import time  
import random  
import csv  
from datetime import datetime  
import matplotlib.pyplot as plt  
from matplotlib.animation import FuncAnimation  
  
# Simulation settings  
LOW\_WATER\_LEVEL = 20 # Minimum safe water level (in percentage)  
HIGH\_WATER\_LEVEL = 80 # Maximum safe water level (in percentage)  
MAX\_TEMPERATURE = 100 # Maximum safe temperature (in Celsius)  
  
# Initialize parameters  
water\_level = random.randint(LOW\_WATER\_LEVEL, HIGH\_WATER\_LEVEL) # Initial water level (%)  
temperature = random.randint(50, 70) # Initial temperature (°C)  
flow\_rate = random.uniform(2.0, 5.0) # Initial flow rate (liters/minute)  
  
# CSV file setup for data logging  
log\_file = 'boiler\_tank\_data.csv'  
with open(log\_file, 'w', newline='') as file:  
 writer = csv.writer(file)  
 writer.writerow(["Timestamp", "Water Level (%)", "Temperature (°C)", "Flow Rate (L/min)", "Alert"])  
  
# Data storage for visualization  
time\_data = []  
water\_level\_data = []  
temperature\_data = []  
flow\_rate\_data = []  
alert\_data = []  
  
# Simulation function  
def simulate\_tank():  
 global water\_level, temperature, flow\_rate  
 alert = ""  
  
 # Update parameters  
 water\_level += random.randint(-3, 3) # Random fluctuation in water level  
 water\_level = max(0, min(100, water\_level)) # Clamp to [0, 100]  
  
 temperature += random.uniform(-0.5, 0.5) # Random fluctuation in temperature  
 flow\_rate += random.uniform(-0.1, 0.1) # Random fluctuation in flow rate  
  
 # Check for alerts  
 if water\_level < LOW\_WATER\_LEVEL:  
 alert = "LOW WATER LEVEL!"  
 elif water\_level > HIGH\_WATER\_LEVEL:  
 alert = "HIGH WATER LEVEL!"  
 if temperature > MAX\_TEMPERATURE:  
 alert += " HIGH TEMPERATURE!"  
  
 # Log data  
 timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  
 with open(log\_file, 'a', newline='') as file:  
 writer = csv.writer(file)  
 writer.writerow([timestamp, water\_level, temperature, flow\_rate, alert])  
  
 # Store data for visualization  
 time\_data.append(datetime.now())  
 water\_level\_data.append(water\_level)  
 temperature\_data.append(temperature)  
 flow\_rate\_data.append(flow\_rate)  
 alert\_data.append(alert)  
  
 return timestamp, water\_level, temperature, flow\_rate, alert  
  
# Real-time visualization  
def update\_graph(frame):  
 simulate\_tank()  
  
 # Clear the current graph  
 plt.cla()  
  
 # Plot water level  
 plt.subplot(3, 1, 1)  
 plt.plot(time\_data, water\_level\_data, label='Water Level (%)', color='blue')  
 plt.axhline(LOW\_WATER\_LEVEL, color='red', linestyle='--', label='Low Level')  
 plt.axhline(HIGH\_WATER\_LEVEL, color='orange', linestyle='--', label='High Level')  
 plt.legend(loc='upper left')  
 plt.title("Water Level Monitoring")  
 plt.ylabel("Water Level (%)")  
  
 # Plot temperature  
 plt.subplot(3, 1, 2)  
 plt.plot(time\_data, temperature\_data, label='Temperature (°C)', color='green')  
 plt.axhline(MAX\_TEMPERATURE, color='red', linestyle='--', label='Max Temperature')  
 plt.legend(loc='upper left')  
 plt.title("Temperature Monitoring")  
 plt.ylabel("Temperature (°C)")  
  
 # Plot flow rate  
 plt.subplot(3, 1, 3)  
 plt.plot(time\_data, flow\_rate\_data, label='Flow Rate (L/min)', color='purple')  
 plt.legend(loc='upper left')  
 plt.title("Flow Rate Monitoring")  
 plt.ylabel("Flow Rate (L/min)")  
 plt.xlabel("Time")  
  
 plt.tight\_layout()  
  
# Set up the visualization  
fig = plt.figure(figsize=(10, 8))  
ani = FuncAnimation(fig, update\_graph, interval=1000) # Update every second  
  
# Run the simulation and visualization  
print("Starting Boiler Feed Water Tank Monitoring System...")  
plt.show()