MAPÚA MALAYAN DIGITAL COLLEGE

A college under Mapúa Malayan Colleges Laguna

Intellectual Property Notice

This template is an exclusive property of **Mapua-Malayan Digital College** and is protected under **Republic Act No. 8293**, also known as the *Intellectual Property Code of the Philippines* (IP Code). It is provided solely for educational purposes within this course. Students may use this template to complete their tasks but may not **modify, distribute, sell, upload,** or **claim ownership** of the template itself. Such actions constitute copyright infringement under **Sections 172, 177, and 216** of the IP Code and may result in legal consequences. Unauthorized use beyond this course may result in legal or academic consequences.

Additionally, students must comply with the **Mapua-Malayan Digital College Student Handbook**, particularly with the following provisions:

- **Offenses Related to MMDC IT**:
  - **Section 6.2** – Unauthorized copying of files
  - **Section 6.8** – Extraction of protected, copyrighted, and/or confidential information by electronic means using MMDC IT infrastructure
- **Offenses Related to MMDC Admin, IT, and Operations**:
  - **Section 4.5** – Unauthorized collection or extraction of money, checks, or other instruments of monetary equivalent in connection with matters pertaining to MMDC

Violations of these policies may result in **disciplinary actions ranging from suspension to dismissal**, in accordance with the Student Handbook.

For permissions or inquiries, please contact MMDC-ISD at isd@mmdc.mcl.edu.ph.

MAPÚA MALAYAN DIGITAL COLLEGE

A college under Mapúa Malayan Colleges Laguna

Week 2 Template

Write a **Python script** to generate your IoT data. Below is a general template. Modify it depending on your chosen industry and data.

```
import pandas as pd
import numpy as np
from datetime import datetime, timedelta

num_records = 100  # Adjust this number as needed

# Example for Smart Healthcare Monitoring
data = []

for _ in range(num_records):
    record = {
        "timestamp": datetime.now() -
timedelta(minutes=np.random.randint(0, 1440)),  # Random timestamp
in the last 24 hours
        "patient_id": f"PAT{np.random.randint(100, 999)}",  # Random
patient ID
        "heart_rate": np.random.randint(60, 100),  # Normal heart
rate range
        "blood_pressure": f"{np.random.randint(100,
140)}/{np.random.randint(60, 90)}",  # Systolic/Diastolic range
        "oxygen_level": np.random.randint(95, 100),  # Oxygen
saturation in normal range
        "body_temp": round(np.random.uniform(36.0, 38.0), 1)  # Body
temperature in Celsius
    }
    data.append(record)
```

```
# Convert to DataFrame
df = pd.DataFrame(data)

# Save dataset
df.to_csv("healthcare_data.csv", index=False)
df.to_json("healthcare_data.json", orient="records")

# Display first few rows
df.head()
```

MAPÚA MALAYAN DIGITAL COLLEGE

A college under Mapúa Malayan Colleges Laguna

Week 3 Template

Smart Contract Guide

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract IoTDataStorage {
    struct IoTData {
        uint256 timestamp;
        string deviceId;
        string dataType;
        string dataValue;
    }

    uint256 public constant MAX_ENTRIES = 100;
    IoTData[] public dataRecords;
    address public owner;

    event DataStored(uint256 timestamp, string deviceId, string
dataType, string dataValue);

    modifier onlyOwner() {
        require(msg.sender == owner, "Not authorized");
        _;
    }

    constructor() {
        owner = msg.sender;
    }
```

```
    function storeData(string memory _deviceId, string memory
_dataType, string memory _dataValue) public onlyOwner {
        require(dataRecords.length < MAX_ENTRIES, "Storage limit
reached");
        dataRecords.push(IoTData(block.timestamp, _deviceId,
_dataType, _dataValue));
        emit DataStored(block.timestamp, _deviceId, _dataType,
_dataValue);
    }

    function getTotalRecords() public view returns (uint256) {
        return dataRecords.length;
    }

    function getRecord(uint256 index) public view returns (uint256,
string memory, string memory, string memory) {
        require(index < dataRecords.length, "Index out of bounds");
        IoTData memory record = dataRecords[index];
        return (record.timestamp, record.deviceId, record.dataType,
record.dataValue);
    }
}
```

MAPÚA MALAYAN DIGITAL COLLEGE

A college under Mapúa Malayan Colleges Laguna

Week 4 - Milestone 1: Smart Tracking System Blockchain Ledger (Draft)

1. Open **Jupyter Notebook**.
2. Verify that **Ganache is running** in the background. Open the app on your desktop to ensure it's running.
3. Write a script that establishes a **Web3 connection** to the **Ganache RPC URL** (http://127.0.0.1:7545). **Note**: Check the port number in the Ganache settings, whether it is 7545 or 8545, and replace it accordingly.
4. Check if Python is **successfully connected** to the blockchain:

```
from web3 import Web3

# Connect to local Ganache blockchain
ganache_url = "http://127.0.0.1:7545"
web3 = Web3(Web3.HTTPProvider(ganache_url))

if web3.is_connected():
    print("✅ Connected to Ganache successfully!")
else:
    print("❌ Connection failed. Ensure Ganache is running.")
```

5. Retrieve and paste the **contract address** and **ABI** from Remix IDE. You can also retrieve these from your Homework 2.
6. Ensure the smart contract is **recognized and loaded** in Python:

```
# Replace with actual contract address from Remix
contract_address = "0xYourRemixDeployedContractAddress"

# Paste the ABI from Remix
abi = [...]  # Replace with your contract ABI

# Load the smart contract
```

```
contract = web3.eth.contract(address=contract_address, abi=abi)

# Set the default sender address (first account from Ganache)
web3.eth.default_account = web3.eth.accounts[0]

print(f"✅ Connected to Smart Contract at {contract_address}")
```

7. Call **`getTotalRecords()`** to check if the contract is responding.
8. Store a **dummy IoT data entry** manually to test if transactions work:

```
txn = contract.functions.storeData("TEST001", "Temperature",
"22.5°C").transact({
    'from': web3.eth.default_account,
    'gas': 1000000
})
web3.eth.wait_for_transaction_receipt(txn)
print("✅ Dummy data stored on blockchain!")
```

9. Verify if data retrieval works:

```
total_records = contract.functions.getTotalRecords().call()
print(f"Total Records: {total_records}")

record = contract.functions.getRecord(0).call()
print("First Stored Record:", record)
```

10. Upload this to your GitHub repository.
    a. **Save your document** in an appropriate format (Python script `.py`, Jupyter Notebook `.ipynb`, or Markdown `.md`).
    b. **Navigate to your GitHub repository** where your project files are stored.
    c. Click **"Add file"** → **"Upload files"**, then select your document.
    d. Add a **commit message** (e.g., "Added transaction verification script").
    e. Click **"Commit changes"** to save your work to the repository.

MAPÚA MALAYAN DIGITAL COLLEGE

A college under Mapúa Malayan Colleges Laguna

Week 5 - Milestone 1: Smart Tracking System Blockchain Ledger (Submission)

1. Open your Python script and load the CSV file from your Homework 1:

```python
import pandas as pd

# Load IoT sensor data from CSV (Generated in Homework 1)
df = pd.read_csv("formatted_healthcare_data.csv")

# Display the first few rows
print(df.head())
```

2. Here is the expected outcome:

```
    timestamp  device_id      data_type  data_value
0  1707408200     PAT001     Heart Rate       75 BPM
1  1707408260     PAT002  Oxygen Level         98%
2  1707408320     PAT003   Temperature       36.5°C
```

3. Connect Python to the smart contract.
    a. Ensure Ganache is running in the background.
    b. Establish connection with Web3.py:

```python
from web3 import Web3



# Connect to local blockchain

ganache_url = "http://127.0.0.1:7545"

web3 = Web3(Web3.HTTPProvider(ganache_url))
```

```
# Verify connection
if web3.is_connected():
    print("✅ Connected to Ganache successfully!")
else:
    print("❌ Connection failed. Ensure Ganache is running.")
```

c. Load the smart contract:

```
# Replace with actual contract address from Remix
contract_address = "0xYourRemixDeployedContractAddress"

# Paste the ABI from Remix
abi = [...]  # Replace with your contract ABI

# Load the smart contract
contract = web3.eth.contract(address=contract_address,
abi=abi)

# Set default sender (first account from Ganache)
web3.eth.default_account = web3.eth.accounts[0]
```

```
print(f"✅ Connected to Smart Contract at {contract_address}")
```

4. Send the IoT data to the blockchain. Each row from the CSV file will be stored as a transaction on the blockchain.
    a. Define a function to store IoT data:

```
import time



def send_iot_data(device_id, data_type, data_value):

    """Sends IoT data to the deployed smart contract"""

    txn = contract.functions.storeData(device_id, data_type,
data_value).transact({

        'from': web3.eth.default_account,

        'gas': 3000000

    })



    receipt = web3.eth.wait_for_transaction_receipt(txn)

    print(f"✅ Data Stored: {data_type} - {data_value}, Txn
Hash: {receipt.transactionHash.hex()}")
```

    b. Loop through the CSV file and send each row to the blockchain:

```
for _, row in df.iterrows():

    send_iot_data(str(row["device_id"]),
str(row["data_type"]), str(row["data_value"]))

    time.sleep(1)  # Delay to prevent flooding transactions
```

5. Now that the data is **on the blockchain, retrieve** it to verify storage.
    a. Get total stored records:

```
total_records = contract.functions.getTotalRecords().call()

print(f"Total IoT records stored: {total_records}")
```

    b. Retrieve and print a specific record

```
record = contract.functions.getRecord(0).call()

print("First Stored Record:", record)
```

MAPÚA MALAYAN DIGITAL COLLEGE

A college under Mapúa Malayan Colleges Laguna

Week 6: Data Retrieval and Processing

1. Retrieve your Milestone 1 output to start the retrieval process.
2. Get the total number of stored records:

```
total_records = contract.functions.getTotalRecords().call()
print(f"Total IoT records stored: {total_records}")
```

3. Fetch all stored IoT data and structure it in a DataFrame:

```
import pandas as pd

# Retrieve all IoT records
data = []
for i in range(total_records):
    record = contract.functions.getRecord(i).call()
    data.append({
        "timestamp": record[0],
        "device_id": record[1],
        "data_type": record[2],
        "data_value": record[3]
    })

# Convert to a DataFrame
df = pd.DataFrame(data)

# Convert timestamp to readable format
df["timestamp"] = pd.to_datetime(df["timestamp"], unit="s")

# Display first few records
print(df.head())
```

4. After the data is structured and **cleaned, it is preprocessed** for further analysis. Convert numerical values where applicable:
    a. Some IoT sensor readings may contain units or text (e.g., "22.5°C", "50% humidity", "15.3 kWh"). You need to extract the numerical values to make the data usable.
    b. Identify missing values.
        i. If missing values are **minor**, replace them with `0`.
        ii. If missing values are significant, use the mean or median of the column.

```
import numpy as np

# Extract numeric values from 'data_value' where applicable
df["numeric_value"] =
df["data_value"].str.extract(r'(\d+\.?\d*)').astype(float)

# Handle missing values (if any)
df.fillna(0, inplace=True)

# Display cleaned data
print(df.head())
```

5. This is how your expected output should look like:

	timestamp	device_id	data_type	data_value	numeric_value
0	2024-02-10 14:30	PAT001	Heart Rate	75 BPM	75.0
1	2024-02-10 14:35	PAT002	Oxygen Level	98%	98.0
2	2024-02-10 14:40	PAT003	Temperature	36.5°C	36.5

6. Save the DataFrame as a CSV file:

```
# Save cleaned IoT data to a CSV file
df.to_csv("cleaned_iot_data.csv", index=False)

print("✅ Cleaned IoT data saved successfully as
cleaned_iot_data.csv")
```

7. Upload this to your GitHub repository.

a. **Save your document** in an appropriate format (Python script `.py`, Jupyter Notebook `.ipynb`, or Markdown `.md`).
b. **Navigate to your GitHub repository** where your project files are stored.
c. Click **"Add file"** → **"Upload files"**, then select your document.
d. Add a **commit message** (e.g., "Added transaction verification script").
e. Click **"Commit changes"** to save your work to the repository.

MAPÚA MALAYAN DIGITAL COLLEGE

A college under Mapúa Malayan Colleges Laguna

Week 7: Line Plot of IoT Sensor Readings Over Time

1. Access your preprocessed data from Homework No. 3.

```
import pandas as pd

# Load cleaned IoT data
df = pd.read_csv("cleaned_iot_data.csv")

# Display first few rows to verify data
print(df.head())
```

2. To ensure **proper plotting**, convert the timestamp column into a **datetime format**.

```
# Convert timestamp column to datetime
df["timestamp"] = pd.to_datetime(df["timestamp"])
```

3. Import the required libraries:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

4. Set the visualization style.

```
sns.set(style="whitegrid")
```

5. Create the line plot.

```
plt.figure(figsize=(12, 6))  # Adjust figure size
sns.lineplot(x=df["timestamp"], y=df["numeric_value"],
hue=df["data_type"], marker="o")

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Add title and labels
```

```
plt.title("IoT Sensor Readings Over Time", fontsize=14)
plt.xlabel("Timestamp", fontsize=12)
plt.ylabel("Sensor Value", fontsize=12)

# Show legend
plt.legend(title="Sensor Type")

# Display the plot
plt.show()
```