pragma solidity ^0.8.0;

contract SmartMeterContract {

    struct MeterData {

        uint256 timestamp;

        string hexData;  // Single string field to store meter data in hexadecimal format

    }

    struct Location {

        string latitude;

        string longitude;

    }

    string private password;

    mapping(address => MeterData) public meterData;

    mapping(address => Location) public meterLocations;

    event NewMeterData(

        address indexed meterAddress,

        uint256 timestamp,

        string hexData,

        string latitude,

        string longitude

    );

    // Function to set the password

    function setPassword(string memory \_password) public {

        password = \_password;

    }

    // Function to record meter data with one hexData string

    function recordMeterData(

        string memory \_password,

        string memory \_hexData,       // Single string to store all meter data in hex format

        string memory \_latitude,

        string memory \_longitude

    ) public {

        require(

            keccak256(abi.encodePacked(\_password)) == keccak256(abi.encodePacked(password)),

            "Invalid password"

        );

        MeterData memory newData = MeterData({

            timestamp: block.timestamp,

            hexData: \_hexData          // Store the hex data as a single string

        });

        meterData[msg.sender] = newData;

        meterLocations[msg.sender] = Location(\_latitude, \_longitude);

        emit NewMeterData(

            msg.sender,

            newData.timestamp,

            newData.hexData,

            \_latitude,

            \_longitude

        );

    }

    // Function to get meter data

    function getMeterData(address \_meterAddress) public view returns (

        uint256 timestamp,

        string memory hexData,

        string memory latitude,

        string memory longitude

    ) {

        MeterData memory meter = meterData[\_meterAddress];

        Location memory location = meterLocations[\_meterAddress];

        return (

            meter.timestamp,

            meter.hexData,

            location.latitude,

            location.longitude

        );

    }

}