: Updates are deployed on the server and are immediately available to all users.

→Desktop Applications:

1. Accessibility: Installed on a specific device and can be used offline.
2. Installation: Requires installation on the user's device.

3. Performance: Typically faster and more responsive as they run directly on the device’s hardware002E
4. Updates: Users need to install updates manually or through an update manager.

27. Web Application

THEORY EXERCISE: What are the advantages of using web applications over desktop applications?

→ 1. Accessibility: Accessible from any device with an internet connection and a web browser, providing greater flexibility.

2. No Installation Required: Users can access the application via a URL without the need to install software on their devices.

3. Automatic Updates: Updates are deployed on the server, ensuring that all users have access to the latest version without needing to manually install updates.

4. Collaboration: Often designed with collaboration features, making it easier for multiple users to work together in real-time.

28. Designing

THEORY EXERCISE: What role does UI/UX design play in application development?

UI Design

Visual Appeal: Ensures the application looks attractive and engaging.

Consistency: Provides a uniform look and feel across the app, enhancing usability.

Usability: Makes the app easy to navigate and use, improving user satisfaction.

UX Design

User Research: Understands user needs and behaviors to inform design decisions.

User Journey: Maps out an intuitive and smooth user journey through the app.

Functionality: Aligns the app's functionality with user expectations.

Accessibility: Ensures the app is accessible to all users, including those with disabilities.

29. Mobile Application

THEORY EXERCISE: What are the differences between native and hybrid mobile apps?

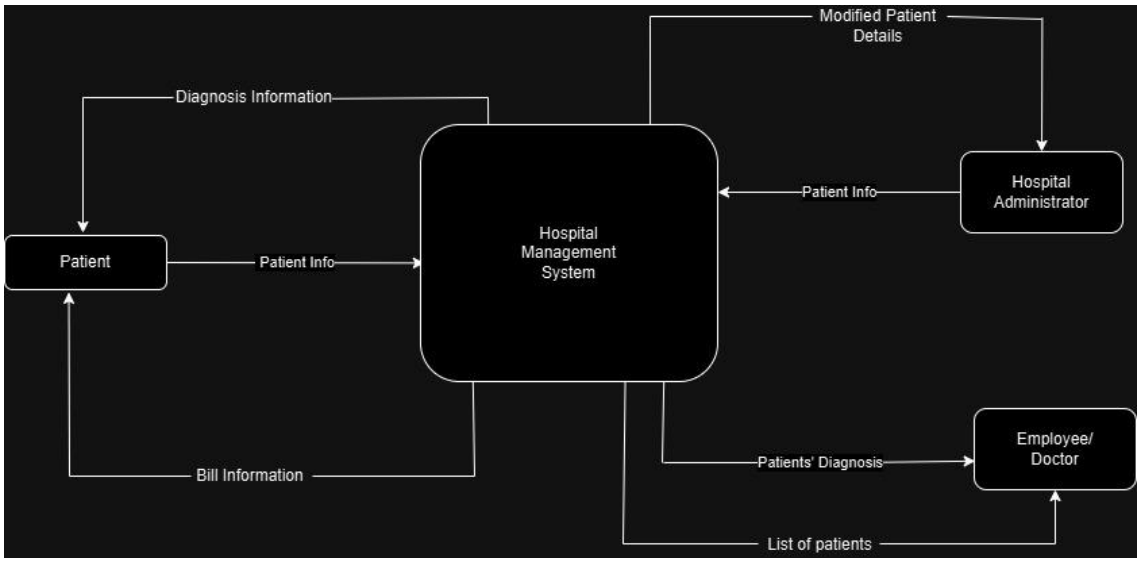
→

Feature	Native Apps	Hybrid Apps
Technology	Platform-specific languages (Swift, Java, Kotlin)	Web technologies (HTML, CSS, JavaScript)
Performance	Superior, optimized for the platform	Slower, dependent on web view
Development Time	Longer, separate codebases for each platform Faster, single	Faster, single codebase for all platforms

	codebase for all platforms	
User Experience	High, platform-specific UI/UX	Generally lower, may not adhere to platform conventions
Access to Features	Full access to device hardware and features	Limited access, depends on plugins
Maintenance	Requires managing multiple codebases	Easier to maintain with a single codebase
App Store Approval	Follows platform-specific guidelines	Same, but sometimes scrutinized more

30. DFD (Data Flow Diagram)

LAB EXERCISE: Create a DFD for a hospital management system



THEORY EXERCISE: What is the significance of DFDs in system analysis?

→ Data Flow Diagrams (DFDs) system analysis and design by offering a structured approach to understanding how data moves within a system. They help analysts and designers visualize the flow of information processes for improved efficiency.

31. Desktop Application

THEORY EXERCISE: What are the pros and cons of desktop applications compared to web applications?

→Pros:

Performance: Generally faster and more responsive as they run directly on the device's hardware.

Offline Access: Can be used without an internet connection.

Rich Functionality: Often provide more complex and powerful features due to direct access to the device's resources.

→Cons:

Installation Required: Must be installed on each device, which can be time-consuming and cumbersome.

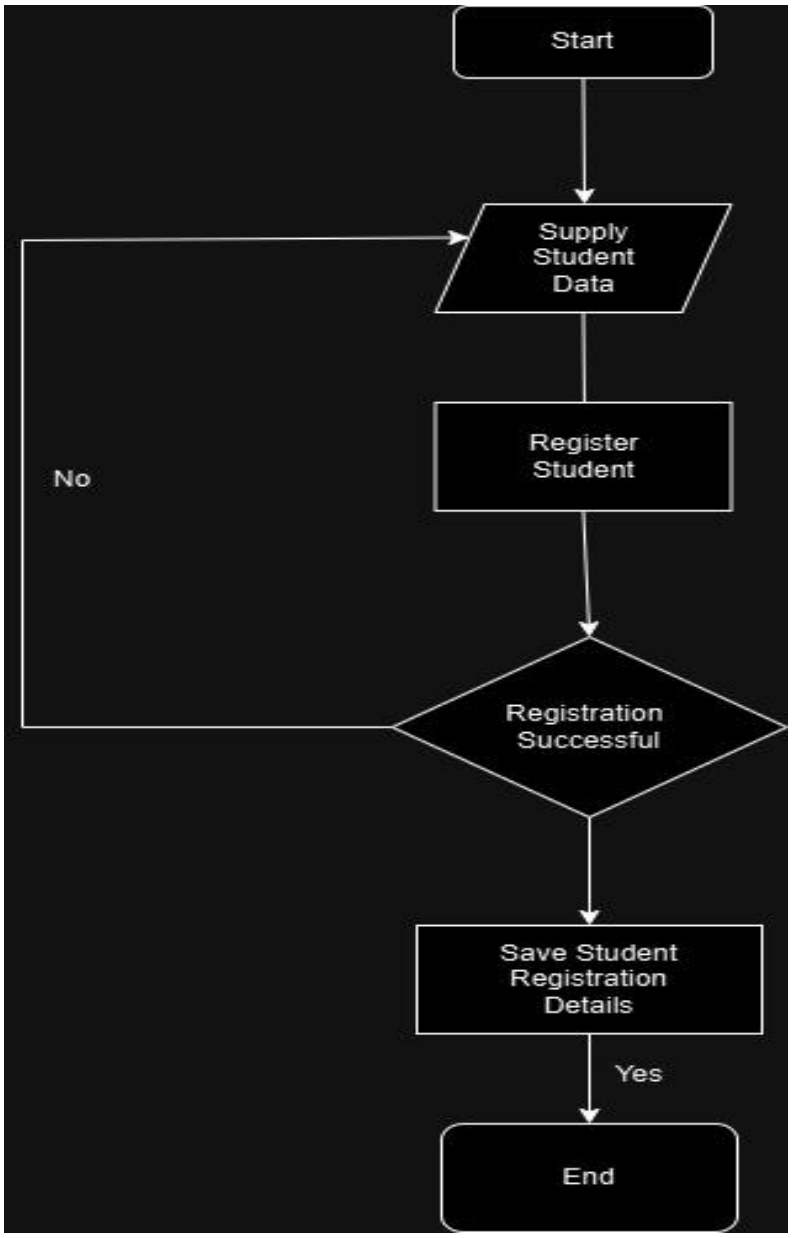
Updates: Users need to manually install updates unless the application includes an automatic update mechanism.

Platform Dependent: Typically designed for specific operating systems, limiting cross-platform compatibility.

Resource Intensive: Can consume significant device resources.

32. Flow Chart

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



THEORY EXERCISE: How do flowcharts help in programming and system design?

→Visual Clarity: Provides a clear, visual representation of the process, making it easier to understand and follow.

→Problem-Solving: Aids in identifying and troubleshooting errors or inefficiencies.

→Planning and Design: Acts as a blueprint for the development process, ensuring all necessary steps are included.