Module 1 – Overview of IT Industry

Q:-What is a Program?

Ans:-A program is a set of instructions that a computer can follow to perform a specific tasks.

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

C:-

```
#include <stdio.h>
int main() {
   printf("Hello, World!\n");
   return 0;
}
```

PYTHON:-

```
Print(“hello world”)
```

C:-

1.In c we need to import standard input output header(stdio.h) to print our hello world.

2.Main function to be included in c to make operations in that function.

3.printf(“”); function to display our hello world

4.return statement to be mentioned .

Python:

1.In python all modules for simple program will be imported priorly for print statement we need to just use print(“”) syntax.

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

A program is a set of instructions that computer follows to perform a specific tasks.

Functions:

1.It consists of a sequence of commands, algorithms, and data that tell the computer what to do step-by-step.

2. Programs are written in languages like Python, Java, C++, JavaScript, etc., which are then translated into machine-readable code (either directly by an interpreter or compiled into an executable file).

3. When a program is run (executed), the computer's central processing unit (CPU) follows the instructions to manipulate data, perform calculations, interact with hardware, and produce output.

4.Compiler is a program that translates high level program (human readable) into machine level code to execute code. Using step by step process like lexical analysis,syntax analysis,semantic analysis, intermediate code generation,code optimization,code generation.

Q:-What is Programming?

Ans:- Programming is the process of creating a set of instructions that a computer can understand and execute to perform a specific task or solve a particular problem.

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

Analyzing the Problem:- we need to analyze problem statement given, like what measures to solve problem , what type of functions to be used variables etc..,

Algorithm design / Pseudocode:- After analyzing the problem algorithm designing is most important aspect (algorithm is step by step process to solve a problem), we need to make a rough code(pseudo code) for better analysis of algorithm.

Flowchart:-Flowchart in programming gives clarity of how data flows in the loops and conditional statements.

Coding:- choosing any programming language in which we must code like python ,c ,c++ , java etc…,

Debugging:- It is the process of isolating and correcting the errors

Testing:- Execution of program in the intent of finding errors. Testing is done to improve quality, for verification and validation, for reliability estimation.

Final output:

Documentation:- A program analysis document with objectives, inputs, outputs, and processing procedures.

Types of Programming Languages:

1.Low-Level Languages: These languages are very close to the computer's hardware and are difficult for humans to read and write. Offer control over computer resources.

> Machine Language : Consists entirely of binary code (0s and 1s) that the CPU can execute directly.
>
> Assembly Language : Uses mnemonic codes (short, human-readable abbreviations like ADD, MOV, SUB) to represent machine instructions. It's a symbolic representation of machine language.

2. High-Level Languages : These languages are designed to be more human-readable and abstract away the complexities of the underlying hardware. Higher level of abstraction, focus on problem-solving rather than hardware details, more portable across different systems.

Examples: Python, Java, C++, JavaScript, Ruby, C#, PHP, Swift, Go, Kotlin, etc.

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

Feature	High-Level Language	Low-Level Language
Readability	More human-readable, using syntax similar to natural language	Difficult to read and understand, often using binary code (0s and 1s) or mnemonics .
Machine Dependency	Generally machine-independent (portable). Code can run on different systems with minimal or no changes (with an appropriate compiler/interpreter).	Machine-dependent (not portable). Code is specific to a particular CPU architecture and will not run on others.
Development Speed	Faster development time due to simplified syntax, extensive libraries, and built-in functionalities.	Slower development time as every detail needs to be explicitly managed.
Execution Speed	Generally slower because the code needs to be translated (compiled or interpreted) into machine code before execution.	Generally faster because it's closer to machine code and offers direct hardware access, leading to optimized performance.
Memory Management	Often handled automatically (e.g., garbage collection in Python, Java).	Requires manual memory management, which can be complex and error-prone.
Control over Hardware	Limited direct control over hardware and system resources.	Provides fine-grained, direct control over hardware, memory registers, and CPU.
Debugging	Easier to debug with better tools and clear error messages.	More challenging and time-consuming to debug due to the intricate nature of the code.
Maintenance	Easier to maintain and update due to readability and modularity.	More complex and demanding to maintain.
Examples	Python, Java, C++, JavaScript, Ruby, C#, PHP.	Machine language, Assembly language.

World Wide Web & How Internet Works.

The World Wide Web (WWW or simply "the Web") is a service built on top of that infrastructure. It's a system of interlinked hypertext documents and other web resources (images, videos, applications) that are accessed via the Internet using web browsers. It's like one of the many "applications" or "services" that run on the Internet, much like email, online gaming, or file sharing.

How the Web Works:

1. You (User) open a Web Browser: You type a URL (e.g., www.google.com) into your browser's address bar or click on a hyperlink.
2. DNS Lookup: Your browser needs to find the IP address (a numerical address like 172.217.160.142) of the web server hosting www.google.com. It does this by querying a

Domain Name System (DNS) server, which acts like the internet's phonebook, translating human-readable domain names into machine-readable IP addresses.

3. HTTP Request: Once your browser has the IP address, it sends an HTTP request over the Internet to that web server. This request asks for the specific web page or resource associated with the URL.
4. Server Processes Request: The web server receives the request, locates the requested web page file, and prepares to send it back.
5. HTTP Response: The web server sends an HTTP response back to your browser. This response includes the HTML content of the web page, along with any associated resources like images, CSS stylesheets (for styling), and JavaScript files (for interactivity).
6. Browser Renders Page: Your browser receives these files, interprets the HTML, CSS, and JavaScript, and then displays the web page on your screen, arranging the text, images, and other elements as intended.

The Internet is the infrastructure. It's the vast global network of interconnected computer networks, cables (fiber optic, copper), wireless connections, routers, and servers that allows computers to communicate with each other. It's the "roads" and "pipes" that carry information.

How the Internet Works

Data Creation

Packetization (TCP)

Addressing (IP)

Routing:

Transmission (Physical Layer)

Reception and Reassembly

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

Client-server architecture is a network model in which two main entities clients and servers communicate with each other to complete specific tasks or share data.

Client: The client initiates requests, waits for the server’s response, and displays it to the user.

Server: The server processes these requests, retrieves the relevant information, and sends it back to the client.

Types of Client-Server Architecture

1.Two-Tier Architecture: The simplest form where clients and servers communicate directly.

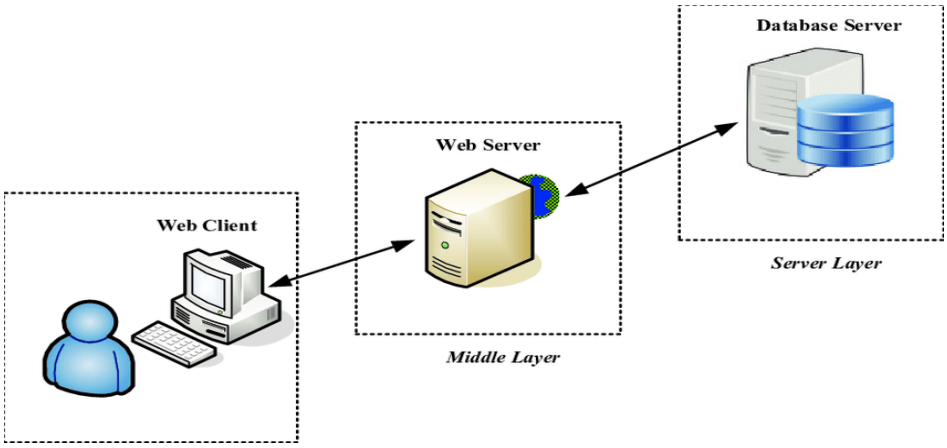
2.Three-Tier Architecture: A middleware layer, often called the application layer, is added between the client and server.

3.N-Tier Architecture: Involves multiple intermediary layers, such as security and business logic, to manage complex data processing and ensure security.

First the client sends request to server to display some data or some information.

Then server accepts the request and send information in the form of response http response.

The response will be carried in form of http response and send it to client.



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

Client-server architecture is a network model in which two main entities clients and servers communicate with each other to complete specific tasks or share data.

Client: The client initiates requests, waits for the server's response, and displays it to the user.

Server: The server processes these requests, retrieves the relevant information, and sends it back to the client.

Q:-Network Layers on Client and Server.

The Network Layer is primarily responsible for:Logical addressing (IP addresses),Routing packets between networks,Packet forwarding,Fragmentation and reassembly.

Assigns its own IP address as the source and the server's as the destination. Decides whether to send the packet directly or via a default gateway. Wraps the transport layer segment (e.g., TCP) in an IP packet. Splits packets if they're too large for the data link layer .

On server side it Checks whether incoming packets are addressed to its own IP address. Sends response packets back to the client, possibly through a router. Wraps outgoing data (like an HTTP response) in a new IP packet.

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

Server (server.py):

```
from http.server import BaseHTTPRequestHandler, HTTPServer

class SimpleHandler(BaseHTTPRequestHandler):
    def do_GET(self):
```

```
        self.send_response(200)
        self.send_header("Content-type", "text/plain")
        self.end_headers()
        self.wfile.write(b"Hello from the server!")

def run():
    server_address = ('', 8000)
    httpd = HTTPServer(server_address, SimpleHandler)
    print("Server running on port 8000...")
    httpd.serve_forever()

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

Client (client.py):

```
import requests


response = requests.get("http://localhost:8000")

print("Server response:", response.text)
```

THEORY EXERCISE: Explain the function of the TCP/IP model and its layers.

TCP/IP (TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL) is a protocol that enables communication and exchange of data between devices over internet.

TCP and IP are the two main protocols within this suite, with TCP ensuring reliable data transmission and IP handling the addressing and routing of data packets.

How TCP/IP Works:

Data Segmentation: When an application sends data, TCP divides it into smaller packets.

Packet Formatting: Each packet is given a header containing information like source and destination addresses (IP) and sequencing information (TCP).

Routing: IP uses these addresses to route the packets through the network.

Reassembly: At the destination, TCP reassembles the packets in the correct order based on the sequence numbers.

Error Handling: TCP also handles errors like lost or corrupted packets, requesting retransmissions as needed

Layers of TCP/IP Protocol:

1.Application Layer:

It supports different protocols like HTTP (for websites), FTP (for file transfers), SMTP (for emails), and DNS (for finding website addresses).and It supports data formatting

2.Transport Layer:

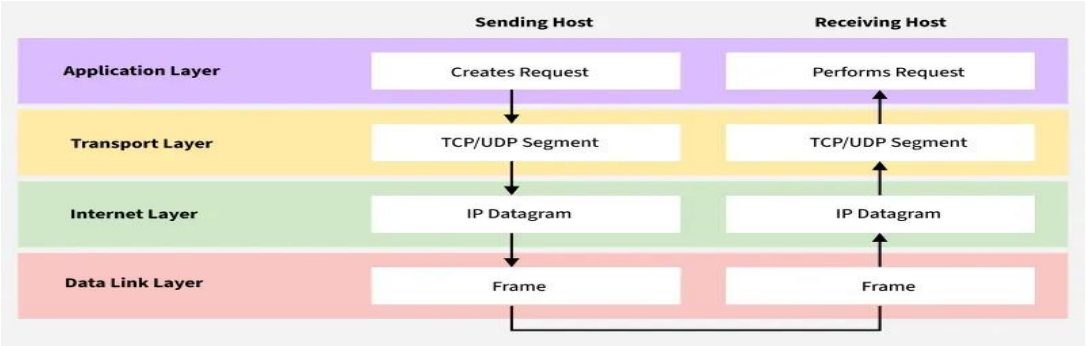
This layer establishes communication errors and control data it contains 2 main protocols TCP AND UDP. It checks that the data you send like a message, file, or video arrives safely and completely.

3.Internet Layer:

This layer control tasks, gives a unique ip address to devices.it also take care of packet forwarding and fragmentation.

4.Network Access Layer:

It also handles important tasks like using MAC addresses to identify devices, creating frames (the format used to send data over the physical link), and checking for basic errors during transmission.



Q:-Client and Servers

Ans:-Client:A program or device that makes a request (e.g., a browser requesting a web page).

Server: A program or machine that waits for requests and responds (e.g., a web server).

Protocol:Rules for communication, like HTTP, TCP, FTP, SMTP.

Q:- THEORY EXERCISE: Explain Client Server Communication

Ans:-Client-server communication is a model in computer networking where a client requests services and a server provides them. It is the foundation of how most of the internet works.

Client Sends an HTTP request, server Processes the request and sends back an HTTP response.

Client opens a connection to the server .Client sends a request .Server processes the request and prepares a response.Server sends the response back to the client.Connection may close or stay open .

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

There are several types of internet connections, each with pros and cons. Broadband (DSL or cable) is widely available and offers decent speed, but performance can drop during peak hours. Fiber-optic is the fastest and most reliable, ideal for heavy streaming or gaming, but it's more expensive and not yet available everywhere. Satellite works in remote areas without wired infrastructure, but it's slower, more expensive, and affected by weather. Mobile data (3G/4G/5G) offers flexibility and is good for on-the-go access, but data limits and speed can vary. Dial-up is very outdated and extremely slow, but still used in some rural areas with no alternatives.

Q:-THEORY EXERCISE: How does broadband differ from fiber-optic internet? Protocols

Broadband is a general term for high-speed internet that can be delivered through various technologies like DSL, cable, satellite, or fiber-optic. Fiber-optic internet is a specific type of broadband that uses thin strands of glass or plastic (fiber) to transmit data as light signals, offering much faster speeds and higher reliability than other broadband types. While broadband includes several transmission methods, fiber-optic stands out for its advanced technology and superior performance. Different internet protocols like PPPoE, IP, and TCP help manage data transfer, regardless of the broadband type used.

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

1. Simulate HTTP Requests with curl

- GET Request: Fetch the homepage of a website
- curl http://example.com
- POST Request: Send data to a server (e.g., login form)
- curl -X POST -d "username=user&password=pass" http://example.com/login
- Fetch headers only:
- curl -I http://example.com
- Download a file:
- curl -O http://example.com/file.zip

2. Simulate FTP Requests with curl

- List files in an FTP directory:
- curl ftp://ftp.example.com/ --user username:password

- Download a file via FTP:
- curl -u username:password ftp://ftp.example.com/file.txt -O
- Upload a file via FTP:
- curl -T localfile.txt -u username:password ftp://ftp.example.com/

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

Application Security

HTTP (Hypertext Transfer Protocol) and HTTPS (Hypertext Transfer Protocol Secure) are both used to transfer data between a web browser and a website. The main difference is that HTTPS adds a layer of security by using SSL/TLS encryption, which protects the data from being read or modified by attackers. This makes HTTPS much safer for sensitive activities like online banking or shopping. In terms of application security, HTTPS helps ensure data privacy, integrity, and secure user authentication, while HTTP does not provide these protections.

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

we explore three common application security vulnerabilities: SQL Injection, Cross-Site Scripting (XSS), and Broken Authentication.

SQL Injection happens when attackers insert malicious SQL code into input fields to access or change data in a database. To prevent this, developers should use parameterized queries and input validation.

Cross-Site Scripting (XSS) allows attackers to inject harmful scripts into web pages viewed by others. This can be prevented by sanitizing user input and escaping output in web applications.

Broken Authentication occurs when login systems are weak, allowing attackers to take over user accounts. Solutions include using strong password policies, multi-factor authentication (MFA), and session management best practices.

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

Encryption plays a key role in securing software applications by converting sensitive data into unreadable code, so only authorized users can access it. This helps protect information such as passwords, personal details, and financial data from hackers.

In all types of software applications—whether web, mobile, or desktop—encryption ensures data privacy and integrity, both during storage and transmission. It is a crucial part of application security, helping to prevent data breaches and unauthorized access.

Software Applications and Its Types

Software applications are programs designed to help users perform specific tasks on a computer or device. These applications can be categorized into different types based on their purpose:

1. System Software – Helps run the computer hardware and system, like operating systems (e.g., Windows, Linux).
2. Application Software – Performs specific user tasks, such as word processors (e.g., MS Word), spreadsheets, and web browsers.
3. Web Applications – Run in web browsers and require internet access, like Gmail or online banking apps.
4. Mobile Applications – Designed for smartphones and tablets, like WhatsApp or Instagram.

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

1. WhatsApp – *Application Software* (used for messaging and voice/video communication)
2. Instagram – *Application Software* (used for sharing photos, videos, and social networking)
3. Facebook – *Application Software* (used for social networking, messaging, and media sharing)
4. X (formerly Twitter) – *Application Software* (used for microblogging and news sharing)
5. File Manager – *System Software* (used to manage files and folders within the device's operating system)

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

System software and application software serve different roles in software architecture. System software is designed to manage and operate computer hardware, providing a platform for other software to run. Examples include operating systems like Windows or Android.

Application software, on the other hand, is created to help users perform specific tasks, such as browsing the internet, editing documents, or using social media apps like WhatsApp or Instagram. In simple terms, system software runs the computer, while application software lets users do useful things with it.

Software Architecture.

Software architecture is the fundamental structure of a software system, encompassing its components, their relationships, and the principles that guide its design. It's a blueprint that defines how a system is organized, how its parts interact, and how it meets its requirements for functionality, performance, and maintainability

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

[Presentation Tier]

(Web Browser / Mobile App)

[Application Tier]

(Web Server / API Server)

[Data Tier]

(Database Server)

Q:-THEORY EXERCISE: What is the significance of modularity in software architecture? Layers in Software Architecture

Modularity in software architecture means designing a system by breaking it into smaller, independent modules or components. This is important because it makes the software easier to understand, develop, test, and maintain. Each module can be worked on separately without affecting the entire system, which improves flexibility and helps in finding and fixing bugs faster.

In software architecture, layers are a way to organize these modules based on their roles, such as presentation, business logic, and data layers. This layered approach supports modularity by separating concerns, making the system more organized and scalable.

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

Case Study: Functionality of Presentation, Business Logic, and Data Access Layers

In a typical e-commerce software system, the presentation layer is responsible for interacting with users through a web interface or mobile app, displaying products, and capturing user inputs like orders and payments. The business logic layer processes these inputs by applying rules such as calculating totals, managing discounts, and handling payment validations. It acts as the core decision-maker of the system. The data access layer handles communication with the database, retrieving product details, storing order information, and updating inventory. This layered approach separates concerns, making the system easier to develop, maintain, and scale while ensuring smooth data flow from the user interface to the database.

Here's a simple case study example explaining the functionality of the three layers in a software system, such as an online shopping app:

Case Study: Students Marks Entry System

1. Presentation Layer:
This is the user interface used by teachers or administrators to enter and view students' marks. It includes forms where users input marks for different subjects, buttons to submit or update marks, and pages to display students' performance reports. The presentation layer ensures easy interaction and displays feedback like success or error messages.

2. Business Logic Layer:
This layer processes the data entered by users. It validates the marks to ensure they are within allowed ranges, calculates total scores and averages, determines grades based on predefined criteria, and applies any rules like pass/fail conditions. It also handles any updates or deletions of marks securely, ensuring data consistency.

3. Data Access Layer:
This layer manages communication with the database. It stores the students' marks, retrieves records when requested, and updates data as needed. It ensures data is saved accurately and efficiently, allowing quick access to information when teachers or administrators want to generate reports or analyze performance.

Q:-THEORY EXERCISE: Why are layers important in software architecture? Software Environments

Software architecture commonly utilizes a layered approach, organizing the application into distinct tiers or layers, each with specific responsibilities. These layers typically include the presentation layer, application layer, business layer, and data layer (or persistence layer). This structure promotes modularity, maintainability, and scalability by separating concerns and enabling independent development and modification of each layer

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

Software environments are categorized into development, testing, and production, each serving a distinct purpose in the software development lifecycle. A development environment is where coding and initial testing occur, a testing environment is used for more rigorous quality assurance, and a production environment is where the live application is accessible to end-users. Setting up a basic virtual machine (VM) environment allows for practical exploration of these concepts

Types of Software Environments:

Development Environment: This is where developers write, build, and test code. It includes tools like IDEs, compilers, and debuggers.

Testing Environment: This environment is designed to simulate the production environment for thorough testing of the software's functionality, performance, and security. Different types of testing environments can include performance testing, system integration testing (SIT), user acceptance testing (UAT), quality assurance (QA), security testing, chaos testing, alpha testing, and beta testing.

Production Environment: This is the live environment where the application is used by end-users.

Q:-THEORY EXERCISE: Explain the importance of a development environment in software production. Source Code.

The development environment is the first environment in software development that serves as a workspace for developers to perform programming and other processes around software and/or systems develop it defines a development environment as “a set of procedures and tools for developing, testing and debugging an application or a program.”

Q:-LAB EXERCISE: Write and upload your first source code file to Github.

In the upper-right corner of any page, select , then click New repository.

In the "Repository name" box, type a name for your project. For example, type "my-first-project."

In the "Description" box, type a short description. For example, type "This is my first project on GitHub."

Select whether your repository will be Public or Private. Select "Public" if you want others to be able to see your project.

Select Add a README file. You will edit this file in a later step.

Click Create repository.

Upload files to your project's repository

So far, you should only see one file listed in the repository, the README.md file you created when you initialized the repository. Now, we'll upload some of your own files.

1. To the right of the page, select the **Add file** dropdown menu.
2. From the dropdown menu, click **Upload files**.
3. On your computer, open the folder containing your work, then drag and drop all files and folders into the browser.
4. At the bottom of the page, under "Commit changes", select "Commit directly to the main branch, then click **Commit changes**.

Q:-THEORY EXERCISE: What is the difference between source code and machine code? Github and Introductions.

Source code is human-readable instructions written in a programming language, while machine code is the low-level binary code that a computer's processor directly executes. Essentially, source code needs to be translated into machine code before a computer can understand and run it.

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

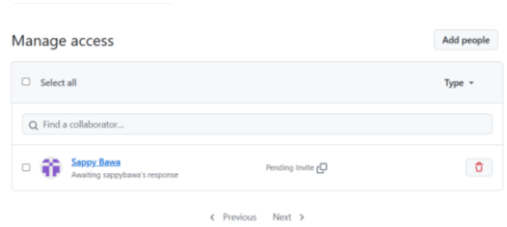
To create a GitHub repository and document how to commit and push code changes in a simple paragraph, follow these steps:

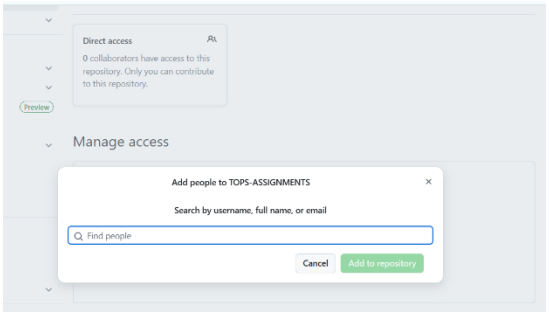
First, create a new repository on GitHub by logging in, clicking "New" repository, providing a name (e.g., my-project-workflow), checking "Initialize this repository with a README", and clicking "Create repository." Once created, navigate to the README.md file within your new repository on GitHub, click the pencil icon to edit it, and add a simple explanation. For instance, you can write: "After cloning this repository to your local machine (git clone <repository-url>), make your desired code changes. Then, use git add . to stage all modified or new files, followed by git commit -m 'Your concise message here' to save your changes locally with a descriptive note. Finally, execute git push origin main (or master if that's your branch name) to upload these committed changes from your local computer to this GitHub repository, making them available online." Save these changes to the README.md by committing them directly on GitHub.

Q:-THEORY EXERCISE: Why is version control important in software development? Student Account in Github

Version control is crucial in software development because it acts like a time machine and a collaboration manager for your code. It meticulously tracks every change made to your project, allowing you to revert to earlier versions if mistakes occur, compare different states of your code, and understand who made what changes and why. This robust history not only provides a vital safety net against errors but also enables multiple developers to work simultaneously on the same project without overwriting each other's work, seamlessly merging their contributions. For students, the GitHub Student Developer Pack is an incredible resource, offering free access to professional developer tools and services like premium IDEs, cloud credits, and learning platforms, which are invaluable for gaining real-world experience and building a strong portfolio.

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





Q:-THEORY EXERCISE: What are the benefits of using Github for students? Types of Software

GitHub offers immense benefits for students, primarily by providing a professional platform to manage code, collaborate on projects, and build a public portfolio. Through the GitHub Student Developer Pack, students gain free access to a treasure trove of industry-standard tools like premium IDEs, cloud hosting credits, and AI coding assistants (e.g., GitHub Copilot Pro), which would otherwise be expensive. This access helps them learn practical skills, experiment with various technologies, and gain real-world experience, making them more competitive in the job market.

System Software: This manages the computer hardware and provides a platform for other software to run, including operating systems (e.g., Windows, macOS, Linux), device drivers, and utility software (e.g., antivirus, disk defragmenters).

Application Software: Designed for end-users to perform specific tasks, this category is vast and includes productivity software (e.g., Microsoft Word, Google Sheets), multimedia software (e.g., VLC Media Player, Adobe Photoshop), web browsers, communication software (e.g., Zoom, Slack), and more specialized applications for various industries.

Programming Software: These are tools used by developers to write, test, and debug other software, such as code editors (e.g., VS Code), compilers, debuggers, and integrated development environments (IDEs).

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

Application Software:

WhatsApp

Instagram

Photos (as a photo viewing/editing app)

System Software:

File Manager (this is often part of or deeply integrated with the operating system)

Q:-THEORY EXERCISE: What are the differences between open-source and proprietary software? GIT and GITHUB Training.

Open-source software (OSS) is characterized by its publicly accessible source code, meaning anyone can view, modify, and distribute it. This fosters community collaboration, rapid bug fixes, and high customizability, often at no direct monetary cost for the software itself. In contrast, proprietary software (or closed-source) has restricted access to its source code, owned and controlled by a specific company or individual. Users typically purchase licenses to use it, relying on the vendor for updates, support, and bug fixes, with customization usually limited to what the vendor allows.

For GIT and GITHUB training, the process generally involves:

Installing Git: This is the version control system that runs locally on your computer.

Creating a GitHub account: GitHub is a web-based hosting service for Git repositories, providing a platform for collaboration and showcasing projects.

Basic Git Commands: Learning fundamental commands like git clone (to get a copy of a repository), git add (to stage changes), git commit (to save changes locally with a message), and git push (to upload local changes to GitHub).

GitHub Workflow: Understanding how to create repositories, invite collaborators, use branches for feature development, and create pull requests to merge changes, enabling efficient teamwork. Numerous free tutorials, documentation (like GitHub Docs), and interactive courses are available online to guide beginners through these steps.

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

To practice Git, I'll follow a common interactive tutorial workflow that demonstrates cloning, branching, and merging. First, I'll find a simple, public Git repository (or create a new one on GitHub for this purpose). Then, I'll use git clone <repository-url> in my local terminal to create a copy of that repository on my computer. Next, to practice branching, I'll create a new branch for a feature or fix using git branch <new-branch-name> and then switch to it

with git checkout <new-branch-name>. After making some changes to files within this new branch, I'll stage them with git add . and commit them with git commit -m "My descriptive message for changes on this branch". Finally, to practice merging, I'll switch back to the main branch (git checkout main) and then integrate the changes from my feature branch using git merge <new-branch-name>, resolving any potential merge conflicts if they arise, and confirming the successful merge.

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

Git drastically improves collaboration in software development teams by enabling multiple developers to work on the same project simultaneously without overwriting each other's work. Its core features like branching allow team members to create isolated environments for new features or bug fixes, preventing conflicts with the main codebase. When changes are ready, merging capabilities efficiently integrate these separate lines of development. Furthermore, Git provides a clear, detailed history of all changes, fostering accountability and making it easy to track and revert mistakes. Platforms built on Git, like GitHub, enhance this by facilitating code reviews through pull requests, which encourages feedback and higher code quality before changes are integrated.

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

Application software comprises programs designed for end-users to perform specific tasks, directly improving productivity across various domains. These diverse tools enhance efficiency by automating routine processes, streamlining workflows, and facilitating better organization and communication.

Key types of application software and their productivity benefits include:

Word Processing Software (e.g., Microsoft Word, Google Docs): Enables quick creation, editing, and formatting of text documents, automating tasks like spell-checking and grammar correction, thus speeding up document production.

Spreadsheet Software (e.g., Microsoft Excel, Google Sheets): Offers powerful tools for data organization, complex calculations, and analysis. It automates numerical tasks, generates charts for quick insights, and helps manage large datasets efficiently.

Presentation Software (e.g., Microsoft PowerPoint, Google Slides): Facilitates the creation of visually engaging presentations with multimedia integration, allowing users to convey information effectively and professionally without manual design.

Communication & Collaboration Software (e.g., WhatsApp, Slack, Zoom): Enables real-time messaging, video conferencing, and file sharing, breaking down communication barriers and fostering efficient teamwork, especially for remote or distributed teams.

Database Management Systems (DBMS) (e.g., MySQL, Oracle): Helps store, retrieve, and manage large volumes of structured data, ensuring data integrity, quick access, and efficient reporting, which is crucial for informed decision-making.

Project Management Software (e.g., Asana, Trello): Streamlines task assignment, deadline tracking, and resource allocation, improving project visibility, team coordination, and overall project completion rates.

Graphics & Multimedia Software (e.g., Adobe Photoshop, VLC Media Player): Provides tools for creating, editing, and manipulating images, videos, and audio, allowing professionals and amateurs to produce high-quality visual content efficiently.

In essence, application software empowers individuals and organizations to accomplish tasks faster, with greater accuracy, and often with less manual effort, leading to significant boosts in overall productivity.

Q:- THEORY EXERCISE: What is the role of application software in businesses? Software Development Process

Application software plays a pivotal role in businesses by automating tasks, streamlining workflows, and enhancing overall operational efficiency. These programs, which range from specialized tools like Customer Relationship Management (CRM) or Enterprise Resource Planning (ERP) systems to common office suites, enable businesses to manage data, communicate effectively, analyze performance, and make informed decisions, ultimately leading to increased productivity and a competitive edge.

The Software Development Process (SDP), or Software Development Life Cycle (SDLC), is a structured methodology for creating high-quality software, typically involving several key phases:

Planning & Requirements Analysis: Defining the software's purpose, scope, and gathering detailed requirements from stakeholders.

Design: Creating a blueprint for the software's architecture, user interface, and overall system structure.

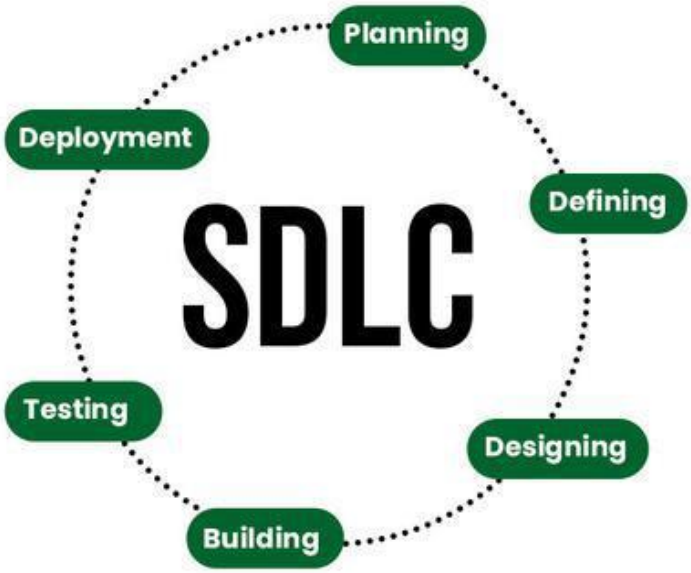
Development/Coding: Writing the actual code based on the design specifications.

Testing: Rigorously checking the software for bugs, errors, and ensuring it meets all requirements.

Deployment: Releasing the finished software to users or integrating it into the business environment.

Maintenance: Ongoing support, bug fixes, updates, and enhancements to keep the software functional and relevant.

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



Q:- THEORY EXERCISE: What are the main stages of the software development process? Software Requirement

The main stages of the software development process are: Planning & Requirements Analysis, Design, Development (Coding), Testing, Deployment, and Maintenance. These phases, often part of a Software Development Life Cycle (SDLC), ensure a structured and efficient approach to creating software.

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

A simple Library Management System will allow librarians to efficiently manage book inventory and member information. The system must enable librarians to add new books to the collection, including details like title, author, ISBN, and quantity. It must also support registering new library members, capturing their name, contact information, and a unique member ID. The core functionality will include borrowing books, where librarians can assign a book to a member, recording the borrow date. Conversely, the system must facilitate returning books, marking them as available again and noting the return date. Librarians should also be able to search for books by title or author, and search for members by name or ID. The system will track the availability status of each book (borrowed/available) and provide a basic overview of currently borrowed books.

Q:- THEORY EXERCISE: Why is the requirement analysis phase critical in software development? Software Analysis

The requirement analysis phase is absolutely critical in software development because it serves as the blueprint for the entire project. In this phase, Software Analysts (often Business Analysts) work closely with stakeholders to understand, define, and document precisely what the software needs to do, how it should perform, and what constraints it must operate within. Without a thorough understanding of these requirements, developers

might build the wrong features or miss critical functionalities, leading to costly reworks, delays, budget overruns, and ultimately, a product that doesn't meet user or business needs. It minimizes risks, ensures all parties have a shared understanding of the project's goals, and lays a solid foundation for efficient design, development, and testing.

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

A functional analysis for an online shopping system identifies all the specific actions the system *must* perform to meet user and business needs. For customers, this includes user registration and login, allowing them to create accounts, log in securely, and manage their profiles. Key functionalities also involve product Browse and search, enabling customers to find items through categories, keywords, and filters. A crucial part is the shopping cart management, where users can add, remove, or update product quantities before proceeding to checkout, which encompasses selecting shipping methods, entering delivery addresses, and completing secure payment processing. For administrators, the system must support product management (adding, editing, deleting products), order management (processing, tracking, and fulfilling orders), and potentially user management. Additionally, functions like product reviews and ratings, promotions and discounts, and customer support inquiry submission enhance the user experience and contribute to the system's overall value.

Q:- THEORY EXERCISE: What is the role of software analysis in the development process? System Design

Software analysis plays a crucial role in the development process by acting as the bridge between raw ideas and a concrete system design.[1] It involves deeply understanding the problem a software aims to solve, gathering detailed requirements from all stakeholders (users, clients, etc.), and identifying any potential issues or inconsistencies in those requirements.[2] This process helps Software Analysts define *what* the system needs to do, creating a clear and shared understanding for everyone involved.[3] Without thorough software analysis, the subsequent System Design phase, which focuses on *how* the system will be built (architecture, modules, database), would lack a solid foundation, leading to misinterpretations, costly reworks, and ultimately, a software product that fails to meet its intended purpose or business goals.

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

Frontend (Clients)

- Customer App/Web
    - Browse restaurants & menu
    - Place orders & make payments
    - Track delivery in real-time
- Restaurant App/Web

- Manage menu & inventory
- Accept/reject orders
- View order status

- Delivery Agent App
  - Receive delivery assignments
  - Navigate to restaurants and customer locations
  - Update delivery status

Q:- THEORY EXERCISE: What are the key elements of system design? Software Testing

Key Elements of System Design (Software Testing)

System design involves planning the architecture and components of a software system to ensure it meets functional and non-functional requirements. Key elements include scalability, which ensures the system can handle growth; modularity, which breaks the system into manageable, independent components; reliability, to keep the system working under various conditions; maintainability, which allows for easy updates and bug fixes; and performance, to ensure fast and efficient operations. In software testing, system design also considers how components interact and how data flows, helping testers identify integration points, dependencies, and potential failure areas.

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

Test Case ID	Operation	Input	Expected Output	Remarks
TC01	Addition	2 + 3	5	Basic addition
TC02	Subtraction	10 - 4	6	Basic subtraction
TC03	Multiplication	3 * 7	21	Basic multiplication
TC04	Division	20 / 4	5	Basic division

Test Case ID	Operation	Input	Expected Output	Remarks
TC05	Division by Zero	8 / 0	Error / Exception	Division by zero case
TC06	Negative Numbers	-5 + (-3)	-8	Addition with negatives
TC07	Decimal Addition	2.5 + 3.1	5.6	Floating-point addition
TC08	Zero Multiplication	0 * 9	0	Multiplying with zero
TC09	Large Numbers	100000 * 10000	1000000000	Stress test for large input
TC10	Mixed Operations	(2 + 3) * 4	20	Operator precedence test

Q:- THEORY EXERCISE: Why is software testing important? Maintenance

Software testing is crucial for maintenance because it helps ensure that updates, bug fixes, and new features do not break existing functionality. During maintenance, changes are made to improve performance, fix errors, or add enhancements. Without proper testing, these changes could introduce new bugs or negatively affect other parts of the system. Regular testing during maintenance—especially regression testing—helps maintain stability, reliability, and user trust, ensuring the software continues to function correctly over time.

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

Real-World Case: Critical Software Maintenance – Facebook Outage (2021)

In October 2021, Facebook, along with Instagram and WhatsApp, experienced a major global outage that lasted for several hours. The issue was caused by a faulty configuration change in the backbone routers that coordinate network traffic. This critical maintenance incident disrupted communication for billions of users and businesses worldwide. It highlighted the importance of careful planning, testing, and monitoring during software updates and infrastructure changes. The outage emphasized how even a small mistake in software maintenance can lead to massive real-world impacts.

Q:- THEORY EXERCISE: What types of software maintenance are there? Development

Types of Software Maintenance (Development)

In software development, there are four main types of software maintenance, each serving a specific purpose to keep the software reliable, efficient, and up to date:

1. Corrective Maintenance
   Fixes bugs or defects found after the software is released. Example: Resolving a crash issue in a mobile app.
2. Adaptive Maintenance
   Updates software to work in a new environment or with updated systems. Example: Modifying code to run on a new operating system.
3. Perfective Maintenance
   Enhances performance or adds new features based on user feedback. Example: Improving loading speed or adding a new search function.
4. Preventive Maintenance
   Makes the software more maintainable by restructuring code or removing outdated functions. Example: Refactoring code to prevent future issues.

These types of maintenance ensure that the software continues to meet user needs and technical requirements over time.

Q:- THEORY EXERCISE: What are the key differences between web and desktop applications? 27. Web Application

Key Differences Between Web and Desktop Applications

Web applications run in a web browser and require an internet connection, while desktop applications are installed directly on a computer and can often run offline. Web apps are typically platform-independent (accessible from any device with a browser), while desktop apps are usually designed for specific operating systems like Windows or macOS. Updating web apps is easier since changes are made on the server, whereas desktop apps often require users to download updates manually. Overall, web apps are more accessible and easier to maintain, while desktop apps can offer better performance and access to system resources.

Q:- T**HEORY EXERCISE: What are the advantages of using web applications over desktop applications? 28. Designing**

Advantages of Using Web Applications Over Desktop Applications (Designing)

In terms of designing, web applications offer several advantages over desktop applications. They provide a consistent user interface across different devices and platforms, making them easier to design responsively. Designers can use web standards like HTML, CSS, and JavaScript to create flexible, modern interfaces without worrying about operating system

compatibility. Web apps also allow for faster updates and design improvements, since changes are made on the server and instantly visible to users. This makes web applications more efficient to design, maintain, and improve over time.

Q:- THEORY EXERCISE: What role does UI/UX design play in application development? 29. Mobile Application

Role of UI/UX Design in Mobile Application Development

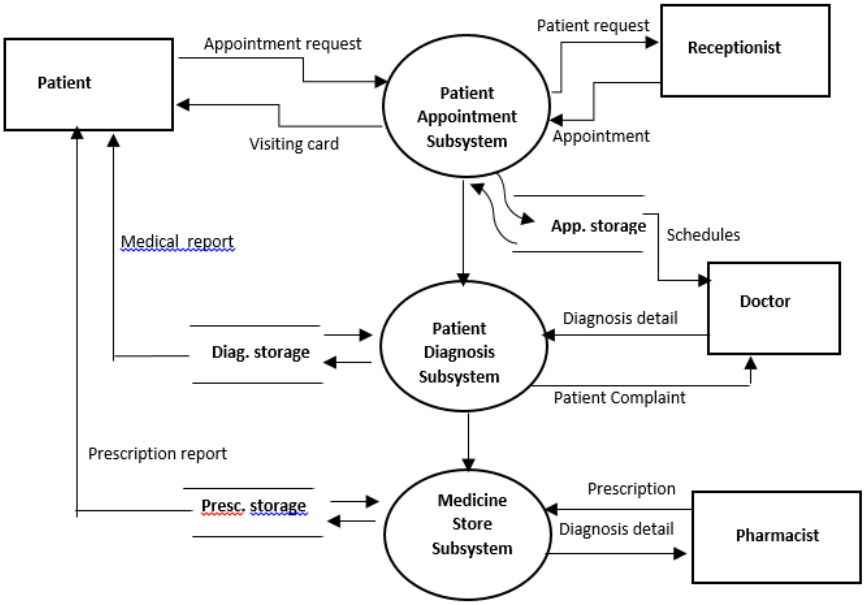
UI (User Interface) and UX (User Experience) design play a crucial role in mobile application development by ensuring the app is easy to use, visually appealing, and intuitive. A good UI makes the app look clean and organized, while UX focuses on how smoothly users can navigate and interact with the app. In mobile apps, where screen space is limited, thoughtful design improves usability, boosts user satisfaction, and encourages long-term engagement. Effective UI/UX design can also reduce user errors and increase app success by providing a seamless and enjoyable experience.

Q:- THEORY EXERCISE: What are the differences between native and hybrid mobile apps? 30. DFD (Data Flow Diagram)

Differences Between Native and Hybrid Mobile Apps (DFD Context)

Native mobile apps are built specifically for one platform (like iOS or Android) using platform-specific languages and tools, which allows them to fully utilize device features and deliver high performance. Hybrid apps, on the other hand, are developed using web technologies (like HTML, CSS, and JavaScript) and run inside a native container, allowing a single codebase to work across multiple platforms. When designing a Data Flow Diagram (DFD) for these apps, native apps often have more direct interactions with device hardware and system services, while hybrid apps have an extra layer between the user interface and the device, impacting how data flows through the system.

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



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

31. Desktop Application

Significance of DFDs in System Analysis (Desktop Application)

Data Flow Diagrams (DFDs) are significant in system analysis because they provide a clear, visual representation of how data moves through a system, including inputs, processes, and outputs. For desktop applications, DFDs help analysts and developers understand the flow of information between different components and user interactions, making it easier to identify inefficiencies or errors early. This clarity supports better design decisions, improves communication among stakeholders, and ensures the system meets user requirements effectively.

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

```
import tkinter as tk


def click_button(value):
    # Update display when buttons are clicked
    pass


def calculate():
    # Evaluate the expression and display result
    pass
```

```
def clear():
    # Clear the display
    pass

# Set up main window
root = tk.Tk()
root.title("Simple Calculator")

# Create display widget

# Create buttons for digits and operations

# Arrange buttons using grid or pack

root.mainloop()
```

Q:- THEORY EXERCISE: What are the pros and cons of desktop applications compared to web applications? 32. Flow Chart

Pros and Cons of Desktop Applications Compared to Web Applications (Flow Chart Context)

Desktop applications generally offer better performance, offline access, and deeper integration with system hardware compared to web applications. However, they require installation and updates on each device, which can be less convenient. Web applications are platform-independent and easier to update since changes are made on the server side, but they depend on internet connectivity and may have slower performance. When designing flow charts to represent processes in these applications, desktop apps might have more complex interactions with local resources, while web apps focus more on server-client data flow and user sessions.

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

```
graph LR
    A[Start] --> B{Open Registration Page};
    B --> C{Enter User Information};
    C --> D{Validate Information};
    D -- Valid --> E{Submit Registration};
    E --> F{Registration Successful};
```

F --> G[End];

D -- Invalid --> C;

B -- Already Registered --> H{Go to Login Page};

H --> I[End];

E -- Error --> J{Display Error Message};

J --> C;

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

Flowcharts are incredibly valuable visual tools in programming and system design because they provide a clear, step-by-step representation of processes, algorithms, or system workflows. They help in visualizing complex logic, breaking down intricate problems into manageable sequences of operations and decisions. This visual clarity aids programmers in planning their code structure before writing a single line, minimizing errors and fostering a modular approach. In system design, flowcharts illustrate how data flows between different components, revealing potential bottlenecks or inefficiencies. They also serve as excellent documentation tools, making it easier for team members to understand existing systems, onboard new developers, and facilitate effective communication and collaboration by providing a shared visual language for discussing processes.