TASK -1(AI)

from transformers import pipeline

def main():

print("="\*50)

print("TEXT SUMMARIZATION TOOL")

print("="\*50)

# Input long text

input\_text = input("\nEnter the article or paragraph to summarize:\n\n")

# Load summarization pipeline with BART model

summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

# Generate summary

print("\nSummarizing...\n")

summary = summarizer(input\_text, max\_length=130, min\_length=30, do\_sample=False)

# Output the result

print("="\*50)

print("SUMMARY:")

print("="\*50)

print(summary[0]['summary\_text'])

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

main()

INTERNSHIP TASK -2 SPEECH RECOGNITION SYSTEM I NSTRUCTIONS: BUILD A BASIC SPEECH-TO-TEXT SYSTEM USING PRE-TRAINED MODELS AND LIBRARIES LIKE SPEECHRECOGNITION OR WAV2VEC. DELIVERABLE: A FUNCTIONAL SYSTEM CAPABLE OF TRANSCRIBING SHORT AUDIO CLIP

import speech\_recognition as sr

# Initialize recognizer

recognizer = sr.Recognizer()

# Load an audio file (e.g., WAV format)

with sr.AudioFile('sample.wav') as source:

print("Listening...")

audio\_data = recognizer.record(source)

print("Transcribing...")

try:

# Using Google Web Speech API

text = recognizer.recognize\_google(audio\_data)

print("Transcription:")

print(text)

except sr.UnknownValueError:

print("Could not understand audio.")

except sr.RequestError as e:

print(f"API error: {e}")

INTERNSHIP TASK -3 NEURAL STYLE TRANSFER I NSTRUCTIONS: IMPLEMENT A NEURAL STYLE TRANSFER MODEL TO APPLY ARTISTIC STYLES TO PHOTOGRAPHS DELIVERABLE: A PYTHON SCRIPT OR NOTEBOOK WITH EXAMPLES OF STYLED IMAGES.

import torch

import torchvision.transforms as transforms

from torchvision.models import vgg19

from PIL import Image

import matplotlib.pyplot as plt

from torchvision.utils import save\_image

# Image preprocessing

def load\_image(img\_path, max\_size=400):

image = Image.open(img\_path).convert('RGB')

size = max\_size if max(image.size) > max\_size else max(image.size)

transform = transforms.Compose([

transforms.Resize(size),

transforms.ToTensor()

])

image = transform(image).unsqueeze(0)

return image

# Load content and style images

content = load\_image("content.jpg").to(torch.float)

style = load\_image("style.jpg").to(torch.float)

# Load pre-trained VGG19

vgg = vgg19(pretrained=True).features.eval()

# Initialize generated image

generated = content.clone().requires\_grad\_(True)

# Loss and optimizer

optimizer = torch.optim.Adam([generated], lr=0.004)

mse = torch.nn.MSELoss()

# Extract features

def get\_features(image):

layers = {'0': 'conv1\_1', '5': 'conv2\_1', '10': 'conv3\_1',

'19': 'conv4\_1', '28': 'conv5\_1'}

features = {}

x = image

for name, layer in vgg.\_modules.items():

x = layer(x)

if name in layers:

features[layers[name]] = x

return features

# Style Transfer Loop

for step in range(201):

gen\_features = get\_features(generated)

content\_features = get\_features(content)

style\_features = get\_features(style)

style\_loss = content\_loss = 0

for layer in gen\_features:

# Content loss

content\_loss += mse(gen\_features[layer], content\_features[layer])

# Style loss

G = gen\_features[layer].view(gen\_features[layer].shape[1], -1)

A = style\_features[layer].view(style\_features[layer].shape[1], -1)

style\_loss += mse(G @ G.T, A @ A.T)

total\_loss = content\_loss + 1e3 \* style\_loss

optimizer.zero\_grad()

total\_loss.backward()

optimizer.step()

if step % 50 == 0:

print(f"Step [{step}/200], Loss: {total\_loss.item():.4f}")

# Save and show result

save\_image(generated, "stylized.jpg")

plt.imshow(generated.squeeze().permute(1, 2, 0).detach())

plt.title("Stylized Output")

plt.axis('off')

plt.show()

INTERNSHIP TASK -4 GENERATIVE TEXT MODEL I NSTRUCTIONS: CREATE A TEXT GENERATION MODEL USING GPT OR LSTM TO GENERATE COHERENT PARAGRAPHS ON SPECIFIC TOPICS. DELIVERABLE: A NOTEBOOK DEMONSTRATING GENERATED TEXT BASED ON USER PROMPTS.

from transformers import GPT2LMHeadModel, GPT2Tokenizer

# Load model and tokenizer

tokenizer = GPT2Tokenizer.from\_pretrained("gpt2")

model = GPT2LMHeadModel.from\_pretrained("gpt2")

# Input prompt

prompt = "The future of artificial intelligence in healthcare is"

inputs = tokenizer.encode(prompt, return\_tensors="pt")

# Generate text

outputs = model.generate(inputs, max\_length=100, num\_return\_sequences=1, temperature=0.7)

# Decode and print

print("Generated Text:\n")

print(tokenizer.decode(outputs[0], skip\_special\_tokens=True))

(ANDROID)INTERNSHIP TASK -1 EXPENSE TRACKER APP I NSTRUCTIONS: DEVELOP AN APP TO TRACK EXPENSES, CATEGORIZE THEM, AND DISPLAY A SUMMARY USING CHARTS. DELIVERABLE: AN APP WITH DATA PERSISTENCE USING ROOM OR SQLITE.

MainActivity.kt

│

├── data/

│ └── Expense.kt (Entity)

│ └── ExpenseDao.kt

│ └── ExpenseDatabase.kt

│

├── ui/

│ └── AddExpenseActivity.kt

│ └── SummaryFragment.kt

│

└── utils/

└── ChartHelper.kt

INTERNSHIP TASK -2 MEDIA PLAYER APP I NSTRUCTIONS: BUILD A MEDIA PLAYER APP CAPABLE OF PLAYING AUDIO FILES FROM LOCAL STORAGE. DELIVERABLE: A FUNCTIONAL MEDIA PLAYER WITH PLAY, PAUSE, AND SKIP CONTROLS

val mediaPlayer = MediaPlayer()

mediaPlayer.setDataSource(audioUri)

mediaPlayer.prepare()

mediaPlayer.start()

INTERNSHIP TASK -3 RIDE-SHARING APP PROTOTYPE I NSTRUCTIONS: BUILD A PROTOTYPE FOR A RIDE SHARING APP WITH FEATURES LIKE LOCATION TRACKING, RIDE REQUESTS, AND DRIVER DETAILS DELIVERABLE: AN APP UTILIZING GOOGLE MAPS SDK AND FIREBASE BACKEND.

INTERNSHIP TASK -4 FITNESS TRACKER APP I NSTRUCTIONS: DESIGN AN APP TO TRACK STEPS, DISTANCE, AND CALORIES BURNED USING ANDROID’S BUILT-IN SENSORS. DELIVERABLE: AN APP WITH DATA VISUALIZATION AND REAL-TIME UPDATES.

val sensorManager = getSystemService(SENSOR\_SERVICE) as SensorManager

val stepSensor = sensorManager.getDefaultSensor(Sensor.TYPE\_STEP\_COUNTER)

sensorManager.registerListener(this, stepSensor, SensorManager.SENSOR\_DELAY\_UI)

(AUTOMATION) INTERNSHIP

TASK -1

DATA-DRIVEN

TESTING

I NSTRUCTIONS:

IMPLEMENT DATA-DRIVEN TESTING

USING SELENIUM AND READ INPUTS

FROM A CSV FILE.

DELIVERABLE: A SCRIPT THAT

EXECUTES TESTS WITH MULTIPLE

INPUT SETS AND LOGS RESULTS.

import pandas as pd

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.chrome.service import Service

from time import sleep

# Read CSV data

data = pd.read\_csv('test\_data.csv')

# Set up WebDriver

driver = webdriver.Chrome() # Or use Service + ChromeOptions if needed

driver.maximize\_window()

# Open log file

log\_file = open("test\_log.txt", "w")

# Test loop

for index, row in data.iterrows():

username = row['username']

password = row['password']

try:

driver.get("https://example.com/login") # Replace with actual URL

# Interact with input fields

driver.find\_element(By.ID, "email").send\_keys(username)

driver.find\_element(By.ID, "password").send\_keys(password)

driver.find\_element(By.ID, "login-button").click()

sleep(2) # Wait for response

# Check for success/failure (modify selector/logic as needed)

if "dashboard" in driver.current\_url:

result = "SUCCESS"

else:

result = "FAILURE"

log\_file.write(f"{username},{password} → {result}\n")

print(f"Tested {username}: {result}")

except Exception as e:

log\_file.write(f"{username},{password} → ERROR: {str(e)}\n")

print(f"Error testing {username}: {e}")

# Cleanup

log\_file.close()

driver.quit()

INTERNSHIP TASK -3 CI/CD INTEGRATION I NSTRUCTIONS: INTEGRATE AUTOMATED TESTS INTO A CI/CD PIPELINE USING JENKINS OR GITHUB ACTIONS. DELIVERABLE: A PIPELINE SETUP WITH AUTOMATED TEST EXECUTION AND REPORT GENERATION

name: Run Selenium Tests

on:

push:

branches: [main]

pull\_request:

branches: [main]

jobs:

test:

runs-on: ubuntu-latest

steps:

- name: Checkout Code

uses: actions/checkout@v3

- name: Set up Python

uses: actions/setup-python@v4

with:

python-version: '3.10'

- name: Install Dependencies

run: pip install -r requirements.txt

- name: Install Chrome and Chromedriver

run: |

sudo apt-get update

sudo apt-get install -y unzip xvfb libxi6 libgconf-2-4

wget https://dl.google.com/linux/direct/google-chrome-stable\_current\_amd64.deb

sudo dpkg -i google-chrome\*.deb || sudo apt-get -f install -y

CHROME\_VERSION=$(google-chrome --version | grep -oP "\d+\.\d+\.\d+")

wget -N https://chromedriver.storage.googleapis.com/${CHROME\_VERSION}/chromedriver\_linux64.zip || wget https://chromedriver.storage.googleapis.com/114.0.5735.90/chromedriver\_linux64.zip

unzip chromedriver\_linux64.zip

sudo mv chromedriver /usr/local/bin/

chmod +x /usr/local/bin/chromedriver

- name: Run Tests with Xvfb

run: |

sudo apt-get install -y xvfb

xvfb-run --auto-servernum python3 tests/data\_driven\_test.py

- name: Archive Log

uses: actions/upload-artifact@v3

with:

name: test-log

path: test\_log.txt

INTERNSHIP TASK -4 SECURITY TESTING AUTOMATION I NSTRUCTIONS: USE TOOLS LIKE OWASP ZAP OR BURP SUITE TO AUTOMATE SECURITY TESTING FOR A WEB APPLICATION. DELIVERABLE: A REPORT DETAILING VULNERABILITIES AND REMEDIATION STEPS.

from zapv2 import ZAPv2

import time

target = 'http://your-web-app-url.com'

zap = ZAPv2()

# Access target

print(f'Scanning target: {target}')

zap.urlopen(target)

time.sleep(2)

# Spidering

print('Spidering...')

zap.spider.scan(target)

time.sleep(5)

while int(zap.spider.status()) < 100:

time.sleep(2)

print('Spidering complete')

# Active Scan

print('Starting active scan...')

zap.ascan.scan(target)

while int(zap.ascan.status()) < 100:

time.sleep(5)

print('Scan complete')

# Generate Report

with open('zap\_report.html', 'w') as f:

f.write(zap.core.htmlreport())

print('Report saved as zap\_report.html')

(BACK END) INTERNSHIP TASK -1 BLOG BACKEND SYSTEM I NSTRUCTIONS: CREATE THE BACKEND FOR A BLOG PLATFORM WITH FEATURES LIKE USER AUTHENTICATION, BLOG CREATION, AND COMMENTING USING A FRAMEWORK LIKE DJANGO OR EXPRESS.JS DELIVERABLE: A FUNCTIONAL API WITH A CONNECTED DATABASE (MYSQL/POSTGRESQL)

module.exports = (sequelize, DataTypes) => {

const Blog = sequelize.define("Blog", {

title: DataTypes.STRING,

content: DataTypes.TEXT,

});

Blog.associate = (models) => {

Blog.belongsTo(models.User);

Blog.hasMany(models.Comment);

};

return Blog;

};

INTERNSHIP TASK -2 REAL-TIME CHAT SERVER I NSTRUCTIONS: DEVELOP A REAL-TIME CHAT BACKEND USING WEBSOCKETS OR SOCKET.IO WITH SUPPORT FOR MULTIPLE CHAT ROOMS. DELIVERABLE: A NODE.JS OR PYTHON BASED SERVER HANDLING REAL-TIME COMMUNICATION

const express = require('express');

const http = require('http');

const socketIO = require('socket.io');

const app = express();

const server = http.createServer(app);

const io = socketIO(server);

const PORT = 3000;

io.on('connection', socket => {

console.log('User connected:', socket.id);

socket.on('joinRoom', ({ room, username }) => {

socket.join(room);

io.to(room).emit('message', `${username} joined room: ${room}`);

});

socket.on('chatMessage', ({ room, username, message }) => {

io.to(room).emit('message', `${username}: ${message}`);

});

socket.on('disconnect', () => {

console.log('User disconnected:', socket.id);

});

});

server.listen(PORT, () => console.log(`Server running on port ${PORT}`));

INTERNSHIP TASK -3 E-COMMERCE BACKEND I NSTRUCTIONS: DEVELOP A BACKEND SYSTEM FOR AN E COMMERCE PLATFORM, INCLUDING USER AUTHENTICATION, PRODUCT MANAGEMENT, AND ORDER PROCESSING. DELIVERABLE: A FULLY FUNCTIONAL API USING A MODERN FRAMEWORK AND DATABASE

**Core Features to Implement**

1. **User Authentication**
   * Register, login, logout
   * JWT-based token authentication
2. **Product Management**
   * CRUD operations for products (admin)
   * View product list and details (public)
3. **Order Processing**
   * Add to cart, checkout, place order
   * Track order status (user & admin)

**✅ Suggested Tech Stack**

**Option A: Django + Django REST Framework**

* DB: PostgreSQL or MySQL
* Auth: SimpleJWT
* Admin panel built-in

**Option B: Express.js + MongoDB or MySQL**

* Auth: jsonwebtoken, bcrypt
* DB: Mongoose or Sequelize

**✅ Recommended API Endpoints**

**🔐 Auth**

* POST /api/register
* POST /api/login
* POST /api/logout

**📦 Products**

* GET /api/products/
* GET /api/products/:id
* POST /api/products/ *(admin only)*
* PUT /api/products/:id
* DELETE /api/products/:id

**🛒 Orders**

* POST /api/orders/ – place order
* GET /api/orders/ – user's orders
* GET /api/orders/:id – order detail
* PUT /api/orders/:id/status *(admin only)*

**✅ Deliverables**

* ✅ API backend with routes, models, controllers
* ✅ Connected to PostgreSQL/MySQL
* ✅ Postman collection or Swagger UI for testing
* ✅ README with:
  + Setup instructions
  + API documentation
  + ER diagram / DB schema (optional)

INTERNSHIP TASK -4 AI-POWERED RECOMMENDATION ENGINE I NSTRUCTIONS: DEVELOP A RECOMMENDATION ENGINE BACKEND (E.G., FOR MOVIES, PRODUCTS) USING MACHINE LEARNING LIBRARIES AND EXPOSE IT AS A RESTFUL API. DELIVERABLE: A PYTHON-BASED BACKEND WITH ENDPOINTS FOR FETCHING RECOMMENDATIONS

import pandas as pd

from sklearn.metrics.pairwise import cosine\_similarity

from sklearn.feature\_extraction.text import TfidfVectorizer

# Load and process data

df = pd.read\_csv("data/movies.csv") # columns: id, title, description

tfidf = TfidfVectorizer(stop\_words='english')

tfidf\_matrix = tfidf.fit\_transform(df['description'])

similarity = cosine\_similarity(tfidf\_matrix, tfidf\_matrix)

# Recommend top-N items similar to a given item

def get\_recommendations(movie\_id, top\_n=5):

idx = df.index[df['id'] == movie\_id].tolist()[0]

scores = list(enumerate(similarity[idx]))

scores = sorted(scores, key=lambda x: x[1], reverse=True)

movie\_indices = [i[0] for i in scores[1:top\_n+1]]

return df['title'].iloc[movie\_indices].tolist()

from fastapi import FastAPI

from recommend import get\_recommendations

app = FastAPI()

@app.get("/recommend/{movie\_id}")

def recommend(movie\_id: int):

recs = get\_recommendations(movie\_id)

return {"recommendations": recs}

(BIG DATA) INTERNSHIP TASK -1 DATA CLEANING WITH PYSPARK I NSTRUCTIONS: USE PYSPARK TO CLEAN AND PREPROCESS A LARGE DATASET, HANDLING MISSING VALUES AND DUPLICATES. DELIVERABLE: A PYTHON SCRIPT OR NOTEBOOK SHOWCASING THE DATA CLEANING PROCESS

from pyspark.sql import SparkSession

from pyspark.sql.functions import col

# Initialize Spark

spark = SparkSession.builder.appName("DataCleaning").getOrCreate()

# Load dataset (e.g., CSV)

df = spark.read.csv("data/your\_dataset.csv", header=True, inferSchema=True)

# Show initial info

df.printSchema()

df.show(5)

# Drop duplicate rows

df = df.dropDuplicates()

# Count nulls per column

df.select([col(c).isNull().alias(c) for c in df.columns]).show()

# Fill missing values (example: fill numeric columns with 0)

df = df.fillna({"column1": 0, "column2": "unknown"})

# Drop rows with any nulls

# df = df.dropna()

# Save cleaned data

df.write.csv("output/cleaned\_dataset.csv", header=True, mode="overwrite")

INTERNSHIP TASK -2 DISTRIBUTED DATA PROCESSING I NSTRUCTIONS: USE APACHE SPARK TO ANALYZE A LARGE DATASET, IMPLEMENTING OPERATIONS LIKE FILTERING, GROUPING, AND AGGREGATIONS. DELIVERABLE: A SPARK JOB SCRIPT WITH OUTPUT SHOWING ANALYSIS RESULTS

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, avg, count

# Initialize Spark session

spark = SparkSession.builder.appName("DistributedProcessing").getOrCreate()

# Load dataset

df = spark.read.csv("data/your\_dataset.csv", header=True, inferSchema=True)

# Filter example: Select only active users

filtered\_df = df.filter(col("status") == "active")

# Group and aggregate: Average age per region

grouped\_df = filtered\_df.groupBy("region").agg(

avg("age").alias("average\_age"),

count("\*").alias("user\_count")

)

# Show results

grouped\_df.show()

# Save results

grouped\_df.write.csv("output/region\_wise\_user\_stats.csv", header=True, mode="overwrite")

INTERNSHIP TASK -3 DATA VISUALIZATION WITH BIG DATA TOOLS I NSTRUCTIONS: USE A TOOL LIKE TABLEAU OR POWER BI TO VISUALIZE INSIGHTS FROM A LARGE DATASET. DELIVERABLE: A SET OF INTERACTIVE DASHBOARDS SHOWCASING KEY INSIGHTS

**✅ Goal**

Use **Tableau** or **Power BI** to create **interactive dashboards** that highlight insights from a **large dataset**.

**✅ Steps to Complete**

1. **Choose or Load a Dataset**
   * Can be your own or something like:
     + Kaggle datasets
     + Spark output (from previous tasks)
     + CSVs or SQL databases
2. **Clean and Prepare Data**
   * Use Power Query (Power BI) or Data Source tab (Tableau)
   * Ensure date/time formats, numeric columns, and categories are clean.
3. **Visualizations to Include**
   * Bar Charts (e.g., Sales by Region)
   * Line Graphs (e.g., Monthly Trends)
   * Pie/Donut Charts (e.g., Category Distribution)
   * Heatmaps/Maps (e.g., User Density by State)
   * KPIs (e.g., Total Revenue, Avg Order Size)
4. **Make It Interactive**
   * Use filters (date range, category)
   * Add tooltips, slicers, drill-downs
   * Use bookmarks or navigation buttons for multi-tab dashboards

**✅ Example Use Case (If Dataset Is E-commerce)**

* **Total Sales by Product Category**
* **Customer Acquisition by Month**
* **Sales by Region/City**
* **Top 10 Customers by Spend**

**✅ Deliverables**

* Power BI: .pbix file + screenshots
* Tableau: .twb or .twbx file + screenshots
* (Optional) PDF Export of Dashboard Views

INTERNSHIP TASK -4 BIG DATA SECURITY AND COMPLIANCE I NSTRUCTIONS: DESIGN A FRAMEWORK TO ENSURE SECURITY AND COMPLIANCE (E.G., GDPR, HIPAA) FOR A BIG DATA SYSTEM. DELIVERABLE: A REPORT OR PROTOTYPE DEMONSTRATING SECURE DATA HANDLING PRACTICES

**✅ Suggested Report Structure or Prototype Components**

**1. Overview**

* Define big data security challenges (e.g., data volume, variety, velocity).
* Mention the importance of compliance in data handling.

**2. Regulatory Focus**

Choose and summarize key points from regulations:

* **GDPR** (EU): Consent, Right to Erasure, Data Minimization, Encryption.
* **HIPAA** (US Healthcare): PHI protection, Access Controls, Audit Logs.

**3. Architecture for Secure Big Data**

Create a diagram or outline covering:

* **Data Ingestion Security** (TLS, source validation)
* **Storage Security** (encryption at rest using tools like HDFS encryption, S3 server-side encryption)
* **Access Controls** (Role-Based Access Control, IAM policies)
* **Data Masking/Anonymization** (for sensitive fields)
* **Logging & Monitoring** (SIEM integration)

**4. Technologies/Tools to Use**

* **Apache Ranger / Sentry** – for access control
* **Kerberos** – for authentication
* **Apache Atlas** – for metadata governance
* **GDPR/PII scanners** – e.g., AWS Macie, Google DLP
* **Hadoop + Knox Gateway** – secure REST APIs

**5. Compliance Checks and Auditing**

* Describe how audits are logged.
* Describe breach response policies.
* Demonstrate sample audit log outputs.

**6. Prototype (Optional if Report + Framework)**

* Use **Jupyter Notebook or HTML report** to simulate:
  + Data encryption
  + Access control scenarios
  + PII masking/anonymization functions

**7. Deliverable Tips**

* Export your final document as a PDF or structured Word file.
* Include diagrams using tools like draw.io or Lucidchart.

BLOCK CHAIN TECHNOLOGIES.pdf"

INTERNSHIP TASK -1 BASIC SMART CONTRACT DEVELOPMENT I NSTRUCTIONS: CREATE A SIMPLE SMART CONTRACT USING SOLIDITY FOR A TOKEN TRANSFER SYSTEM. DELIVERABLE: THE SOLIDITY CODE AND DEPLOYMENT STEPS ON A TEST ETHEREUM NETWORK.

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.20;

contract SimpleToken {

string public name = "MyToken";

string public symbol = "MTK";

uint8 public decimals = 18;

uint256 public totalSupply;

mapping(address => uint256) public balanceOf;

event Transfer(address indexed from, address indexed to, uint256 value);

constructor(uint256 \_initialSupply) {

totalSupply = \_initialSupply \* (10 \*\* uint256(decimals));

balanceOf[msg.sender] = totalSupply;

}

function transfer(address \_to, uint256 \_value) public returns (bool success) {

require(balanceOf[msg.sender] >= \_value, "Insufficient balance");

balanceOf[msg.sender] -= \_value;

balanceOf[\_to] += \_value;

emit Transfer(msg.sender, \_to, \_value);

return true;

}

}

**✅ 2. Deployment Steps (Using Remix + MetaMask)**

**🧪 Prerequisites:**

* MetaMask installed and connected to a **testnet** (e.g., Goerli)
* Test ETH in your MetaMask wallet (get from [Goerli faucet](https://goerlifaucet.com/))
* Open Remix IDE

**🛠 Steps:**

1. Paste the code into **Remix → Contracts → New File.sol**
2. Go to **Solidity Compiler** tab → Compile the contract
3. Go to **Deploy & Run Transactions**
   * Select **Injected Provider - MetaMask**
   * Choose the contract and click **Deploy**
4. Confirm the transaction in MetaMask
5. After deployment:
   * Use the transfer() function to test token transfers between addresses

**✅ 3. Deliverable Format**

You should submit:

* SimpleToken.sol (Solidity file)
* A PDF with:
  + Deployment screenshots (Remix + MetaMask)
  + Test transactions (e.g., transferring tokens)
  + Short explanation of how it works

INTERNSHIP TASK -2 TOKEN CREATION (ERC-20) I NSTRUCTIONS: CREATE A CUSTOM ERC-20 TOKEN USING SOLIDITY AND DEPLOY IT ON AN ETHEREUM TESTNET. DELIVERABLE: SMART CONTRACT CODE, TOKEN DETAILS, AND DEPLOYMENT PROOF

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract SimpleToken {

mapping(address => uint256) public balanceOf;

constructor(uint256 initialSupply) {

balanceOf[msg.sender] = initialSupply;

}

function transfer(address recipient, uint256 amount) public returns (bool) {

require(balanceOf[msg.sender] >= amount, "Not enough tokens");

balanceOf[msg.sender] -= amount;

balanceOf[recipient] += amount;

return true;

}

}

**🧾 Instructions:**  
Create a **simple smart contract** using **Solidity** for a **token transfer system**.

**🎯 Objective:**  
To build a basic understanding of how smart contracts work and how token transfers are handled on the Ethereum blockchain.

**📁 Deliverables:**

* Solidity smart contract code
* Deployment steps (using Remix, MetaMask, and an Ethereum test network like Goerli or Sepolia)

**📌 Implementation Steps:**

1. **Write the Smart Contract**
   * Define a token with:
     + balanceOf mapping
     + transfer() function
   * Example (simple version):

solidity

Copy code

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract SimpleToken {

mapping(address => uint256) public balanceOf;

constructor(uint256 initialSupply) {

balanceOf[msg.sender] = initialSupply;

}

function transfer(address recipient, uint256 amount) public returns (bool) {

require(balanceOf[msg.sender] >= amount, "Not enough tokens");

balanceOf[msg.sender] -= amount;

balanceOf[recipient] += amount;

return true;

}

}

1. **Deploy the Contract:**
   * Use **Remix IDE** (https://remix.ethereum.org)
   * Connect **MetaMask** with testnet tokens (Goerli/Sepolia via faucet)
   * Compile and deploy the contract
   * Interact with it using Remix or frontend (optional)
2. **Documentation:**
   * Add comments to code
   * Provide step-by-step screenshots or written guide of deployment
   * Mention wallet address and transaction hash as proof

INTERNSHIP TASK -3 DECENTRALIZED FINANCE (DEFI) APPLICATION I NSTRUCTIONS: DEVELOP A DEFI APPLICATION FOR LENDING AND BORROWING TOKENS WITH INTEREST RATES CALCULATED DYNAMICALLY. DELIVERABLE: A DAPP WITH SMART CONTRACTS, A FRONTEND, AND INTEGRATION WITH A TEST BLOCKCHAIN.

**🧾 Instructions:**

Develop a **DeFi application** for **lending and borrowing tokens** with **dynamically calculated interest rates**.

**🎯 Goal:**

To build a **Decentralized Application (DApp)** that allows:

* **Lenders** to deposit tokens and earn interest
* **Borrowers** to borrow tokens by providing collateral
* **Smart contracts** that dynamically adjust interest rates based on supply/demand

**📁 Deliverables:**

* ✅ Smart Contracts (Solidity)
* ✅ Frontend DApp (React.js or similar)
* ✅ Integration with Ethereum testnet (Goerli/Sepolia via MetaMask)
* ✅ Documentation: How to use and deploy

**🧱 Architecture Overview:**

| **Component** | **Tool/Tech** |
| --- | --- |
| Smart Contract | Solidity (ERC-20, lending logic) |
| Frontend | React.js + Web3.js or Ethers.js |
| Blockchain | Ethereum Testnet (Goerli/Sepolia) |
| Wallet | MetaMask |

**🔐 Smart Contract Features:**

1. **Token Support**:
   * Use existing ERC-20 token (e.g., DAI/USDC on testnet) or custom token.
2. **Lending Functionality**:
   * Lenders deposit tokens
   * Get **interest-bearing tokens** in return (e.g., aToken)
   * Balance accrues interest dynamically
3. **Borrowing Functionality**:
   * Borrowers supply collateral (e.g., ETH or stablecoins)
   * Can borrow against collateral (Loan-to-Value ratio, e.g., 75%)
   * Interest charged and must be repaid
4. **Interest Rate Model**:
   * Dynamic interest rates based on utilization:

solidity

Copy code

function calculateInterestRate(uint256 totalSupplied, uint256 totalBorrowed) public pure returns (uint256) {

uint256 utilization = (totalBorrowed \* 1e18) / totalSupplied;

if (utilization < 80 \* 1e16) {

return 5 \* 1e16; // 5% base interest

} else {

return 10 \* 1e16; // 10% interest if high utilization

}

}

**🖥️ Frontend (DApp):**

* Connect to wallet (MetaMask)
* Display:
  + Total liquidity
  + Total borrowed
  + Current interest rate
* Features:
  + Deposit tokens
  + Borrow tokens
  + Repay loans
  + Withdraw deposits
* Use ethers.js or web3.js to interact with contracts

**🔗 Integration Steps:**

1. **Set up contracts on Remix or Hardhat**
2. **Deploy to testnet (Goerli/Sepolia)**
3. **Connect frontend with contract ABI + address**
4. **Test all flows: deposit, borrow, repay, withdraw**
5. **Use faucets for test tokens (Goerli ETH, USDC, etc.)**

**🧪 Testing Tools:**

* Remix IDE or Hardhat for contract development
* MetaMask for wallet interaction
* Etherscan (testnet) for verification

**📘 Sample Contract Repository Structure:**

java

Copy code

/defi-app/

├── contracts/

│ └── LendingProtocol.sol

├── frontend/

│ ├── public/

│ ├── src/

│ │ ├── App.js

│ │ └── web3.js

├── scripts/

│ └── deploy.js

├── package.json

└── README.md

**✅ Final Deliverable Should Include:**

* ✅ Deployed contract address + ABI
* ✅ Working frontend demo (localhost or hosted)
* ✅ Screenshot or video walkthrough
* ✅ Deployment & usage instructions in README

INTERNSHIP TASK -4 PRIVATE BLOCKCHAIN IMPLEMENTATION I NSTRUCTIONS: SET UP A PRIVATE BLOCKCHAIN USING HYPERLEDGER FABRIC OR ETHEREUM AND DEPLOY A SAMPLE APPLICATION. DELIVERABLE: BLOCKCHAIN CONFIGURATION FILES, DEPLOYMENT STEPS, AND A WORKING DAPP.

### 🧾 ****Instructions****:

Set up a **Private Blockchain** using **Hyperledger Fabric** or **Ethereum** and deploy a **sample application (DApp)** on it.

### 🎯 ****Goal****:

Create a **private blockchain network** (not connected to public Ethereum) and run a **smart contract-based app** to demonstrate transaction handling and permissioned control.

**🛠️ TECHNOLOGY CHOICES:**

| **Option** | **Description** |
| --- | --- |
| 🔷 Ethereum (Private) | Use Geth to spin up a local Ethereum network with multiple nodes |
| 🔶 Hyperledger Fabric | Use Docker to set up permissioned blockchain with channels, peers, and CAs |

**OPTION 1️⃣: 🔷 Ethereum-Based Private Blockchain**

**🔨 Steps to Set Up a Private Ethereum Network:**

1. **Install Geth**:

bash

Copy code

sudo apt install geth

1. **Create Genesis Block** (genesis.json):

json

Copy code

{

"config": {

"chainId": 2025,

"homesteadBlock": 0,

"eip150Block": 0,

"eip155Block": 0,

"eip158Block": 0,

"byzantiumBlock": 0

},

"difficulty": "1",

"gasLimit": "8000000",

"alloc": {}

}

1. **Initialize Blockchain**:

bash

Copy code

geth --datadir mychain init genesis.json

1. **Start the Node**:

bash

Copy code

geth --datadir mychain --networkid 2025 --http --http.port 8545 --port 30303 --http.api personal,eth,net,web3,miner --allow-insecure-unlock console

1. **Create Account and Start Mining**:

bash

Copy code

personal.newAccount()

miner.start(1)

**📦 Sample DApp to Deploy:**

* **Smart Contract**: Voting, Token, or Supply Chain Tracking
* **Language**: Solidity
* **Deployment Tool**: Truffle or Hardhat
* **Frontend**: React + Web3.js

**📁 Deliverables:**

* genesis.json + networkid
* Contract source + deployment script
* Screenshots of the working DApp on private net
* Documentation: Setup, run, and use

"C LANGUAGE.pdf"

**✅ INTERNSHIP TASK - 1: File Handling Program in C**

**📌 Instructions:**

Write a program to **create**, **read**, **write**, and **append** data to a file.

**🧾 Features to Implement:**

* Create a new file.
* Write to a file.
* Append to a file.
* Read contents of a file.

**🧠 Sample Code:**

c

Copy code

#include <stdio.h>

void createAndWriteFile() {

FILE \*f = fopen("sample.txt", "w");

if (f == NULL) {

printf("File creation failed.\n");

return;

}

fprintf(f, "Hello, this is a test.\n");

fclose(f);

}

void appendToFile() {

FILE \*f = fopen("sample.txt", "a");

if (f == NULL) {

printf("File append failed.\n");

return;

}

fprintf(f, "Appending new line to the file.\n");

fclose(f);

}

void readFile() {

char ch;

FILE \*f = fopen("sample.txt", "r");

if (f == NULL) {

printf("File read failed.\n");

return;

}

while ((ch = fgetc(f)) != EOF) {

putchar(ch);

}

fclose(f);

}

int main() {

createAndWriteFile();

appendToFile();

readFile();

return 0;

}

**✅ INTERNSHIP TASK - 2: Linked List Implementation in C**

**📌 Instructions:**

Implement a **singly linked list** with the following operations:

* Insert node
* Delete node
* Traverse the list

**🧠 Sample Code:**

c

Copy code

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

void insert(struct Node\*\* head, int data) {

struct Node\* newNode = malloc(sizeof(struct Node));

newNode->data = data;

newNode->next = \*head;

\*head = newNode;

}

void delete(struct Node\*\* head, int key) {

struct Node \*temp = \*head, \*prev = NULL;

if (temp != NULL && temp->data == key) {

\*head = temp->next;

free(temp);

return;

}

while (temp != NULL && temp->data != key) {

prev = temp;

temp = temp->next;

}

if (temp == NULL) return;

prev->next = temp->next;

free(temp);

}

void traverse(struct Node\* head) {

while (head != NULL) {

printf("%d -> ", head->data);

head = head->next;

}

printf("NULL\n");

}

int main() {

struct Node\* head = NULL;

insert(&head, 10);

insert(&head, 20);

insert(&head, 30);

traverse(head);

delete(&head, 20);

traverse(head);

return 0;

}

**✅ INTERNSHIP TASK - 3: Lexical Analyzer in C**

**📌 Instructions:**

Build a simple program that detects **keywords**, **identifiers**, and **operators** from an input file.

**🧠 Sample Code:**

c

Copy code

#include <stdio.h>

#include <ctype.h>

#include <string.h>

char keywords[5][10] = {"int", "float", "if", "else", "return"};

int isKeyword(char\* word) {

for (int i = 0; i < 5; i++) {

if (strcmp(word, keywords[i]) == 0)

return 1;

}

return 0;

}

int main() {

char ch, buffer[20];

int i = 0;

FILE \*f = fopen("input.c", "r");

if (f == NULL) {

printf("File not found.\n");

return 1;

}

while ((ch = fgetc(f)) != EOF) {

if (isalnum(ch)) {

buffer[i++] = ch;

} else if (ch == ' ' || ch == '\n') {

buffer[i] = '\0';

i = 0;

if (strlen(buffer) > 0) {

if (isKeyword(buffer))

printf("%s is a Keyword\n", buffer);

else

printf("%s is an Identifier\n", buffer);

}

} else {

printf("%c is an Operator\n", ch);

}

}

fclose(f);

return 0;

}

**✅ INTERNSHIP TASK - 4: Data Compression Tool (Run-Length Encoding)**

**📌 Instructions:**

Implement **Run-Length Encoding (RLE)** for compressing a string like:

* Input: "aaaabbbcc"
* Output: "a4b3c2"

**🧠 Sample Code:**

c

Copy code

#include <stdio.h>

#include <string.h>

void compressRLE(char \*str) {

int len = strlen(str);

for (int i = 0; i < len; i++) {

int count = 1;

while (i < len - 1 && str[i] == str[i + 1]) {

count++;

i++;

}

printf("%c%d", str[i], count);

}

printf("\n");

}

int main() {

char str[100];

printf("Enter a string: ");

scanf("%s", str);

printf("Compressed string: ");

compressRLE(str);

return 0;

}

**📂 Suggested Folder Structure for Submission:**

arduino

Copy code

C\_LANGUAGE/

├── Task1\_FileHandling.c

├── Task2\_LinkedList.c

├── Task3\_LexicalAnalyzer.c

├── Task4\_DataCompression.c

├── input.c ← (for Task 3)

└── README.md

" C++.pdf"

**✅ INTERNSHIP TASK - 1: File Management Tool in C++**

**📌 Instructions:**

Create a program that performs:

* Reading
* Writing
* Appending to **text files**.

**🧠 Sample Code:**

cpp

Copy code

#include <iostream>

#include <fstream>

using namespace std;

void writeToFile() {

ofstream file("data.txt");

if (file.is\_open()) {

file << "This is a write operation.\n";

file.close();

} else {

cout << "Failed to open file.\n";

}

}

void appendToFile() {

ofstream file("data.txt", ios::app);

if (file.is\_open()) {

file << "This is an append operation.\n";

file.close();

} else {

cout << "Failed to open file.\n";

}

}

void readFromFile() {

string line;

ifstream file("data.txt");

if (file.is\_open()) {

while (getline(file, line)) {

cout << line << endl;

}

file.close();

} else {

cout << "Failed to open file.\n";

}

}

int main() {

writeToFile();

appendToFile();

readFromFile();

return 0;

}

**✅ INTERNSHIP TASK - 2: Multithreaded File Compression Tool in C++**

**📌 Instructions:**

Build a tool to **compress and decompress files** using **multithreading** (e.g., using Run-Length Encoding for compression).

**🔧 Tools:**

* Use std::thread from <thread>.
* Use mutexes if writing to shared data.

**🧠 Sample Concept Code (Multithreaded RLE compression):**

cpp

Copy code

#include <iostream>

#include <fstream>

#include <thread>

#include <mutex>

#include <vector>

using namespace std;

mutex mtx;

string runLengthEncode(const string& input) {

string result;

int n = input.length();

for (int i = 0; i < n; i++) {

int count = 1;

while (i < n - 1 && input[i] == input[i + 1]) {

count++;

i++;

}

result += input[i] + to\_string(count);

}

return result;

}

void compressChunk(const string& chunk, string& result) {

string encoded = runLengthEncode(chunk);

lock\_guard<mutex> lock(mtx);

result += encoded;

}

int main() {

ifstream in("input.txt");

string data((istreambuf\_iterator<char>(in)), istreambuf\_iterator<char>());

in.close();

int numThreads = 2;

int chunkSize = data.length() / numThreads;

vector<thread> threads;

string compressed;

for (int i = 0; i < numThreads; i++) {

int start = i \* chunkSize;

int end = (i == numThreads - 1) ? data.length() : (i + 1) \* chunkSize;

string chunk = data.substr(start, end - start);

threads.emplace\_back(compressChunk, chunk, ref(compressed));

}

for (auto& t : threads) t.join();

ofstream out("compressed.txt");

out << compressed;

out.close();

cout << "Compression complete.\n";

return 0;

}

**✅ INTERNSHIP TASK - 3: Snake Game Using C++ (SFML)**

**📌 Instructions:**

Develop a **graphical snake game** using libraries like **SFML**.

**🧾 Requirements:**

* Graphical movement.
* Increasing difficulty.
* Sound effects (optional).

**🎮 Sample SFML Setup (Basic Structure):**

⚠️ Requires: Install SFML

cpp

Copy code

#include <SFML/Graphics.hpp>

#include <vector>

using namespace sf;

const int width = 800, height = 600, gridSize = 20;

struct SnakeSegment {

int x, y;

};

int main() {

RenderWindow window(VideoMode(width, height), "Snake Game");

Clock clock;

std::vector<SnakeSegment> snake = { {10, 10} };

int dx = 1, dy = 0;

while (window.isOpen()) {

float time = clock.getElapsedTime().asSeconds();

if (time > 0.1f) {

for (int i = snake.size() - 1; i > 0; --i)

snake[i] = snake[i - 1];

snake[0].x += dx;

snake[0].y += dy;

clock.restart();

}

Event e;

while (window.pollEvent(e)) {

if (e.type == Event::Closed)

window.close();

if (Keyboard::isKeyPressed(Keyboard::Up)) { dx = 0; dy = -1; }

if (Keyboard::isKeyPressed(Keyboard::Down)) { dx = 0; dy = 1; }

if (Keyboard::isKeyPressed(Keyboard::Left)) { dx = -1; dy = 0; }

if (Keyboard::isKeyPressed(Keyboard::Right)) { dx = 1; dy = 0; }

}

window.clear(Color::Black);

for (auto& s : snake) {

RectangleShape rect(Vector2f(gridSize - 2, gridSize - 2));

rect.setPosition(s.x \* gridSize, s.y \* gridSize);

rect.setFillColor(Color::Green);

window.draw(rect);

}

window.display();

}

return 0;

}

**✅ INTERNSHIP TASK - 4: Arithmetic Expression Parser**

**📌 Instructions:**

Build a simple **compiler parser** that:

* Accepts arithmetic expressions.
* Parses and evaluates them.

**🧠 Sample Code (Parser using stacks):**

cpp

Copy code

#include <iostream>

#include <stack>

#include <cctype>

using namespace std;

int precedence(char op) {

if (op == '+' || op == '-') return 1;

if (op == '\*' || op == '/') return 2;

return 0;

}

int applyOp(int a, int b, char op) {

switch (op) {

case '+': return a + b;

case '-': return a - b;

case '\*': return a \* b;

case '/': return b ? a / b : 0;

}

return 0;

}

int evaluate(string expr) {

stack<int> values;

stack<char> ops;

for (int i = 0; i < expr.length(); i++) {

if (isspace(expr[i])) continue;

if (isdigit(expr[i])) {

int val = 0;

while (i < expr.length() && isdigit(expr[i]))

val = (val \* 10) + (expr[i++] - '0');

values.push(val);

i--;

} else if (expr[i] == '(') {

ops.push(expr[i]);

} else if (expr[i] == ')') {

while (!ops.empty() && ops.top() != '(') {

int b = values.top(); values.pop();

int a = values.top(); values.pop();

char op = ops.top(); ops.pop();

values.push(applyOp(a, b, op));

}

ops.pop();

} else {

while (!ops.empty() && precedence(ops.top()) >= precedence(expr[i])) {

int b = values.top(); values.pop();

int a = values.top(); values.pop();

char op = ops.top(); ops.pop();

values.push(applyOp(a, b, op));

}

ops.push(expr[i]);

}

}

while (!ops.empty()) {

int b = values.top(); values.pop();

int a = values.top(); values.pop();

char op = ops.top(); ops.pop();

values.push(applyOp(a, b, op));

}

return values.top();

}

int main() {

string expr;

cout << "Enter an arithmetic expression: ";

getline(cin, expr);

cout << "Result: " << evaluate(expr) << endl;

return 0;

}

**📁 Suggested Folder Structure:**

mathematica

Copy code

C++/

├── Task1\_FileManager.cpp

├── Task2\_MultithreadedCompression.cpp

├── Task3\_SnakeGame.cpp ← (Needs SFML or SDL)

├── Task4\_ArithmeticParser.cpp

├── input.txt ← For Task 2

└── README.md

" CS & EH.pdf"

**🔐 TASK 1: File Integrity Checker (Python + hashlib)**

**Goal:** Detect if a file has been altered by comparing its hash values.

**Deliverable:** Python script that:

* Calculates hash (MD5/SHA256)
* Stores hash in a file
* Rechecks file to detect changes

**✅ I can provide:**

* Full Python script
* How to run and use it
* Sample output and explanation

**🛡 TASK 2: Web Application Vulnerability Scanner**

**Goal:** Detect common vulnerabilities like **SQL Injection** and **XSS**.

**Deliverable:** Python tool using requests, BeautifulSoup, or Selenium.

**✅ I can provide:**

* A basic vulnerability scanner script
* SQLi/XSS test cases
* Code explanations
* How to test on a demo website (like DVWA or local app)

**🧰 TASK 3: Penetration Testing Toolkit**

**Goal:** Modular toolkit with:

* Port scanner
* Brute-force module (e.g., SSH, login forms)
* Banner grabbing

**✅ I can provide:**

* Python scripts for each module
* CLI interface to switch between tools
* Well-commented code + usage examples

**🔒 TASK 4: Advanced Encryption Tool**

**Goal:** Encrypt and decrypt files using **AES-256**.

**Deliverable:** GUI-based tool (Tkinter or CLI) with:

* File selection
* AES encryption/decryption
* Password protection

**✅ I can provide:**

* Full Python implementation using cryptography or pycryptodome
* GUI with Tkinter (optional)
* Test instructions

**⚙️ To proceed:**

Please tell me **which task you'd like to start with**, and I’ll give you:

1. Source code
2. Usage instructions
3. Output samples
4. Report format (if needed)

import hashlib

import json

import os

HASH\_STORE = "hash\_store.json"

def calculate\_hash(file\_path):

sha256 = hashlib.sha256()

try:

with open(file\_path, "rb") as f:

while chunk := f.read(8192):

sha256.update(chunk)

return sha256.hexdigest()

except FileNotFoundError:

print(f"File not found: {file\_path}")

return None

def load\_hashes():

if os.path.exists(HASH\_STORE):

with open(HASH\_STORE, "r") as f:

return json.load(f)

return {}

def save\_hashes(hashes):

with open(HASH\_STORE, "w") as f:

json.dump(hashes, f, indent=4)

def check\_integrity(file\_path):

hashes = load\_hashes()

current\_hash = calculate\_hash(file\_path)

if current\_hash is None:

return

if file\_path in hashes:

if hashes[file\_path] == current\_hash:

print(f"[OK] File is unchanged: {file\_path}")

else:

print(f"[ALERT] File has been modified: {file\_path}")

else:

print(f"[NEW] Storing hash for new file: {file\_path}")

hashes[file\_path] = current\_hash

save\_hashes(hashes)

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

print("=== FILE INTEGRITY CHECKER ===")

while True:

path = input("\nEnter file path to check (or type 'exit' to quit): ").strip()

if path.lower() == "exit":

break

check\_integrity(path

import requests

from bs4 import BeautifulSoup

from urllib.parse import urljoin

# Payloads for testing vulnerabilities

xss\_payload = "<script>alert('XSS')</script>"

sql\_payload = "' OR '1'='1"

def is\_vulnerable\_to\_xss(response):

return xss\_payload in response.text

def is\_vulnerable\_to\_sql\_injection(response):

error\_signatures = [

"you have an error in your sql syntax;",

"unclosed quotation mark",

"warning: mysql",

"sql error"

]

return any(error.lower() in response.text.lower() for error in error\_signatures)

def scan\_url\_for\_vulnerabilities(url):

print(f"\nScanning URL: {url}")

try:

res = requests.get(url)

except requests.RequestException as e:

print(f"Request failed: {e}")

return

soup = BeautifulSoup(res.text, "html.parser")

forms = soup.find\_all("form")

print(f"Found {len(forms)} form(s)")

for form in forms:

action = form.get("action")

method = form.get("method", "get").lower()

target\_url = urljoin(url, action)

inputs = form.find\_all("input")

data = {}

for input\_tag in inputs:

name = input\_tag.get("name")

if name:

data[name] = xs

**✅ INTERNSHIP TASK - 3: PENETRATION TESTING TOOLKIT**

**🔹 Objective:** Create a modular toolkit in Python with multiple components:

* Port Scanner
* Brute Force Login Tester (Basic)
* Network Info Grabber

**📁 penetration\_toolkit.py**

python

Copy code

import socket

import requests

import threading

# --------------------- Module 1: Port Scanner ---------------------

def port\_scanner(target, ports=[21, 22, 23, 80, 443, 8080]):

print(f"\n🔍 Scanning ports on: {target}")

for port in ports:

try:

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.settimeout(1)

result = s.connect\_ex((target, port))

if result == 0:

print(f"✅ Port {port} is OPEN")

s.close()

except Exception as e:

print(f"Error scanning port {port}: {e}")

# --------------------- Module 2: Brute Force Tester ---------------------

def brute\_force\_login(url, username\_list, password\_list):

print(f"\n🔐 Brute Force Attack on: {url}")

for username in username\_list:

for password in password\_list:

data = {'username': username.strip(), 'password': password.strip()}

response = requests.post(url, data=data)

if "login failed" not in response.text.lower():

print(f"✅ Successful Login: {username}/{password}")

return

print("❌ Brute force failed. No valid credentials found.")

# --------------------- Module 3: Network Info ---------------------

def grab\_host\_info(target):

try:

print(f"\n📡 Gathering info for {target}")

ip = socket.gethostbyname(target)

host = socket.gethostbyaddr(ip)

print(f"✅ IP Address: {ip}")

print(f"✅ Host Info: {host}")

except socket.error as e:

print(f"Error fetching host info: {e}")

# --------------------- Main Menu ---------------------

def main():

print("=== 🛠️ PENETRATION TESTING TOOLKIT ===")

while True:

print("\n1. Port Scanner")

print("2. Brute Force Login Tester")

print("3. Network Info Grabber")

print("4. Exit")

choice = input("Select option: ").strip()

if choice == "1":

target = input("Enter target IP/domain: ").strip()

port\_scanner(target)

elif choice == "2":

url = input("Enter login form URL: ").strip()

usernames = input("Enter comma-separated usernames: ").split(",")

passwords = input("Enter comma-separated passwords: ").split(",")

brute\_force\_login(url, usernames, passwords)

elif choice == "3":

target = input("Enter domain: ").strip()

grab\_host\_info(target)

elif choice == "4":

print("Exiting toolkit. 🛡️")

break

else:

print("Invalid choice. Try again.")

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

main()

" DATA ANALYSIS.pdf"

### ✅ ****INTERNSHIP TASK 1: BIG DATA ANALYSIS USING PYSPARK****

#### 🧩 Objective:

Perform analysis on a large dataset using PySpark to demonstrate scalability and processing power.

### 🔧 ****Requirements:****

* Python
* Apache Spark (pyspark package)
* Sample large dataset (e.g., NYC Taxi Trips, COVID-19 dataset)

### 📦 ****Installation:****

bash

Copy code

pip install pyspark

### 📁 ****Sample Dataset:****

We'll use a large CSV dataset (e.g., COVID-19 dataset or any large .csv file with millions of rows).

### 🧪 ****Sample PySpark Script:****

python

Copy code

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, avg, max, min

# Step 1: Initialize Spark Session

spark = SparkSession.builder \

.appName("Big Data Analysis with PySpark") \

.getOrCreate()

# Step 2: Load the dataset

df = spark.read.csv("large\_dataset.csv", header=True, inferSchema=True)

# Step 3: Preview the data

df.show(5)

df.printSchema()

# Step 4: Basic analysis

print("Total Records:", df.count())

print("Columns:", df.columns)

# Step 5: Grouping and Aggregation

# Example: Average of a numeric column grouped by a category

df.groupBy("category\_column").agg(

avg("numeric\_column").alias("avg\_value"),

max("numeric\_column").alias("max\_value"),

min("numeric\_column").alias("min\_value")

).show()

# Step 6: Filtering

filtered\_df = df.filter(col("numeric\_column") > 1000)

filtered\_df.show(5)

# Step 7: Save processed data

filtered\_df.write.csv("output/filtered\_data", header=True)

# Step 8: Stop Spark

spark.stop()

### 📊 ****Example Use Case:****

If using a COVID dataset:

python

Copy code

df.groupBy("Country").agg(avg("Confirmed").alias("AvgConfirmed")).orderBy("AvgConfirmed", ascending=False).show()

" DATA SCIENCE.pdf"

### ✅ ****TASK 1: Data Pipeline Development (ETL with pandas & scikit-learn)****

python

Copy code

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

# Load data

df = pd.read\_csv("data.csv")

# Handle missing values

df.fillna(df.mean(), inplace=True)

# Encode categorical columns

df = pd.get\_dummies(df, drop\_first=True)

# Normalize numerical features

scaler = StandardScaler()

scaled\_features = scaler.fit\_transform(df.drop('target', axis=1))

X = pd.DataFrame(scaled\_features, columns=df.columns[:-1])

y = df['target']

# Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

print("Data pipeline completed. X\_train shape:", X\_train.shape)

### ✅ ****TASK 2: Deep Learning Project (Image Classification using TensorFlow)****

python

Copy code

import tensorflow as tf

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Flatten

# Load MNIST dataset

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# Normalize

x\_train, x\_test = x\_train / 255.0, x\_test / 255.0

# Build model

model = Sequential([

Flatten(input\_shape=(28, 28)),

Dense(128, activation='relu'),

Dense(10, activation='softmax')

])

model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

# Train

model.fit(x\_train, y\_train, epochs=5, validation\_split=0.1)

# Evaluate

test\_loss, test\_acc = model.evaluate(x\_test, y\_test)

print("Test accuracy:", test\_acc)

### ✅ ****TASK 3: End-to-End Data Science Project (Flask API for Model)****

bash

Copy code

# Folder structure:

# ├── app.py

# ├── model.pkl

# └── requirements.txt

**Training & Saving Model (prepare model.pkl):**

python

Copy code

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

import pickle

df = pd.read\_csv('data.csv')

X = df.drop('target', axis=1)

y = df['target']

model = RandomForestClassifier()

model.fit(X, y)

pickle.dump(model, open("model.pkl", "wb"))

**Flask API (app.py):**

python

Copy code

from flask import Flask, request, jsonify

import pickle

import numpy as np

app = Flask(\_\_name\_\_)

model = pickle.load(open("model.pkl", "rb"))

@app.route('/predict', methods=['POST'])

def predict():

data = request.json['input']

prediction = model.predict([np.array(data)])

return jsonify({"prediction": int(prediction[0])})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

### ✅ ****TASK 4: Optimization Model (Linear Programming using PuLP)****

python

Copy code

from pulp import LpProblem, LpVariable, LpMaximize, value

# Problem: Maximize profit

model = LpProblem("Maximize\_Profit", LpMaximize)

# Variables

x = LpVariable('Product\_A', lowBound=0, cat='Integer')

y = LpVariable('Product\_B', lowBound=0, cat='Integer')

# Objective Function: Maximize 40x + 30y

model += 40 \* x + 30 \* y

# Constraints

model += 2 \* x + y <= 100 # Resource 1

model += x + y <= 80 # Resource 2

model.solve()

print("Optimal solution:")

print("Product A:", x.varValue)

print("Product B:", y.varValue)

print("Profit:", value(model.objective))

" DEVEOPS.pdf"

### ✅ ****TASK 1: VERSION CONTROL WITH GIT****

#### 🔧 Instructions:

* Set up a Git repository.
* Create multiple branches.
* Demonstrate merge conflict and resolution.

#### 📂 Deliverables:

* A GitHub/GitLab repo with:
  + main, feature-1, feature-2 branches.
  + Merge conflict scenario and resolution steps documented.

#### ✅ Steps:

1. **Initialize repo:**

bash

Copy code

git init

echo "# DevOps Internship" > README.md

git add README.md

git commit -m "Initial commit"

1. **Create branches:**

bash

Copy code

git checkout -b feature-1

echo "Feature 1 content" > file.txt

git add file.txt

git commit -m "Add Feature 1"

git checkout main

git checkout -b feature-2

echo "Feature 2 content" > file.txt

git add file.txt

git commit -m "Add Feature 2"

1. **Merge & resolve conflict:**

bash

Copy code

git checkout main

git merge feature-1

git merge feature-2

* + Conflict in file.txt will occur. Open it, resolve manually, then:

bash

Copy code

git add file.txt

git commit -m "Resolved merge conflict between feature-1 and feature-2"

1. **Documentation:**
   * Add a file MERGE\_CONFLICT\_RESOLUTION.md:

markdown

Copy code

# Merge Conflict Resolution

- Conflict occurred in `file.txt`

- Feature-1 added: "Feature 1 content"

- Feature-2 added: "Feature 2 content"

- Final resolved version includes both features.

### ✅ ****TASK 2: CI/CD PIPELINE SETUP****

#### 🔧 Instructions:

* Automate web app deployment using **GitHub Actions** or **Jenkins**.

#### 📂 Deliverables:

* A working pipeline YAML (or Jenkinsfile).
* Proof of automated deployment (e.g., screenshots or video demo).

#### 🔧 GitHub Actions Example:

**.github/workflows/deploy.yml**

yaml

Copy code

name: CI/CD Pipeline

on:

push:

branches: [ main ]

jobs:

build-deploy:

runs-on: ubuntu-latest

steps:

- name: Checkout Code

uses: actions/checkout@v3

- name: Set up Node.js

uses: actions/setup-node@v3

with:

node-version: 18

- name: Install Dependencies

run: npm install

- name: Run Build

run: npm run build

- name: Deploy to Netlify (example)

uses: nwtgck/actions-netlify@v1.2

with:

publish-dir: ./build

production-branch: main

github-token: ${{ secrets.GITHUB\_TOKEN }}

netlify-auth-token: ${{ secrets.NETLIFY\_AUTH\_TOKEN }}

netlify-site-id: ${{ secrets.NETLIFY\_SITE\_ID }}

### ✅ ****TASK 3: KUBERNETES DEPLOYMENT****

#### 🔧 Instructions:

* Deploy microservices on Kubernetes.

#### 📂 Deliverables:

* YAML files: deployment.yaml, service.yaml, namespace.yaml, etc.
* Documentation of each file’s purpose.

#### 🛠️ Example Setup:

**deployment.yaml**

yaml

Copy code

apiVersion: apps/v1

kind: Deployment

metadata:

name: webapp-deployment

spec:

replicas: 2

selector:

matchLabels:

app: webapp

template:

metadata:

labels:

app: webapp

spec:

containers:

- name: webapp

image: your-dockerhub/webapp:latest

ports:

- containerPort: 80

**service.yaml**

yaml

Copy code

apiVersion: v1

kind: Service

metadata:

name: webapp-service

spec:

selector:

app: webapp

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

**namespace.yaml**

yaml

Copy code

apiVersion: v1

kind: Namespace

metadata:

name: devops-intern

### ✅ ****TASK 4: SECURE DEVOPS PRACTICES****

#### 🔧 Instructions:

* Integrate **SonarQube** or **OWASP ZAP** in CI/CD.
* Generate vulnerability reports.

#### 📂 Deliverables:

* Pipeline config with scanner integrated.
* Security scan report (zap-report.html, sonarqube-report.xml, etc.)

#### 🛡️ OWASP ZAP + GitHub Actions:

yaml

Copy code

name: Security Scan with ZAP

on:

push:

branches: [ main ]

jobs:

zap\_scan:

runs-on: ubuntu-latest

steps:

- name: OWASP ZAP Baseline Scan

uses: zaproxy/action-baseline@v0.7.0

with:

target: 'https://your-deployed-app.com'

cmd\_options: '-t 60'

#### 📘 SonarQube in GitHub Actions:

yaml

Copy code

- name: SonarQube Scan

uses: SonarSource/sonarcloud-github-action@master

with:

projectBaseDir: .

args: >

-Dsonar.organization=your-org

-Dsonar.projectKey=your-project

env:

SONAR\_TOKEN: ${{ secrets.SONAR\_TOKEN }}

"C: DIGITAL MARKETING.pdf"

## ✅ ****TASK 1: SOCIAL MEDIA POST CALENDAR****

### 🔧 Instructions:

Create a **30-day social media content calendar** for a brand (real or fictional), including:

* Daily **post topics**
* **Captions** with engaging hooks
* Relevant **hashtags**
* **Image/video ideas**

### 📂 Deliverable:

An **Excel or Google Sheet** with columns:  
| Date | Platform | Post Topic | Caption | Hashtags | Image/Video Idea |

### 📌 Example Entries:

| **Date** | **Platform** | **Post Topic** | **Caption** | **Hashtags** | **Image Idea** |
| --- | --- | --- | --- | --- | --- |
| 1st Aug | Instagram | Brand Intro | "Welcome to the future of skincare! ✨" | #Skincare #GlowNaturally | Flat lay of product + logo |
| 2nd Aug | Twitter | Customer Review | "Another happy customer 😍 Read why:" | #CustomerLove #Review | Screenshot of review |
| 3rd Aug | LinkedIn | Founder Story | "From a kitchen experiment to a brand" | #Entrepreneurship #Startup | Old photo of founder |

## ✅ ****TASK 2: CONTENT MARKETING STRATEGY****

### 🔧 Instructions:

Create a **comprehensive content marketing plan** for a brand, including:

* Clear **goals**
* Defined **target audience**
* **Blog topics** or formats
* **Promotion channels**

### 📂 Deliverable:

A **Word/PDF Document** covering the following:

### 📌 Template:

#### **1. Brand Overview**

* Brand Name, Industry, Unique Value Proposition

#### **2. Goals**

* Increase blog traffic by 40% in 3 months
* Grow newsletter subscribers by 1,000

#### **3. Target Audience**

* Age: 20–40
* Interests: Skincare, wellness, eco-friendly products

#### **4. Content Types**

* Blogs, infographics, how-to videos, customer stories

#### **5. Blog Topics**

* “5 Skincare Myths Busted”
* “Morning Routine for Glowing Skin”
* “How Our Ingredients Are Sourced Sustainably”

#### **6. Promotion Channels**

* Email newsletters, Instagram Reels, Facebook Ads, Medium

## ✅ ****TASK 3: SEARCH ENGINE OPTIMIZATION (SEO) STRATEGY****

### 🔧 Instructions:

Design an **SEO strategy** including:

* Keyword strategy
* On-page & technical SEO
* Backlink building plan

### 📂 Deliverables:

1. **SEO Strategy Document**
2. **SEO Checklist for Implementation** (Excel or Doc)

### 📌 Strategy Document Outline:

#### **1. Keyword Research**

* Use tools like Ubersuggest, Ahrefs, or Google Keyword Planner
* Focus: “natural skincare for oily skin,” “eco-friendly moisturizers”

#### **2. On-Page SEO**

* Meta tags, H1-H3 structure
* Internal linking strategy
* Image alt texts

#### **3. Technical SEO**

* Mobile responsiveness
* Page speed optimization (use Google PageSpeed Insights)
* XML Sitemap, robots.txt setup

#### **4. Backlink Building**

* Guest blogging
* Outreach to skincare bloggers
* Submit to directories

### 📋 SEO Checklist:

| **Task** | **Status** | **Notes** |
| --- | --- | --- |
| Add meta titles/descriptions | ✅ |  |
| Optimize page load speed | ⏳ | Work in progress |
| Submit sitemap to Google | ✅ |  |

## ✅ ****TASK 4: BRAND REPUTATION MANAGEMENT****

### 🔧 Instructions:

Create a strategy to:

* Monitor brand reputation
* Respond to negative reviews
* Build positive sentiment

### 📂 Deliverables:

* Strategy Document
* List of monitoring tools

### 📌 Strategy Outline:

#### **1. Monitoring Tools**

* Google Alerts
* Social Mention
* Brand24 or Mention
* Trustpilot / Google Reviews

#### **2. Negative Review Response Framework**

* Respond within 24 hrs
* Apologize and take it offline
* Offer compensation if applicable

#### **3. Positive Sentiment Building**

* Encourage reviews via email campaigns
* Share testimonials on social media
* Collaborate with micro-influencers

#### **4. Monthly Sentiment Analysis**

* Use tools to track brand sentiment
* Create a dashboard (Excel or Google Data Studio)

" EMBEDDED SYSYTEMS.pdf"

**✅ TASK 1: PUSH BUTTON COUNTER**

**🔧 Instructions:**

Create a **counter that increments** every time a **push button** is pressed. Display the count on an **LCD** or via **Serial Monitor**.

**📂 Deliverables:**

* Circuit Diagram
* Arduino Code
* Output Demo (photo/video or screenshot of serial output)

**🔌 Components:**

* Arduino Uno
* Push button
* 10kΩ resistor
* LCD (optional) or Serial Monitor

**💻 Sample Code (Serial Monitor):**

cpp

Copy code

int buttonPin = 2;

int counter = 0;

bool lastState = LOW;

void setup() {

pinMode(buttonPin, INPUT\_PULLUP);

Serial.begin(9600);

}

void loop() {

bool currentState = digitalRead(buttonPin);

if (lastState == HIGH && currentState == LOW) {

counter++;

Serial.println("Button pressed: Count = " + String(counter));

}

lastState = currentState;

delay(50);

}

**✅ TASK 2: HOME AUTOMATION WITH BLUETOOTH**

**🔧 Instructions:**

Control appliances/devices using a **Bluetooth app** (like Bluetooth Terminal or MIT App Inventor app).

**📂 Deliverables:**

* Circuit Diagram
* Arduino Code (with Bluetooth HC-05 module)
* Working Demo (video or screenshots)

**🔌 Components:**

* Arduino Uno
* Bluetooth Module HC-05
* Relay Module or LEDs (to simulate devices)
* Android phone

**💻 Sample Code:**

cpp

Copy code

char btInput;

int device1 = 7;

void setup() {

pinMode(device1, OUTPUT);

Serial.begin(9600);

}

void loop() {

if (Serial.available()) {

btInput = Serial.read();

if (btInput == '1') digitalWrite(device1, HIGH);

else if (btInput == '0') digitalWrite(device1, LOW);

}

}

**Use Bluetooth Terminal** app to send '1' or '0'

**✅ TASK 3: TEMPERATURE MONITORING SYSTEM**

**🔧 Instructions:**

Use a temperature sensor to **read and display data** on an **LCD** or Serial Monitor.

**📂 Deliverables:**

* Circuit Diagram
* Arduino Code
* Output Demo (LCD photo or serial monitor screenshot)

**🔌 Components:**

* Arduino Uno
* LM35 or DHT11/22 Sensor
* LCD (16x2 with I2C) or Serial Monitor

**💻 Sample Code (LM35 + Serial Monitor):**

cpp

Copy code

int tempPin = A0;

void setup() {

Serial.begin(9600);

}

void loop() {

int reading = analogRead(tempPin);

float voltage = reading \* 5.0 / 1024.0;

float temperature = voltage \* 100; // LM35: 10mV per degree

Serial.println("Temp: " + String(temperature) + " °C");

delay(1000);

}

**✅ TASK 4: SPEECH RECOGNITION SYSTEM**

**🔧 Instructions:**

Build a basic voice-controlled system using an embedded board (like Arduino + voice module or ESP32).

**📂 Deliverables:**

* System Design Diagram
* Code
* Working Demo (video showing command-based control)

**🔌 Options:**

* **Option 1**: Arduino + **Voice Recognition Module V3**
* **Option 2**: ESP32 with **Google Assistant or Speech-to-Text app**

**💻 Sample Code (Voice Recognition Module):**

cpp

Copy code

#include <SoftwareSerial.h>

SoftwareSerial mySerial(2, 3); // RX, TX

void setup() {

pinMode(13, OUTPUT);

mySerial.begin(9600);

Serial.begin(9600);

}

void loop() {

if (mySerial.available()) {

int command = mySerial.read();

Serial.println(command);

if (command == 0x11) digitalWrite(13, HIGH); // example voice command

else if (command == 0x12) digitalWrite(13, LOW);

}

}

Train the module with commands like "Turn on" (0x11) and "Turn off" (0x12).

**📦 Want Ready-to-Use Files?**

If you'd like:

* **Circuit diagrams (Fritzing)**
* **Arduino code files (.ino)**
* **Google Docs/Slides for submission**
* **Demo video guide suggestions**

" FIGMA.pdf"

**✅ TASK 1: LOGO DESIGN MOCKUP**

**🔧 Instructions:**

* Design **2–3 variations** of a logo for a brand or fictional company.
* Use different fonts, colors, and layout ideas.

**📂 Deliverable:**

* **Figma file** with:
  + Multiple **logo concepts** on one canvas
  + Optional brand color guide or font choices

**🛠️ Tips:**

* Choose a fictional brand (e.g., "EcoBrew" or "TechNova")
* Use vector shapes and custom typography
* Show mockups on business cards or app icons

**🧩 Suggested Structure:**

* Page 1: Logo Variations (V1, V2, V3)
* Page 2: Logo on Mockup (T-shirt, App Icon, etc.)

**✅ TASK 2: COLLABORATIVE DESIGN WORKFLOW**

**🔧 Instructions:**

* Simulate a real Figma team collaboration:
  + Create a project
  + Share with at least 1-2 collaborators
  + Use **comments**, **version history**, and **shared editing**

**📂 Deliverable:**

* Screenshots or screen recording showing:
  + Shared Figma project
  + Team members commenting/tagging each other
  + Version history with changes

**🛠️ Tips:**

* Add mock comments (e.g., “Can we move this CTA button down?”)
* Use Figma’s “Share” and “Comments” tools
* Enable “Show Version History” (Ctrl+Alt+Shift+H)

**✅ TASK 3: E-COMMERCE PRODUCT PAGE DESIGN**

**🔧 Instructions:**

Design a **modern, interactive e-commerce product page** with:

* Product image (ideally 360° mock)
* Pricing and description
* Add-to-cart button
* Customer reviews

**📂 Deliverable:**

* **Interactive Figma design/prototype**
* Mobile and desktop views (if possible)

**🛠️ Feature Ideas:**

* 360° product view (use carousel or rotating mock image)
* Star rating and user reviews section
* Color/size selector
* Sticky “Add to Cart” button

**🧩 Suggested Layout:**

* Hero product section (left: image, right: info)
* Tabs for "Details", "Reviews", "Shipping Info"
* Related products below

**✅ TASK 4: VOICE-CONTROLLED APP INTERFACE**

**🔧 Instructions:**

Design a **smart home app UI** with voice control flows:

* Commands like “Turn off lights” or “Set thermostat to 22°C”
* Focus on user-friendly, intuitive design

**📂 Deliverable:**

* **Figma prototype** showing voice command interaction flows
* Micro-interactions or animations (optional)

**🛠️ Suggested Features:**

* Mic input icon in header or floating button
* Command history (“You said: Turn off the fan”)
* Confirmation feedback (“Fan turned off”)
* Screens for different rooms/devices

**🎨 Tips:**

* Use Figma's **Prototype mode** to simulate command flow
* Add UI feedback like voice bubbles, loading animation

**📦 Need Templates?**

Let me know if you'd like:

* Logo mockup Figma starter file
* Wireframe template for the product page
* Sample collaborative comments for Task 2
* Ready-made smart home app UI components

int buttonPin = 2;

int counter = 0;

bool lastState = LOW;

void setup() {

pinMode(buttonPin, INPUT\_PULLUP);

Serial.begin(9600);

}

void loop() {

bool currentState = digitalRead(buttonPin);

if (lastState == HIGH && currentState == LOW) {

counter++;

Serial.println("Button pressed. Count = " + String(counter));

delay(200); // Debounce delay

}

lastState = currentState;

}

char btInput;

int device1 = 7; // Connect to LED/relay

void setup() {

pinMode(device1, OUTPUT);

digitalWrite(device1, LOW);

Serial.begin(9600); // Communicates with HC-05

}

void loop() {

if (Serial.available()) {

btInput = Serial.read();

if (btInput == '1') {

digitalWrite(device1, HIGH); // Turn ON

Serial.println("Device ON");

} else if (btInput == '0') {

digitalWrite(device1, LOW); // Turn OFF

Serial.println("Device OFF");

}

}

}

int tempPin = A0;

void setup() {

Serial.begin(9600);

}

void loop() {

int reading = analogRead(tempPin);

float voltage = reading \* 5.0 / 1024.0;

float temperature = voltage \* 100; // LM35 gives 10mV per °C

Serial.print("Temperature: ");

Serial.print(temperature);

Serial.println(" °C");

delay(1000);

}

#include <SoftwareSerial.h>

SoftwareSerial mySerial(2, 3); // RX, TX

int devicePin = 13; // Output pin

void setup() {

Serial.begin(9600);

mySerial.begin(9600);

pinMode(devicePin, OUTPUT);

}

void loop() {

if (mySerial.available()) {

int command = mySerial.read();

Serial.print("Command Received: ");

Serial.println(command);

if (command == 0x11) { // "Turn on light"

digitalWrite(devicePin, HIGH);

} else if (command == 0x12) { // "Turn off light"

digitalWrite(devicePin, LOW);

}

}

}

" FRONT END.pdf"

// TASK 1: INTERACTIVE QUIZ APPLICATION

// Description: Quiz app using JavaScript with instant feedback and scoring.

<!DOCTYPE html>

<html>

<head>

<title>Quiz App</title>

<style>

body { font-family: Arial; margin: 20px; }

.question { margin-bottom: 10px; }

.option { margin: 5px 0; }

</style>

</head>

<body>

<h2>Quiz Application</h2>

<div id="quiz"></div>

<button onclick="submitQuiz()">Submit</button>

<p id="result"></p>

<script>

const quizData = [

{ question: "Capital of France?", options: ["Paris", "London", "Berlin"], answer: 0 },

{ question: "5 + 3 = ?", options: ["6", "8", "9"], answer: 1 }

];

let quizEl = document.getElementById("quiz");

quizData.forEach((q, i) => {

let html = `<div class='question'><strong>${i + 1}. ${q.question}</strong>`;

q.options.forEach((opt, j) => {

html += `<div class='option'><input type='radio' name='q${i}' value='${j}'> ${opt}</div>`;

});

html += `</div>`;

quizEl.innerHTML += html;

});

function submitQuiz() {

let score = 0;

quizData.forEach((q, i) => {

let ans = document.querySelector(`input[name='q${i}']:checked`);

if (ans && parseInt(ans.value) === q.answer) score++;

});

document.getElementById("result").innerText = `Score: ${score}/${quizData.length}`;

}

</script>

</body>

</html>

// TASK 2: REAL-TIME CHAT APPLICATION (FRONTEND ONLY)

// Description: Real-time chat interface using React.js

/\* App.js \*/

import React, { useState, useEffect } from 'react';

import io from 'socket.io-client';

const socket = io('http://localhost:3000');

function App() {

const [messages, setMessages] = useState([]);

const [message, setMessage] = useState("");

useEffect(() => {

socket.on("message", (msg) => setMessages((prev) => [...prev, msg]));

}, []);

const sendMessage = () => {

socket.emit("message", message);

setMessage("");

};

return (

<div>

<h2>Real-Time Chat</h2>

<div style={{ border: "1px solid #ccc", height: 200, overflowY: 'scroll' }}>

{messages.map((msg, i) => <div key={i}>{msg}</div>)}

</div>

<input value={message} onChange={(e) => setMessage(e.target.value)} />

<button onClick={sendMessage}>Send</button>

</div>

);

}

export default App;

// TASK 3: PERSONAL PORTFOLIO WEBPAGE

// Description: Simple responsive portfolio using HTML and CSS

<!DOCTYPE html>

<html>

<head>

<title>My Portfolio</title>

<style>

body { font-family: sans-serif; margin: 0; }

nav { background: #333; color: white; padding: 10px; }

nav a { color: white; margin: 10px; text-decoration: none; }

.section { padding: 20px; }

.projects { display: flex; gap: 20px; flex-wrap: wrap; }

.card { border: 1px solid #ccc; padding: 10px; width: 30%; }

</style>

</head>

<body>

<nav>

<a href="#about">About</a>

<a href="#projects">Projects</a>

<a href="#contact">Contact</a>

</nav>

<div class="section" id="about">

<h2>About Me</h2>

<p>I am a passionate developer with interest in web technologies.</p>

</div>

<div class="section" id="projects">

<h2>Projects</h2>

<div class="projects">

<div class="card">Project 1</div>

<div class="card">Project 2</div>

<div class="card">Project 3</div>

</div>

</div>

<div class="section" id="contact">

<h2>Contact</h2>

<p>Email: me@example.com</p>

</div>

</body>

</html>

// TASK 4: E-LEARNING PLATFORM UI

// Description: Multi-page e-learning UI (HTML + CSS)

<!DOCTYPE html>

<html>

<head>

<title>E-Learning Platform</title>

<style>

body { font-family: Arial; margin: 0; }

.header { background: #007bff; color: white; padding: 15px; }

.course-list { padding: 20px; }

.course { border: 1px solid #ddd; padding: 15px; margin-bottom: 10px; }

.progress { background: #eee; width: 100%; height: 20px; position: relative; }

.bar { background: green; height: 100%; width: 60%; }

</style>

</head>

<body>

<div class="header">

<h1>My Courses</h1>

</div>

<div class="course-list">

<div class="course">

<h3>JavaScript for Beginners</h3>

<p>Progress:</p>

<div class="progress">

<div class="bar"></div>

</div>

<video width="320" height="240" controls>

<source src="video.mp4" type="video/mp4">

</video>

</div>

<div class="course">

<h3>React Crash Course</h3>

<p>Progress:</p>

<div class="progress">

<div class="bar" style="width: 30%;"></div>

</div>

<video width="320" height="240" controls>

<source src="video2.mp4" type="video/mp4">

</video>

</div>

</div>

</body>

</html>

" FULL STACK.pdf"

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Weather App</title>

<style>

body { font-family: Arial; text-align: center; padding: 20px; }

#weather { margin-top: 20px; }

</style>

</head>

<body>

<h1>Weather App</h1>

<input type="text" id="city" placeholder="Enter city">

<button onclick="getWeather()">Get Weather</button>

<div id="weather"></div>

<script>

async function getWeather() {

const city = document.getElementById('city').value;

const res = await fetch(`https://api.weatherapi.com/v1/current.json?key=demo&q=${city}`);

const data = await res.json();

document.getElementById('weather').innerHTML = `

<h2>${data.location.name}</h2>

<p>${data.current.temp\_c} °C</p>

<p>${data.current.condition.text}</p>

`;

}

</script>

</body>

</html>

```

---

### \*\*INTERNSHIP TASK - 2: Chat Application\*\*

\*\*Instructions:\*\* Develop a real-time chat application using WebSocket or Socket.IO.

\*\*Deliverable:\*\* A live chat app with frontend and backend integration.

\*\*Frontend (HTML + Socket.io Client)\*\*

```html

<!DOCTYPE html>

<html>

<head>

<title>Chat App</title>

</head>

<body>

<h2>Real-Time Chat</h2>

<div id="messages"></div>

<input id="msg" autocomplete="off">

<button onclick="send()">Send</button>

<script src="/socket.io/socket.io.js"></script>

<script>

const socket = io();

socket.on('message', msg => {

const div = document.getElementById('messages');

div.innerHTML += `<p>${msg}</p>`;

});

function send() {

const msg = document.getElementById('msg').value;

socket.emit('message', msg);

}

</script>

</body>

</html>

```

\*\*Backend (Node.js + Socket.io)\*\*

```javascript

const express = require('express');

const http = require('http');

const socketIo = require('socket.io');

const app = express();

const server = http.createServer(app);

const io = socketIo(server);

app.use(express.static(\_\_dirname));

io.on('connection', socket => {

socket.on('message', msg => io.emit('message', msg));

});

server.listen(3000, () => console.log('Server started on port 3000'));

```

---

### \*\*INTERNSHIP TASK - 3: Real-Time Collaborative Document Editor\*\*

\*\*Instructions:\*\* Use frameworks like React.js or Vue.js for the front-end, Node.js/Python for backend, MongoDB/PostgreSQL for storage.

\*\*Deliverable:\*\* A real-time collaborative editor.

\*\*Tech Stack:\*\* React.js + Socket.IO + Node.js + MongoDB

\*\*Frontend (React Snippet)\*\*

```jsx

import React, { useEffect, useState } from 'react';

import io from 'socket.io-client';

const socket = io('http://localhost:3000');

function Editor() {

const [text, setText] = useState('');

useEffect(() => {

<h2>Projects</h2>

<div class="projects">

<div class="card">Project 1</div>

<div class="card">Project 2</div>

<div class="card">Project 3</div>

</div>

</div>

<div class="section" id="contact">

<h2>Contact</h2>

<p>Email: me@example.com</p>

</div>

</body>

</html>

// TASK 4: E-LEARNING PLATFORM UI

// Description: Multi-page e-learning UI (HTML + CSS)

<!DOCTYPE html>

<html>

<head>

<title>E-Learning Platform</title>

<style>

body { font-family: Arial; margin: 0; }

.header { background: #007bff; color: white; padding: 15px; }

.course-list { padding: 20px; }

.course { border: 1px solid #ddd; padding: 15px; margin-bottom: 10px; }

.progress { background: #eee; width: 100%; height: 20px; position: relative; }

.bar { background: green; height: 100%; width: 60%; }

</style>

</head>

<body>

<div class="header">

<h1>My Courses</h1>

</div>

<div class="course-list">

<div class="course">

<h3>JavaScript for Beginners</h3>

<p>Progress:</p>

<div class="progress">

<div class="bar"></div>

</div>

<video width="320" height="240" controls>

<source src="video.mp4" type="video/mp4">

</video>

</div>

<div class="course">

<h3>React Crash Course</h3>

<p>Progress:</p>

<div class="progress">

<div class="bar" style="width: 30%;"></div>

</div>

<video width="320" height="240" controls>

<source src="video2.mp4" type="video/mp4">

</video>

</div>

</div>

</body>

</html>

**" IOT.pdf"**

\*\*INTERNSHIP TASK -1: SMART LIGHT CONTROL\*\*

\*\*Objective:\*\* Control an LED using Arduino and a mobile app.

\*\*Components:\*\*

- Arduino Uno

- LED

- Bluetooth Module (HC-05)

- Android App (built with MIT App Inventor or similar)

\*\*Arduino Code:\*\*

```cpp

char data = 0;

void setup() {

Serial.begin(9600);

pinMode(13, OUTPUT); // LED connected to pin 13

}

void loop() {

if (Serial.available()) {

data = Serial.read();

if (data == '1') digitalWrite(13, HIGH);

else if (data == '0') digitalWrite(13, LOW);

}

}

```

---

\*\*INTERNSHIP TASK -2: HOME AUTOMATION SYSTEM\*\*

\*\*Objective:\*\* Control lights and fans using IoT platforms like Blynk.

\*\*Components:\*\*

- NodeMCU ESP8266

- Relay Module

- Blynk App

\*\*Arduino Code (using Blynk):\*\*

```cpp

#define BLYNK\_PRINT Serial

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

char auth[] = "YourBlynkAuthToken";

char ssid[] = "YourSSID";

char pass[] = "YourPassword";

void setup() {

Serial.begin(9600);

Blynk.begin(auth, ssid, pass);

pinMode(D1, OUTPUT); // Light

pinMode(D2, OUTPUT); // Fan

}

void loop() {

Blynk.run();

}

BLYNK\_WRITE(V1) { digitalWrite(D1, param.asInt()); }

BLYNK\_WRITE(V2) { digitalWrite(D2, param.asInt()); }

```

---

\*\*INTERNSHIP TASK -3: IOT SECURITY SYSTEM\*\*

\*\*Objective:\*\* Detect motion, capture image, and send alert.

\*\*Components:\*\*

- NodeMCU ESP32-CAM

- PIR Motion Sensor

\*\*Code:\*\*

```cpp

#include "esp\_camera.h"

#include <WiFi.h>

const char\* ssid = "YourSSID";

const char\* password = "YourPassword";

void startCameraServer();

void setup() {

Serial.begin(115200);

camera\_config\_t config = {

.pin\_pwdn = 32,

.pin\_reset = -1,

.pin\_xclk = 0,

.pin\_sscb\_sda = 26,

.pin\_sscb\_scl = 27,

.pin\_d7 = 35,

.pin\_d6 = 34,

.pin\_d5 = 39,

.pin\_d4 = 36,

.pin\_d3 = 21,

.pin\_d2 = 19,

**✅ TASK 1: PUSH BUTTON COUNTER**

**🔌 Components:**

* Arduino Uno
* Push Button
* 10kΩ Resistor
* (Optional) 16x2 LCD Display

**💻 Code (Serial Monitor version):**

cpp

CopyEdit

int buttonPin = 2;

int counter = 0;

bool lastState = LOW;

void setup() {

pinMode(buttonPin, INPUT\_PULLUP);

Serial.begin(9600);

}

void loop() {

bool currentState = digitalRead(buttonPin);

if (lastState == HIGH && currentState == LOW) {

counter++;

Serial.println("Button pressed. Count = " + String(counter));

delay(200); // Debounce delay

}

lastState = currentState;

}

**✅ TASK 2: HOME AUTOMATION WITH BLUETOOTH**

**🔌 Components:**

* Arduino Uno
* HC-05 Bluetooth Module
* LED or Relay Module

**💻 Code:**

cpp

CopyEdit

char btInput;

int device1 = 7; // Connect to LED/relay

void setup() {

pinMode(device1, OUTPUT);

digitalWrite(device1, LOW);

Serial.begin(9600); // Communicates with HC-05

}

void loop() {

if (Serial.available()) {

btInput = Serial.read();

if (btInput == '1') {

digitalWrite(device1, HIGH); // Turn ON

Serial.println("Device ON");

} else if (btInput == '0') {

digitalWrite(device1, LOW); // Turn OFF

Serial.println("Device OFF");

}

}

}

📱 Use **Bluetooth Terminal App** to send commands:  
1 → Turn ON device  
0 → Turn OFF device

**✅ TASK 3: TEMPERATURE MONITORING SYSTEM**

**🔌 Components:**

* Arduino Uno
* LM35 Temperature Sensor
* Serial Monitor or LCD (optional)

**💻 Code (Serial Monitor):**

cpp

CopyEdit

int tempPin = A0;

void setup() {

Serial.begin(9600);

}

void loop() {

int reading = analogRead(tempPin);

float voltage = reading \* 5.0 / 1024.0;

float temperature = voltage \* 100; // LM35 gives 10mV per °C

Serial.print("Temperature: ");

Serial.print(temperature);

Serial.println(" °C");

delay(1000);

}

**✅ TASK 4: SPEECH RECOGNITION SYSTEM**

**🔌 Components:**

* Arduino Uno
* Voice Recognition Module (V3)
* Relay or LED (for output)

**💻 Sample Code (for trained voice commands):**

cpp

CopyEdit

#include <SoftwareSerial.h>

SoftwareSerial mySerial(2, 3); // RX, TX

int devicePin = 13; // Output pin

void setup() {

Serial.begin(9600);

mySerial.begin(9600);

pinMode(devicePin, OUTPUT);

}

void loop() {

if (mySerial.available()) {

int command = mySerial.read();

Serial.print("Command Received: ");

Serial.println(command);

if (command == 0x11) { // "Turn on light"

digitalWrite(devicePin, HIGH);

} else if (command == 0x12) { // "Turn off light"

digitalWrite(devicePin, LOW);

}

}

}

.

" JAVA.pdf"

\*\*INTERNSHIP TASK -1: SMART LIGHT CONTROL\*\*

\*\*Objective:\*\* Control an LED using Arduino and a mobile app.

\*\*Components:\*\*

- Arduino Uno

- LED

- Bluetooth Module (HC-05)

- Android App (built with MIT App Inventor or similar)

\*\*Arduino Code:\*\*

```cpp

char data = 0;

void setup() {

Serial.begin(9600);

pinMode(13, OUTPUT); // LED connected to pin 13

}

void loop() {

if (Serial.available()) {

data = Serial.read();

if (data == '1') digitalWrite(13, HIGH);

else if (data == '0') digitalWrite(13, LOW);

}

}

```

---

\*\*INTERNSHIP TASK -2: HOME AUTOMATION SYSTEM\*\*

\*\*Objecti

✅ Java code for all **Java Internship Tasks** has been added to your document:

* File Handling Utility
* REST API Client (with JSON parsing using org.json)
* Multithreaded Chat App (Sockets + Threads)
* AI Recommendation System (with Apache Mahout structure)

(Cloud )INTERNSHIP

---

## ☁️ Cloud Internship Tasks (CODTECH)

### ✅ Task 1: Cloud Storage Setup on AWS S3 / Google Cloud Storage

\*\*Instructions:\*\*

- Create and configure a cloud storage bucket.

- Upload sample files and set access permissions.

#### ✅ AWS S3 Example (Python with Boto3):

```python

import boto3

s3 = boto3.client('s3')

# Create a bucket

bucket\_name = "codtech-cloud-bucket-demo"

s3.create\_bucket(Bucket=bucket\_name)

# Upload a file

s3.upload\_file("sample.txt", bucket\_name, "sample.txt")

# Make file public

s3.put\_object\_acl(ACL='public-read', Bucket=bucket\_name, Key='sample.txt')

print("Bucket created and file uploaded!")

```

#### ✅ Google Cloud Storage Example (Python with `google-cloud-storage`):

```python

from google.cloud import storage

client = storage.Client()

bucket = client.create\_bucket("codtech-cloud-bucket-demo")

blob = bucket.blob("sample.txt")

blob.upload\_from\_filename("sample.txt")

blob.make\_public()

print("GCS bucket created and file uploaded!")

```

---

### ✅ Task 2: Cloud Monitoring and Alerts

\*\*Instructions:\*\*

- Set up monitoring for a cloud application using AWS CloudWatch / GCP Monitoring / Azure Monitor.

#### ✅ AWS CloudWatch Monitoring Example:

```bash

# Enable monitoring for EC2 instance:

aws cloudwatch put-metric-alarm \

--alarm-name "HighCPUUsage" \

--metric-name CPUUtilization \

--namespace AWS/EC2 \

--statistic Average \

--period 300 \

--threshold 70 \

--comparison-operator GreaterThanThreshold \

--dimensions Name=InstanceId,Value=i-0123456789abcdef0 \

--evaluation-periods 2 \

--alarm-actions arn:aws:sns:region:account-id:alarm-topic \

--unit Percent

```

#### ✅ Google Cloud Monitoring Example:

```yaml

# alert\_policy.yaml

displayName: "High CPU Usage"

conditions:

- conditionThreshold:

filter: metric.type="compute.googleapis.com/instance/cpu/utilization"

comparison: COMPARISON\_GT

thresholdValue: 0.8

duration: "300s"

combiner: "OR"

notificationChannels:

- "projects/PROJECT\_ID/notificationChannels/CHANNEL\_ID"

```

```bash

gcloud monitoring policies create --policy-from-file=alert\_policy.yaml

```

#### ✅ Dashboard:

Use the respective cloud dashboard (AWS CloudWatch or GCP Monitoring) to visualize metrics (CPU, memory, disk, etc.)

---

### ✅ Task 3: Multi-Cloud Architecture

\*\*Instructions:\*\*

- Design a system where services use multiple cloud providers (e.g., frontend on AWS, backend on GCP).

#### ✅ Example Architecture:

```

[User] ─▶ [CloudFront on AWS] ─▶ [Static Website on S3] ─▶ API calls ─▶ [Cloud Run (GCP)]

✅ All **three Cloud internship tasks** have been updated with code examples and setup instructions, including:

1. **Cloud Storage Setup** (AWS S3 & GCP Storage with Python scripts)
2. **Cloud Monitoring & Alerts** (AWS CloudWatch & GCP Monitoring with CLI/YAML)
3. **Multi-Cloud Architecture** (Interoperable setup across AWS and GCP with example fetch code)

" EMBEDDED SYSYTEMS.pdf"

\*\*INTERNSHIP TASK -1: TEMPERATURE SENSOR DISPLAY SYSTEM\*\*

\*\*Instructions:\*\* Use a temperature sensor to read and display temperature data on an LCD or serial monitor.

\*\*Deliverables:\*\* Circuit design, code, and output demo

\*\*Components:\*\*

- Arduino UNO

- DHT11 temperature sensor

- 16x2 LCD (optional) or Serial Monitor

\*\*Code:\*\*

```cpp

#include <DHT.h>

#define DHTPIN 2

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600);

dht.begin();

}

void loop() {

float temp = dht.readTemperature();

Serial.print("Temperature: ");

Serial.print(temp);

Serial.println(" °C");

delay(2000);

}

```

---

\*\*INTERNSHIP TASK -2: BLUETOOTH-CONTROLLED HOME AUTOMATION SYSTEM\*\*

\*\*Instructions:\*\* Design a Bluetooth-controlled home automation system to switch devices on and off.

\*\*Deliverables:\*\* Circuit diagram, code, and working demo

\*\*Components:\*\*

- Arduino UNO

- HC-05 Bluetooth module

- Relay module

- Android Bluetooth controller app

\*\*Code:\*\*

```cpp

char data = 0;

int relayPin = 7;

void setup() {

pinMode(relayPin, OUTPUT);

Serial.begin(9600);

}

void loop() {

if (Serial.available()) {

data = Serial.read();

if (data == '1') {

digitalWrite(relayPin, HIGH);

} else if (data == '0') {

digitalWrite(relayPin, LOW);

}

}

}

```

---

\*\*INTERNSHIP TASK -3: TEMPERATURE MONITORING SYSTEM (REPEAT)\*\*

\*Same as Task 1: Use DHT11 sensor and display temperature on LCD or Serial Monitor.\*

---

\*\*INTERNSHIP TASK -4: SPEECH RECOGNITION SYSTEM FOR DEVICE CONTROL\*\*

\*\*Instructions:\*\* Build a speech recognition system for command-based control using an embedded board.

\*\*Deliverables:\*\* System design, code, and demo

\*\*Components:\*\*

- Arduino UNO or ESP32

- Voice Recognition Module V3 (or similar)

- Relay Module

\*\*Code (Arduino with Voice Recognition Module):\*\*

```cpp

#include <SoftwareSerial.h>

SoftwareSerial mySerial(2, 3); // RX, TX

void setup() {

Serial.begin(9600);

mySerial.begin(9600);

pinMode(8, OUTPUT); // connected to relay

}

void loop() {

if (mySerial.available()) {

int command = mySerial.read();

if (command == 0x11) { // Assuming 0x11 maps to "Turn ON"

digitalWrite(8, HIGH);

} else if (command == 0x12) { // Assuming 0x12 maps to "Turn OFF"

digitalWrite(8, LOW);

}

}

}

```

" MACHINE LEARNING.pdf"

\*\*INTERNSHIP TASK -1: DECISION TREE MODEL WITH SCIKIT-LEARN\*\*

```python

import pandas as pd

from sklearn.datasets import load\_iris

from sklearn.tree import DecisionTreeClassifier, plot\_tree

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

# Load data

data = load\_iris()

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = data.target

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data.data, data.target, test\_size=0.3, random\_state=42)

# Train model

clf = DecisionTreeClassifier()

clf.fit(X\_train, y\_train)

# Predict and evaluate

y\_pred = clf.predict(X\_test)

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

# Visualize tree

plt.figure(figsize=(12, 8))

plot\_tree(clf, feature\_names=data.feature\_names, class\_names=data.target\_names, filled=True)

plt.show()

```

---

\*\*INTERNSHIP TASK -2: SENTIMENT ANALYSIS USING TF-IDF AND LOGISTIC REGRESSION\*\*

```python

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

# Load dataset

df = pd.read\_csv('https://raw.githubusercontent.com/dD2405/Twitter\_Sentiment\_Analysis/master/train.csv')

df = df[['tweet', 'label']].dropna()

# Preprocessing

X = df['tweet']

y = df['label']

# TF-IDF Vectorization

vectorizer = TfidfVectorizer(max\_fe

" MERN STACK.pdf"

// ✅ TASK 1: Real-Time Chat App using Socket.IO and React

// Backend (Node.js + Express + Socket.IO)

const express = require('express');

const http = require('http');

const socketIo = require('socket.io');

const cors = require('cors');

const app = express();

const server = http.createServer(app);

const io = socketIo(server, {

cors: { origin: '\*' }

});

io.on('connection', socket => {

console.log('User connected');

socket.on('chat message', msg => {

io.emit('chat message', msg);

});

socket.on('disconnect', () => {

console.log('User disconnected');

});

});

server.listen(5000, () => console.log('Server started on port 5000'));

// Frontend (React + Socket.IO-client)

import React, { useState, useEffect } from 'react';

import io from 'socket.io-client';

const socket = io('http://localhost:5000');

function ChatApp() {

const [message, setMessage] = useState('');

const [messages, setMessages] = useState([]);

useEffect(() => {

socket.on('chat message', msg => setMessages(prev => [...prev, msg]));

}, []);

const sendMessage = () => {

socket.emit('chat message', message);

setMessage('');

};

return (

<div>

<ul>{messages.map((m, i) => <li key={i}>{m}</li>)}</ul>

<input value={message} onChange={e => setMessage(e.target.value)} />

<button onClick={sendMessage}>Send</button>

</div>

);

}

// ✅ TASK 2: Weather App using OpenWeatherMap API

// Frontend (React)

import React, { useState } from 'react';

import axios from 'axios';

function WeatherApp() {

const [city, setCity] = useState('');

const [weather, setWeather] = useState(null);

const fetchWeather = async () => {

const res = await axios.get(`https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=YOUR\_API\_KEY&units=metric`);

setWeather(res.data);

};

return (

<div>

<input value={city} onChange={e => setCity(e.target.value)} placeholder="Enter city" />

<button onClick={fetchWeather}>Get Weather</button>

{weather && <div>{weather.name}: {weather.main.temp}°C</div>}

</div>

);

}

// ✅ TASK 3: Real-Time Collaboration Tool (Shared Whiteboard)

// Backend (Node.js + WebSocket)

const WebSocket = require('ws');

const wss = new WebSocket.Server({ port: 8080 });

wss.on('connection', socket => {

socket.on('message', message => {

wss.clients.forEach(client => {

if (client.readyState === WebSocket.OPEN) {

client.send(message);

}

});

});

});

// Frontend (React + Canvas)

import React, { useRef, useEffect } from 'react';

function Whiteboard() {

const canvasRef = useRef(null);

const ws = useRef(new WebSocket('ws://localhost:8080'));

useEffect(() => {

const canvas = canvasRef.current;

const ctx = canvas.getContext('2d');

ws.current.onmessage = e => {

const { x, y } = JSON.parse(e.data);

ctx.lineTo(x, y);

ctx.stroke();

};

canvas.addEventListener('mousemove', e => {

if (e.buttons === 1) {

const x = e.offsetX;

const y = e.offsetY;

ctx.lineTo(x, y);

ctx.stroke();

ws.current.send(JSON.stringify({ x, y }));

}

});

}, []);

return <canvas ref={canvasRef} width={600} height={400} style={{ border: '1px solid black' }} />;

}

// ✅ TASK 4: Productivity Tracker Chrome Extension

// manifest.json

{

"manifest\_version": 3,

"name": "Productivity Tracker",

"version": "1.0",

"permissions": ["tabs", "storage"],

"background": {

"service\_worker": "background.js"

},

"action": {

"default\_popup": "popup.html"

}

}

// background.js

chrome.tabs.onUpdated.addListener((tabId, changeInfo, tab) => {

if (changeInfo.status === 'complete') {

const domain = new URL(tab.url).hostname;

chrome.storage.local.get([domain], data => {

const time = (data[domain] || 0) + 1;

chrome.storage.local.set({ [domain]: time });

});

}

});

// popup.html

<!DOCTYPE html>

<html>

<body>

<h2>Productivity Report</h2>

<div id="report"></div>

<script src="popup.js"></script>

</body>

</html>

// popup.js

chrome.storage.local.get(null, data => {

const report = document.getElementById('report');

for

" NET.pdf"

### ✅ Task 1: ****Basic E-Commerce App****

* ASP.NET MVC + Entity Framework
* Models: Product, CartItem
* Controllers: ProductController
* Features: Listing, Add to Cart

### ✅ Task 2: ****Performance Monitoring Dashboard****

* Application Insights SDK setup in Startup.cs
* Dashboard configured via Azure Portal

### ✅ Task 3: ****Payment Integration****

* Stripe Checkout integration
* PaymentController handles session creation

### ✅ Task 4: ****AI Chatbot Integration****

* Azure Bot Service iframe embedded in Razor view
* Connected using botframework-webchat

// .NET INTERNSHIP TASKS - CODTECH

// TASK 1: Basic E-Commerce Application using ASP.NET MVC and Entity Framework

// Features: Product Listing, Shopping Cart, Order Placement

// Technologies: ASP.NET MVC, Entity Framework, SQL Server

// Models/Product.cs

public class Product {

public int Id { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

public string ImageUrl { get; set; }

}

// Models/CartItem.cs

public class CartItem {

public Product Product { get; set; }

public int Quantity { get; set; }

}

// Controllers/ProductController.cs

public class ProductController : Controller {

private readonly AppDbContext \_context;

public ProductController(AppDbContext context) => \_context = context;

public IActionResult Index() => View(\_context.Products.ToList());

public IActionResult AddToCart(int id) {

// Add product to session cart

}

}

// Views/Product/Index.cshtml

@foreach (var p in Model) {

<div>@p.Name - $@p.Price <a href="/Product/AddToCart/@p.Id">Add to Cart</a></div>

}

// TASK 2: Performance Monitoring Dashboard using Application Insights

// Steps:

// 1. Install Application Insights SDK

// 2. Configure instrumentation key in Startup.cs

// 3. Create dashboard with charts from logged telemetry data

// Startup.cs

services.AddApplicationInsightsTelemetry(Configuration["ApplicationInsights:InstrumentationKey"]);

// Use Azure Application Insights UI for dashboarding

// TASK 3: Payment Gateway Integration (Stripe)

// Features: Checkout with Stripe

// NuGet: Stripe.net

// Controllers/PaymentController.cs

public class PaymentController : Controller {

public IActionResult Checkout(decimal amount) {

var options = new SessionCreateOptions {

PaymentMethodTypes = new List<string> { "card" },

LineItems = new List<SessionLineItemOptions> {

new SessionLineItemOptions {

PriceData = new SessionLineItemPriceDataOptions {

Currency = "usd",

UnitAmount = (long)(amount \* 100),

ProductData = new SessionLineItemPriceDataProductDataOptions { Name = "Item" },

},

Quantity = 1,

},

},

Mode = "payment",

SuccessUrl = "https://yourdomain.com/success",

CancelUrl = "https://yourdomain.com/cancel",

};

var service = new SessionService();

Session session = service.Create(options);

return Redirect(session.Url);

}

}

// TASK 4: AI Chatbot using Azure Bot Service

// Steps:

// 1. Create bot in Azure Bot Service

// 2. Embed iframe or SDK in ASP.NET Core app

// 3. Integrate bot with QnA Maker or LUIS for AI

// Views/Home/Chat.cshtml

<iframe src="https://webchat.botframework.com/embed/yourbot?s=your\_secret\_key"

style="min-width: 400px; width: 100%; height: 500px;"></iframe>

// appsettings.json

"Bot": {

"Secret": "your\_secret\_key",

"EmbedUrl": "https://webchat.botframework.com/embed/yourbot"

}

.

" POWER BI.pdf"

**✅ Task 1: Power BI Sales Dashboard**

**Goal:** Show sales trends, top products, and regional performance.

**Steps:**

1. Load sample sales dataset (CSV/Excel).
2. Create visuals:
   * **Line Chart** for Sales Trend over Time.
   * **Bar/Column Chart** for Top Products.
   * **Map or Donut Chart** for Regional Performance.
3. Add **slicers** for filtering by product, region, or time.
4. Save file as SalesDashboard.pbix.

**Deliverable:** SalesDashboard.pbix file with clean layout and filters.

**✅ Task 2: Data Integration Report (Excel + SQL Server)**

**Goal:** Combine data from Excel and SQL Server to create unified insights.

**Steps:**

1. Load Excel file (e.g., product details) and connect to SQL Server (e.g., sales data).
2. Perform data transformation in **Power Query** (merge or append).
3. Model relationships in **Data View**.
4. Create visuals that use data from both sources.
   * Example: Sales table (SQL) with product categories (Excel).
5. Add filtering options and KPIs.

**Deliverable:** IntegratedReport.pbix showing combined source insights.

**✅ Task 3: Real-Time Streaming Dashboard**

**Goal:** Set up a live-updating Power BI dashboard using streaming data.

**Options:**

* **Azure Stream Analytics** with Power BI output.
* OR Simulate real-time feed with **Power BI REST API + Python script**.

**Steps:**

1. Go to Power BI Service → **My Workspace → Streaming datasets** → Add new.
2. Choose API → define schema → get Push URL.
3. Use a script (e.g., Python) to send data:

python

CopyEdit

import requests, json

url = "https://api.powerbi.com/beta/your\_push\_url"

data = {"time": "2025-07-18T08:00:00", "value": 98}

requests.post(url, headers={"Content-Type": "application/json"}, data=json.dumps([data]))

1. Create visuals (line chart, cards) in Power BI Service.

**Deliverable:** A Power BI dashboard linked to a live or simulated data stream.

**✅ Task 4: Use Python or R in Power BI**

**Goal:** Integrate Python or R for advanced analysis or visualization.

**Steps:**

1. Enable Python/R scripting in Power BI (Options → Python/R scripting).
2. In **Power Query Editor** → Transform → Run Python Script.
   * Example (Python):

python

CopyEdit

import pandas as pd

dataset['SalesGrowth'] = dataset['Sales'].pct\_change()

1. Use custom Python visuals:
   * Example: matplotlib or seaborn charts.
2. Clean formatting and ensure reproducibility.

**Deliverable:** A .pbix file using a Python or R script for transformation or custom visuals.

" PYTHON.pdf"

### ✅ ****INTERNSHIP TASK - 1: API INTEGRATION & DATA VISUALIZATION****

**Task:** Fetch data from a public API (e.g., OpenWeatherMap) and visualize using matplotlib or seaborn.

#### 💡 Suggested Project: Weather Dashboard

#### ✅ Deliverables:

* weather\_dashboard.py (Python script)
* dashboard.png (weather chart image)

#### ✅ Sample Code:

python

Copy code

import requests

import matplotlib.pyplot as plt

API\_KEY = "your\_openweathermap\_api\_key"

CITY = "Mumbai"

URL = f"http://api.openweathermap.org/data/2.5/forecast?q={CITY}&appid={API\_KEY}&units=metric"

response = requests.get(URL)

data = response.json()

dates = []

temps = []

for forecast in data["list"][:10]:

dates.append(forecast["dt\_txt"])

temps.append(forecast["main"]["temp"])

plt.figure(figsize=(10, 5))

plt.plot(dates, temps, marker='o', color='blue')

plt.title(f"Temperature Forecast for {CITY}")

plt.xlabel("Date/Time")

plt.ylabel("Temperature (°C)")

plt.xticks(rotation=45)

plt.tight\_layout()

plt.savefig("dashboard.png")

plt.show()

### ✅ ****INTERNSHIP TASK - 2: AUTOMATED REPORT GENERATION****

**Task:** Read data from a file, analyze it, and generate a formatted PDF using fpdf or reportlab.

#### 💡 Suggested Project: Sales Report Generator

#### ✅ Deliverables:

* generate\_report.py (Python script)
* report.pdf (sample report)

#### ✅ Sample Code (using fpdf):

python

Copy code

from fpdf import FPDF

import pandas as pd

data = pd.read\_csv("sales.csv") # Ensure this file exists

total\_sales = data["Amount"].sum()

pdf = FPDF()

pdf.add\_page()

pdf.set\_font("Arial", size=12)

pdf.cell(200, 10, txt="Sales Report", ln=True, align='C')

pdf.cell(200, 10, txt=f"Total Sales: ₹{total\_sales}", ln=True)

pdf.output("report.pdf")

### ✅ ****INTERNSHIP TASK - 3: AI CHATBOT WITH NLP****

**Task:** Build a chatbot using nltk or spacy.

#### 💡 Suggested Project: Rule-based FAQ Chatbot

#### ✅ Deliverables:

* chatbot.py (Python script)

#### ✅ Sample Code (using nltk):

python

Copy code

import nltk

from nltk.chat.util import Chat, reflections

pairs = [

[r"hi|hello", ["Hello! How can I help you?"]],

[r"what is your name?", ["I'm a simple chatbot created using Python."]],

[r"how are you?", ["I'm good, thanks! How can I assist you today?"]],

[r"bye", ["Goodbye! Have a nice day."]],

]

chatbot = Chat(pairs, reflections)

chatbot.converse()

### ✅ ****INTERNSHIP TASK - 4: MACHINE LEARNING MODEL IMPLEMENTATION****

**Task:** Create a predictive model using scikit-learn.

#### 💡 Suggested Project: Spam Email Detection

#### ✅ Deliverables:

* spam\_classifier.ipynb (Jupyter Notebook)

#### ✅ Sample Code:

python

Copy code

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score

df = pd.read\_csv("spam.csv", encoding='latin-1')[["v1", "v2"]]

df.columns = ["label", "message"]

df["label"] = df["label"].map({"ham": 0, "spam": 1})

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df["message"], df["label"], test\_size=0.2)

vectorizer = CountVectorizer()

X\_train\_vec = vectorizer.fit\_transform(X\_train)

X\_test\_vec = vectorizer.transform(X\_test)

model = MultinomialNB()

model.fit(X\_train\_vec, y\_train)

preds = model.predict(X\_test\_vec)

print(f"Accuracy: {accuracy\_score(y\_test, preds)}")

### 📝 Final Notes:

* Upload scripts and outputs (plots, PDFs, notebooks) to a **GitHub repository**.
* Comment your code clearly.
* Submit links and screenshots as per CodTech instructions.

" REACT.pdf"

**✅ INTERNSHIP TASK 1: Weather App with API Integration**

**🔧 Task:** Create a weather dashboard that fetches and displays weather data based on user input (e.g., city name) using a public API like OpenWeatherMap.

**💡 Key Features:**

* Input field for city name
* Fetch weather data (temperature, humidity, etc.)
* Display weather info in a clean UI
* Responsive design

**✅ Sample Setup:**

bash

Copy code

npx create-react-app weather-app

cd weather-app

npm install axios

**✅ Sample Code:**

jsx

Copy code

import React, { useState } from 'react';

import axios from 'axios';

const API\_KEY = 'your\_api\_key\_here';

function WeatherApp() {

const [city, setCity] = useState('');

const [data, setData] = useState(null);

const fetchWeather = async () => {

const res = await axios.get(

`https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${API\_KEY}&units=metric`

);

setData(res.data);

};

return (

<div className="p-4">

<input value={city} onChange={(e) => setCity(e.target.value)} placeholder="Enter city" />

<button onClick={fetchWeather}>Get Weather</button>

{data && (

<div>

<h2>{data.name}</h2>

<p>Temp: {data.main.temp}°C</p>

<p>Humidity: {data.main.humidity}%</p>

</div>

)}

</div>

);

}

export default WeatherApp;

**✅ INTERNSHIP TASK 2: Movie Search App (OMDB)**

**🔧 Task:** Create an app that fetches and displays movie data from [OMDB API](http://www.omdbapi.com/), with search and favorite-saving functionality.

**💡 Key Features:**

* Movie search by title
* Display poster, title, year, and rating
* Save favorites using localStorage
* Responsive design

**✅ Sample Setup:**

bash

Copy code

npx create-react-app movie-app

cd movie-app

npm install axios

**✅ Sample Code:**

jsx

Copy code

import React, { useState } from 'react';

import axios from 'axios';

const API\_KEY = 'your\_omdb\_api\_key';

function MovieApp() {

const [search, setSearch] = useState('');

const [movies, setMovies] = useState([]);

const searchMovies = async () => {

const res = await axios.get(`https://www.omdbapi.com/?s=${search}&apikey=${API\_KEY}`);

setMovies(res.data.Search || []);

};

return (

<div className="p-4">

<input value={search} onChange={(e) => setSearch(e.target.value)} placeholder="Search movie" />

<button onClick={searchMovies}>Search</button>

<div className="grid grid-cols-3 gap-4 mt-4">

{movies.map((movie) => (

<div key={movie.imdbID} className="border p-2">

<img src={movie.Poster} alt={movie.Title} className="w-full h-60 object-cover" />

<h3>{movie.Title}</h3>

<p>{movie.Year}</p>

</div>

))}

</div>

</div>

);

}

export default MovieApp;

**✅ INTERNSHIP TASK 3: Real-Time Chat App with Socket.IO**

**🔧 Task:** Build a chat app using React and Socket.IO with group chat and typing indicators.

**💡 Key Features:**

* Real-time messaging
* Group chat
* Typing indicator
* Socket.IO backend

**✅ Backend (Node.js + Express + Socket.IO):**

bash

Copy code

npm init -y

npm install express socket.io

js

Copy code

// server.js

const express = require('express');

const http = require('http');

const { Server } = require('socket.io');

const app = express();

const server = http.createServer(app);

const io = new Server(server);

io.on('connection', (socket) => {

socket.on('message', (data) => {

socket.broadcast.emit('message', data);

});

socket.on('typing', () => {

socket.broadcast.emit('typing');

});

});

server.listen(5000, () => console.log('Server running on port 5000'));

**✅ Frontend (React + Socket.IO Client):**

bash

Copy code

npm install socket.io-client

jsx

Copy code

import React, { useEffect, useState } from 'react';

import io from 'socket.io-client';

const socket = io("http://localhost:5000");

function ChatApp() {

const [msg, setMsg] = useState('');

const [messages, setMessages] = useState([]);

const [typing, setTyping] = useState(false);

useEffect(() => {

socket.on('message', (msg) => setMessages((prev) => [...prev, msg]));

socket.on('typing', () => setTyping(true));

const timeout = setTimeout(() => setTyping(false), 2000);

return () => clearTimeout(timeout);

}, []);

const sendMessage = () => {

socket.emit('message', msg);

setMessages([...messages, msg]);

setMsg('');

};

const handleTyping = () => socket.emit('typing');

return (

<div>

<h2>Chat Room</h2>

{typing && <p>Someone is typing...</p>}

<div>

{messages.map((m, i) => <p key={i}>{m}</p>)}

</div>

<input value={msg} onChange={(e) => setMsg(e.target.value)} onKeyPress={handleTyping} />

<button onClick={sendMessage}>Send</button>

</div>

);

}

export default ChatApp;

**✅ INTERNSHIP TASK 4: Chrome Extension for Productivity Tracker**

**🔧 Task:** Build a Chrome extension with a React UI that:

* Sets daily goals
* Tracks website usage
* Visualizes productivity trends

**💡 Key Components:**

* manifest.json
* Background script to monitor tabs
* React popup UI
* Local storage for goal tracking

**✅ Basic manifest.json:**

json

Copy code

{

"manifest\_version": 3,

"name": "Productivity Tracker",

"version": "1.0",

"permissions": ["tabs", "storage", "activeTab", "scripting"],

"background": {

"service\_worker": "background.js"

},

"action": {

"default\_popup": "index.html",

"default\_icon": "icon.png"

}

}

**✅ React Build Instructions:**

1. Create React app
2. Build with npm run build
3. Copy contents of /build to the extension folder
4. Add manifest.json and load unpacked extension via chrome://extensions

**📝 Submission Checklist:**

* Host or zip your projects
* Document usage and screenshots in a README.md
* Submit GitHub repo link and output files (e.g., screenshots or video demo)

" SOFTWARE DEVELOPMENT.pdf"

**✅ INTERNSHIP TASK 1: Static Developer Portfolio Website**

**Task:** Build a responsive static website using HTML, CSS, and JavaScript to showcase skills and projects.

**💡 Features:**

* Home/About section
* Skills list with progress bars or icons
* Projects section with images and links
* Contact form or social links
* Responsive layout (mobile-friendly)

**✅ Basic File Structure:**

pgsql

Copy code

portfolio/

├─ index.html

├─ style.css

└─ script.js

**✅ Sample index.html snippet:**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>Developer Portfolio</title>

<link rel="stylesheet" href="style.css" />

</head>

<body>

<header>

<h1>Bhagyashree Patil</h1>

<nav>

<a href="#skills">Skills</a>

<a href="#projects">Projects</a>

<a href="#contact">Contact</a>

</nav>

</header>

<section id="skills">

<h2>Skills</h2>

<ul>

<li>JavaScript</li>

<li>React.js</li>

<li>Python</li>

<li>REST APIs</li>

</ul>

</section>

<section id="projects">

<h2>Projects</h2>

<div class="project">

<h3>Weather App</h3>

<p>A React app that fetches weather data.</p>

<a href="https://github.com/yourrepo/weather-app" target="\_blank">View Code</a>

</div>

</section>

<section id="contact">

<h2>Contact Me</h2>

<p>Email: your.email@example.com</p>

</section>

</body>

</html>

**✅ INTERNSHIP TASK 2: RESTful API for Library or Inventory System**

**Task:** Design and implement CRUD endpoints.

**💡 Tech stack suggestion:**

* **Backend:** Node.js + Express.js
* **Database:** MongoDB or SQLite/PostgreSQL

**✅ Sample CRUD endpoints for books (Express.js):**

js

Copy code

const express = require('express');

const app = express();

app.use(express.json());

let books = []; // In-memory storage for demo

// CREATE

app.post('/books', (req, res) => {

const book = {...req.body, id: Date.now()};

books.push(book);

res.status(201).json(book);

});

// READ all

app.get('/books', (req, res) => {

res.json(books);

});

// READ one

app.get('/books/:id', (req, res) => {

const book = books.find(b => b.id == req.params.id);

if (!book) return res.status(404).send('Book not found');

res.json(book);

});

// UPDATE

app.put('/books/:id', (req, res) => {

let index = books.findIndex(b => b.id == req.params.id);

if (index === -1) return res.status(404).send('Book not found');

books[index] = {...books[index], ...req.body};

res.json(books[index]);

});

// DELETE

app.delete('/books/:id', (req, res) => {

books = books.filter(b => b.id != req.params.id);

res.status(204).send();

});

app.listen(3000, () => console.log('API running on port 3000'));

**✅ INTERNSHIP TASK 3: Real-time Collaborative Coding or Note-taking Tool**

**Task:** Create a multi-user collaboration tool with real-time updates using WebSocket.

**💡 Approach:**

* Backend: Node.js + ws or Socket.IO
* Frontend: React with WebSocket connection

**✅ Key points:**

* Real-time sync of text editor content
* Broadcast changes to all connected clients
* Handle cursor positions and typing status (optional)

**✅ Simple backend with WebSocket (Node.js):**

js

Copy code

const WebSocket = require('ws');

const wss = new WebSocket.Server({ port: 8080 });

let content = "";

wss.on('connection', (ws) => {

ws.send(JSON.stringify({ type: 'init', content }));

ws.on('message', (message) => {

const data = JSON.parse(message);

if(data.type === 'update') {

content = data.content;

wss.clients.forEach(client => {

if(client !== ws && client.readyState === WebSocket.OPEN) {

client.send(JSON.stringify({ type: 'update', content }));

}

});

}

});

});

**✅ Frontend React snippet to connect WebSocket and sync:**

jsx

Copy code

import React, { useEffect, useState } from 'react';

function CollaborativeEditor() {

const [content, setContent] = useState("");

let ws;

useEffect(() => {

ws = new WebSocket('ws://localhost:8080');

ws.onmessage = (event) => {

const data = JSON.parse(event.data);

if(data.type === 'init' || data.type === 'update') {

setContent(data.content);

}

};

}, []);

const handleChange = (e) => {

setContent(e.target.value);

ws.send(JSON.stringify({ type: 'update', content: e.target.value }));

};

return <textarea value={content} onChange={handleChange} rows={20} cols={80} />;

}

export default CollaborativeEditor;

**✅ INTERNSHIP TASK 4: Refactor Open-Source Project**

**Task:** Choose an open-source repo, improve code readability, performance, and document changes.

**💡 Tips:**

* Select a small to medium-sized project (GitHub search)
* Focus on:
  + Code formatting & commenting
  + Simplifying complex functions
  + Removing duplicate code
* 

Improving algorithm efficiency

* Updating dependencies or libraries if needed

 Document before/after results:

* Lines of code changed
* Performance benchmarks (if applicable)
* Readability improvements
* Testing to ensure no breaking changes

" SOFTWARE TESTING.pdf"

**✅ INTERNSHIP TASK 1: Selenium WebDriver Test Automation**

**Task:** Automate login and navigation tests for a sample web app.

**🔧 Tools:**

* Selenium WebDriver (Python/Java)
* Browser driver (e.g., ChromeDriver)
* Test runner (unittest, pytest, or JUnit)

**✅ Sample Python Selenium script:**

python

Copy code

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

import time

import unittest

class LoginTest(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Chrome() # Make sure chromedriver is in PATH

self.driver.get("https://example.com/login") # Replace with actual URL

def test\_login\_and\_navigation(self):

driver = self.driver

driver.find\_element(By.ID, "username").send\_keys("testuser")

driver.find\_element(By.ID, "password").send\_keys("password123")

driver.find\_element(By.ID, "loginBtn").click()

time.sleep(3) # Wait for login

# Verify login by checking dashboard presence

dashboard = driver.find\_element(By.ID, "dashboard")

self.assertTrue(dashboard.is\_displayed())

# Navigate to profile page

driver.find\_element(By.LINK\_TEXT, "Profile").click()

time.sleep(2)

profile\_header = driver.find\_element(By.TAG\_NAME, "h1")

self.assertEqual(profile\_header.text, "Your Profile")

def tearDown(self):

self.driver.quit()

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

unittest.main()

**Deliverables:**

* Selenium scripts (.py or .java)
* Test execution report (e.g., pytest HTML report or console logs)

**✅ INTERNSHIP TASK 2: API Testing Using Postman**

**Task:** Test RESTful APIs for authentication, data retrieval, etc.

**🔧 Tools:**

* Postman desktop app
* Postman Collections (JSON export)
* Environment variables in Postman

**Steps:**

1. Import or create requests for:
   * Login/authentication API (POST)
   * Get user data or other resource retrieval (GET)
2. Add tests in Postman (JavaScript snippets) to validate response codes, JSON schema, or values.
3. Run Collection Runner or Postman Monitor to execute tests.
4. Export collection and generate test result documentation (screenshots, exported run results).

**Example Postman test snippet for login:**

js

Copy code

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

pm.test("Response has token", function () {

var jsonData = pm.response.json();

pm.expect(jsonData).to.have.property("token");

});

**✅ INTERNSHIP TASK 3: Security Testing (SQL Injection, XSS)**

**Task:** Identify vulnerabilities in a sample web app and suggest mitigations.

**🔧 Tools:**

* Manual testing with browser + proxy tools (e.g., Burp Suite Community)
* Automated scanners (OWASP ZAP)
* SQL Injection payloads & XSS test scripts

**Approach:**

* Test all user input fields with typical SQL injection payloads like ' OR 1=1--
* Test XSS by injecting scripts like <script>alert('XSS')</script>
* Document vulnerability found, affected URL/parameter, severity, and remediation suggestions.

**Sample vulnerability report snippet:**

| **Vulnerability** | **URL** | **Parameter** | **Severity** | **Description** | **Mitigation** |
| --- | --- | --- | --- | --- | --- |
| SQL Injection | /login | username | High | User input is concatenated into SQL without sanitization. | Use parameterized queries / prepared statements. |
| XSS | /comments | comment | Medium | Input not sanitized, allows script injection. | Escape HTML, implement CSP. |

**✅ INTERNSHIP TASK 4: Load Testing with Gatling**

**Task:** Perform load testing simulating many users and analyze performance.

**🔧 Tools:**

* Gatling (Scala-based load testing)
* JDK + Gatling installed

**Sample Gatling test script (Scala):**

scala

Copy code

import io.gatling.core.Predef.\_

import io.gatling.http.Predef.\_

import scala.concurrent.duration.\_

class BasicLoadTest extends Simulation {

val httpProtocol = http.baseUrl("https://example.com")

val scn = scenario("Basic Load Test")

.exec(http("Load Homepage")

.get("/"))

.pause(1)

.exec(http("Login")

.post("/login")

.formParam("username", "testuser")

.formParam("password", "password123"))

.pause(1)

.exec(http("Dashboard")

.get("/dashboard"))

setUp(

scn.inject(

rampUsers(100) during (30.seconds)

).protocols(httpProtocol)

)

}

" SQL.pdf"

## ✅ INTERNSHIP TASK 1: Practice Different SQL JOINS

Assuming two example tables:  
**Employees** (emp\_id, name, dept\_id)  
**Departments** (dept\_id, dept\_name)

### Sample Data:

sql

Copy code

Employees:

emp\_id | name | dept\_id

-------|----------|---------

1 | Alice | 10

2 | Bob | 20

3 | Charlie | NULL

Departments:

dept\_id | dept\_name

--------|-----------

10 | HR

20 | Sales

30 | Marketing

### Sample Queries:

**1. INNER JOIN** — returns matching rows only

sql

Copy code

SELECT e.name, d.dept\_name

FROM Employees e

INNER JOIN Departments d ON e.dept\_id = d.dept\_id;

**2. LEFT JOIN** — all employees, departments if matched, else NULL

sql

Copy code

SELECT e.name, d.dept\_name

FROM Employees e

LEFT JOIN Departments d ON e.dept\_id = d.dept\_id;

**3. RIGHT JOIN** — all departments, employees if matched, else NULL

sql

Copy code

SELECT e.name, d.dept\_name

FROM Employees e

RIGHT JOIN Departments d ON e.dept\_id = d.dept\_id;

**4. FULL OUTER JOIN** — all employees and all departments, matched or not

sql

Copy code

SELECT e.name, d.dept\_name

" UIUX.pdf"

**✅ INTERNSHIP TASK 1: Redesign an Existing Website Landing Page**

**What to do:**

* Choose a website landing page (e.g., a company homepage or product page)
* Identify usability issues (confusing navigation, clutter, bad contrast)
* Improve aesthetics (modern fonts, colors, spacing, images)
* Create **before-and-after visuals** showing redesign

**Tools:**

* Design tools like Figma, Adobe XD, or Canva
* Screenshots for “before”
* New design mockup/prototype for “after”

**Deliverable:**

* Side-by-side comparison images/screenshots
* A brief document explaining design choices (e.g., why changed colors, layout, button placement)

**✅ INTERNSHIP TASK 2: Responsive Webpage Design**

**What to do:**

* Build a webpage that adapts smoothly to various screen sizes (mobile, tablet, desktop)
* Use **media queries** in CSS
* Optionally use flexible layouts (Flexbox, Grid)
* Test responsiveness by resizing browser or using device simulators

**Example CSS snippet:**

css

Copy code

.container {

display: flex;

flex-wrap: wrap;

}

.item {

flex: 1 1 300px;

margin: 10px;

}

@media (max-width: 600px) {

.item {

flex-basis: 100%;

}

}

**Deliverable:**

* HTML, CSS, JS files of the responsive page
* Screenshots or video showing responsiveness on different devices

**✅ INTERNSHIP TASK 3: Mobile App UI/UX Redesign for a Pain Point**

**What to do:**

* Pick a popular mobile app (e.g., Instagram, Uber, WhatsApp)
* Identify one pain point (confusing UI, hard navigation, poor onboarding)
* Redesign that screen/flow to fix the problem
* Provide wireframes or high-fidelity mockups

**Deliverable:**

* Prototype or mockup files (Figma/Adobe XD)
* Write-up explaining:
  + The pain point
  + Your redesign solution
  + How it improves user experience

**✅ INTERNSHIP TASK 4: AR/VR Interface Design**

**What to do:**

* Conceptualize a simple AR/VR interface for an app (e.g., virtual showroom, training app)
* Focus on **intuitive interactions**: gestures, voice commands, gaze control
* Design screens or interaction flow mockups

**Tools:**

* 3D prototyping (optional) — Unity UI mockups, Figma with 3D plugins, or simple 2D mockups
* Storyboards or flow diagrams

**Deliverable:**

* Mockups or prototypes demonstrating UI elements and user interactions
* Explanation document describing:
  + Interaction design decisions
  + How users will intuitively navigate or control the app

" VLSI.pdf"

**✅ INTERNSHIP TASK 1: Design a Basic ALU (Addition, Subtraction, AND, OR, AND NOT)**

**Features:**

* Inputs: Two 8-bit operands (A, B)
* Control signals to select operation:
  + 00 = ADD
  + 01 = SUB
  + 10 = AND
  + 11 = OR
* Output: 8-bit result and carry/borrow flag

**Sample Verilog Code (Basic ALU):**

verilog

Copy code

module ALU(

input [7:0] A, B,

input [1:0] Op, // operation select

output reg [7:0] Result,

output reg CarryFlag

);

always @(\*) begin

case(Op)

2'b00: {CarryFlag, Result} = A + B; // ADD

2'b01: {CarryFlag, Result} = A - B; // SUB

2'b10: Result = A & B; // AND

2'b11: Result = A | B; // OR

default: Result = 8'b0;

endcase

end

endmodule

**✅ INTERNSHIP TASK 2: Simple Synchronous RAM with Read/Write**

**Features:**

* Inputs: clk, we (write enable), addr, data\_in
* Output: data\_out
* RAM size: 256 x 8-bit

**Sample Verilog Code:**

verilog

Copy code

module SyncRAM(

input clk,

input we,

input [7:0] addr,

input [7:0] data\_in,

output reg [7:0] data\_out

);

reg [7:0] ram [0:255];

always @(posedge clk) begin

if (we)

ram[addr] <= data\_in;

data\_out <= ram[addr];

end

endmodule

**✅ INTERNSHIP TASK 3: 4-Stage Pipelined Processor (ADD, SUB, LOAD)**

**Pipeline stages:**

* IF (Instruction Fetch)
* ID (Instruction Decode)
* EX (Execute)
* WB (Write Back)

**Notes:**

* Design instruction register and control signals
* Handle data forwarding or stalls if needed (basic design can skip hazards)
* Support instructions: ADD, SUB, LOAD

Due to complexity, focus on:

* Define pipeline registers between stages
* Simulate instruction flow per clock cycle

**✅ INTERNSHIP TASK 4: Digital FIR Filter Design (Verilog or MATLAB)**

**Task:**

* Design FIR filter with predefined coefficients
* Implement convolution operation on input samples

**Basic Verilog FIR Filter outline:**

verilog

Copy code

module FIR\_Filter(

input clk,

input reset,

input signed [15:0] sample\_in,

output reg signed [31:0] filtered\_out

);

parameter N = 4; // Number of taps

reg signed [15:0] shift\_reg [0:N-1];

wire signed [31:0] coeffs [0:N-1];

assign coeffs[0] = 16'sd1;

assign coeffs[1] = 16'sd2;

assign coeffs[2] = 16'sd3;

assign coeffs[3] = 16'sd4;

integer i;

always @(posedge clk or posedge reset) begin

if (reset) begin

filtered\_out <= 0;

for(i=0; i<N; i=i+1)

shift\_reg[i] <= 0;

end else begin

// Shift input samples

for(i=N-1; i>0; i=i-1)

shift\_reg[i] <= shift\_reg[i-1];

shift\_reg[0] <= sample\_in;

// Multiply and accumulate

filtered\_out <= 0;

for(i=0; i<N; i=i+1)

filtered\_out <= filtered\_out + shift\_reg[i]\*coeffs[i];

end

end

endmodule

**Simulation & Testbench Tips:**

* Write testbenches applying stimulus vectors for each module.
* Use ModelSim/Quartus/Xilinx tools for compiling and waveform analysis.
* Check all functional outputs and timing behavior.

" WEB DEVELOPMENT.pdf"

**✅ TASK 1: Personal Portfolio Website with Modern CSS Animations (GSAP/Three.js)**

**What to do:**

* Build a personal portfolio website (HTML/CSS/JS)
* Use GSAP for smooth animations (scrolling effects, fade-ins, etc.)
* Optionally use Three.js for 3D interactive effects

**Example: Basic GSAP animation for a header fade-in**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Portfolio</title>

<script src="https://cdnjs.cloudflare.com/ajax/libs/gsap/3.12.2/gsap.min.js"></script>

<style>

body { font-family: Arial, sans-serif; }

h1 { opacity: 0; }

</style>

</head>

<body>

<h1>Welcome to My Portfolio</h1>

<script>

gsap.to("h1", { duration: 2, opacity: 1, y: 20 });

</script>

</body>

</html>

**✅ TASK 2: Online Learning Platform (Video, Quizzes, Progress Tracking)**

**Suggested stack:**

* Frontend: React or vanilla JS
* Backend: Node.js (Express) or Django
* Database: MongoDB/PostgreSQL
* Video: Use HTML5 <video> tag or integrate with video hosting APIs
* Authentication: JWT or session-based
* Progress: Store in DB user quiz results and video watched percentage

**Example: Simple Express route serving a video URL**

js

Copy code

const express = require('express');

const app = express();

app.get('/video/:id', (req, res) => {

const videoId = req.params.id;

// Serve video URL from DB or static folder

res.sendFile(\_\_dirname + `/videos/${videoId}.mp4`);

});

app.listen(3000, () => console.log("Server running on port 3000"));

**✅ TASK 3: Real-time Collaborative Document Editor with React & Backend**

**Suggested stack:**

* Frontend: React + WebSocket (Socket.IO)
* Backend: Node.js + Socket.IO
* DB: MongoDB or PostgreSQL to save documents

**Example: React snippet for WebSocket connection**

jsx

Copy code

import React, { useEffect, useState } from "react";

import io from "socket.io-client";

const socket = io("http://localhost:4000");

function CollaborativeEditor() {

const [content, setContent] = useState("");

useEffect(() => {

socket.on("update", (data) => setContent(data));

}, []);

const handleChange = (e) => {

setContent(e.target.value);

socket.emit("update", e.target.value);

};

return <textarea value={content} onChange={handleChange} rows="20" cols="80" />;

}

export default CollaborativeEditor;

**✅ TASK 4: PWA for E-commerce with Offline Support & Push Notifications**

**Key features:**

* Service Worker registration for offline caching
* Cache API to store static assets & API responses
* Push notifications via Push API

**Example: Basic service worker registration and cache setup**

**service-worker.js:**

js

Copy code

const CACHE\_NAME = "ecommerce-cache-v1";

const urlsToCache = ["/", "/index.html", "/styles.css", "/app.js"];

self.addEventListener("install", (event) => {

event.waitUntil(

caches.open(CACHE\_NAME).then((cache) => cache.addAll(urlsToCache))

);

});

self.addEventListener("fetch", (event) => {

event.respondWith(

caches.match(event.request).then((response) => response || fetch(event.request))

);

});

**In your main JS file (e.g., app.js):**

js

Copy code

if ("serviceWorker" in navigator) {

navigator.serviceWorker.register("/service-worker.js")

.then(() => console.log("Service Worker registered"))

.catch((err) => console.log("Service Worker registration failed:", err));

}

**Summary**

* Task 1: Build portfolio, animate with GSAP or 3D with Three.js
* Task 2: Video streaming, quizzes & auth backend API
* Task 3: Collaborative React editor using Socket.IO
* Task 4: PWA with offline caching & push notifications