EAST WEST UNIVERSITY
Department of Computer Science and Engineering
B.Sc. in Computer Science and Engineering Program
Midterm Assessment Project, Summer 2025 Semester

Course:	CSE407 Green Computing, Section: 01
Student Name:	Sudipta Roy
Student ID:	2021-2-60-009

Project Title: IoT based real-time energy monitoring and dashboard development.

Checklists:

- I used IoT technology to monitor the energy consumption of my PC for 3 hours.
- I have prepared an energy monitoring dashboard with detailed information.
- I have considered the financial and business aspects.
- I did PESTEL analysis on the project.

Declaration:

I declare that the project was done by myself and the submitted information is free from any unfair means. I complied with all the relevant legal and organizational requirements.

Sudipta Roy

Signature: _______________

Name: Sudipta Roy

Date: 03-08-2025

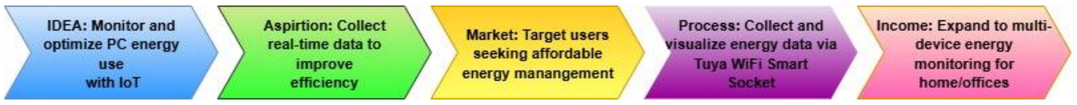
Executive Summary:

This initiative monitors the energy use of a personal computer (PC) using IoT technology. A Tuya WiFi Smart Socket/Plug (CH126 UK Variant) measured and recorded the PC's energy consumption over a 3-hour period. The system collects data like voltage, current, and wattage. This information appears on a web dashboard created with Flask and Chart.js. The recorded data is saved in a JSON file and evaluated to determine energy consumption in kilowatt-hours (kWh). The project aims to offer insights into both real-time and historical energy use, improving energy efficiency management.

Brief Description of the Work:

The goal of this project is to track the energy usage of my computer using an IoT enabled Tuya WiFi Smart Socket and show the information on a live dashboard. The Tuya WiFi Smart Socket is connected to the computer, and it collects data every minute for 3 hours. Flask manages the backend. It pulls data from the plug and saves it in a JSON file. The information is then displayed using Chart.js in a simple HTML interface. This interface includes real-time charts for voltage, current, and power usage.

Flowchart of the Work:


IDEA: Monitor and optimize PC energy use with IoT
Aspirtion: Collect real-time data to improve efficiency
Market: Target users seeking affordable energy manangement
Process: Collect and visualize energy data via Tuya WiFi Smart Socket
Income: Expand to multi-device energy monitoring for home/offices

Detailed Description of Each Step:

1. Planning:

   The project started with exploring the available energy monitoring options. After looking at the different options, the Tuya WiFi Smart Socket (CH126 UK Variant) was chosen; it could be used in conjunction with IoT platforms and is compatible with Python.

   The project goal was to measure the energy use and visualize that data for the PC.

2. Researching:

   I studied the Tuya API and found that it can provide real time data that includes voltage, current and wattage.

   I confirmed the Tuya WiFi Smart Socket could meet the power requirements needed by my PC, as well that I could access the data from the cloud.
3. Purchase:

   I purchased the Tuya WiFi Smart Socket as required.

   The smart plug was connected to the PC to make sure the energy readings were accurate.
4. Setup:

   I setup and connected the Tuya WiFi Smart Socket to the Tuya app and recovered the API keys for use.

   Flask was used to set up the backend to pull data from the Tuya cloud and show it on the web dashboard.
5. Data Collection:

   I created a Python script to gather data from the Tuya WiFi Smart Socket with a frequency of 8 second intervals.

   Measurements were voltage, current and wattage which were recorded in a JSON file and later analyzed.
6. Data Processing and Analysis:

   The data I collected was analyzed with the aim of determining energy consumption in kilowatt-hours (kWh).

   This involved determining the time between data samples, and determining the average wattage in each section of time.
7. Data Visualization:

   The data was visualized on a real-time web dashboard, using Chart.js to depict the energy usage over the 12-hour monitoring period.

   The dashboard could create time-series graphs for voltage, current, and power.

Challenges and Hiccups:

- CORS Issues: I encountered issues with the CORS policy while trying to fetch data from the local JSON file. This was resolved by setting up

Flask to serve the JSON data, which allowed the dashboard to fetch and display the data properly.

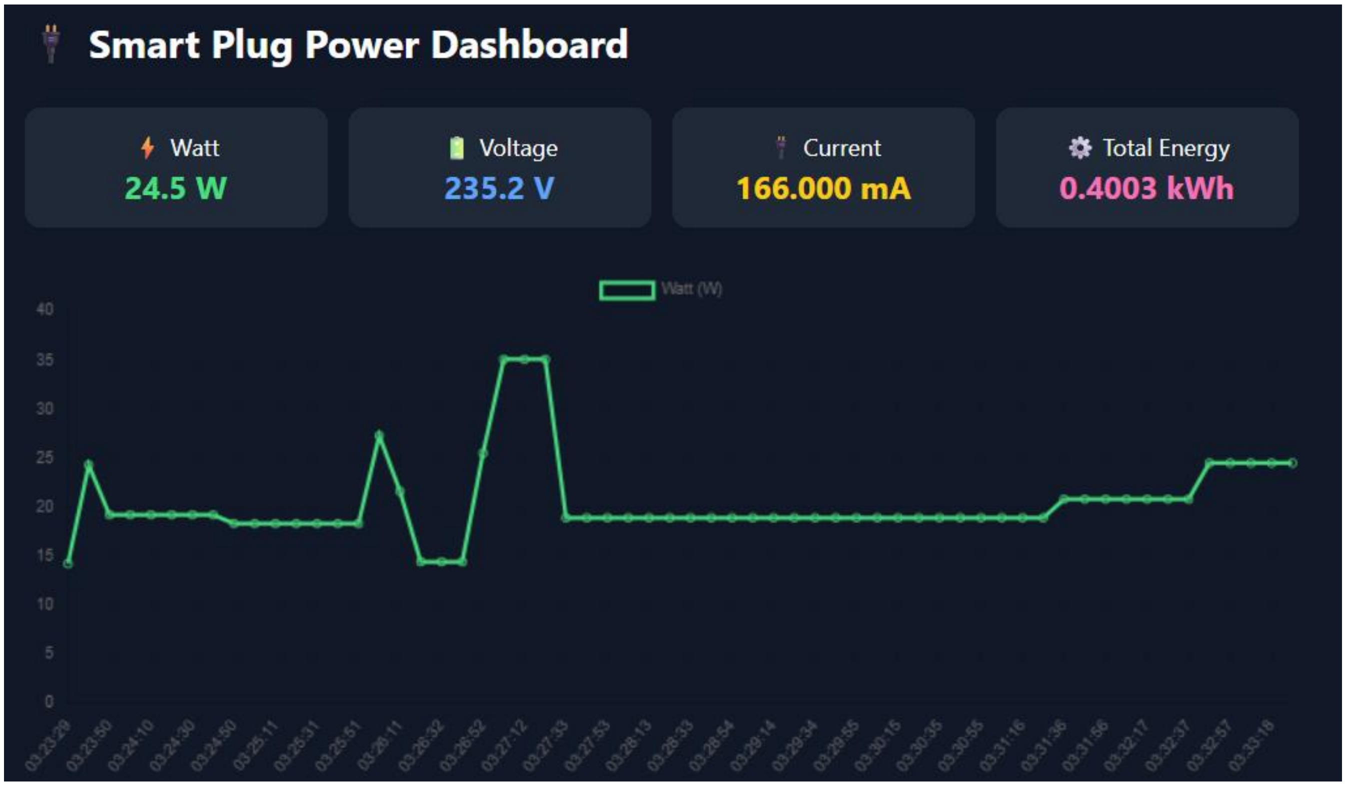
- API Authentications Issues: API authentication with Tuya initially failed due to expired API keys. After regenerating the API keys and updating them in the backend code, the problem was solved.

Demonstration:

- Link to Energy Dashboard:

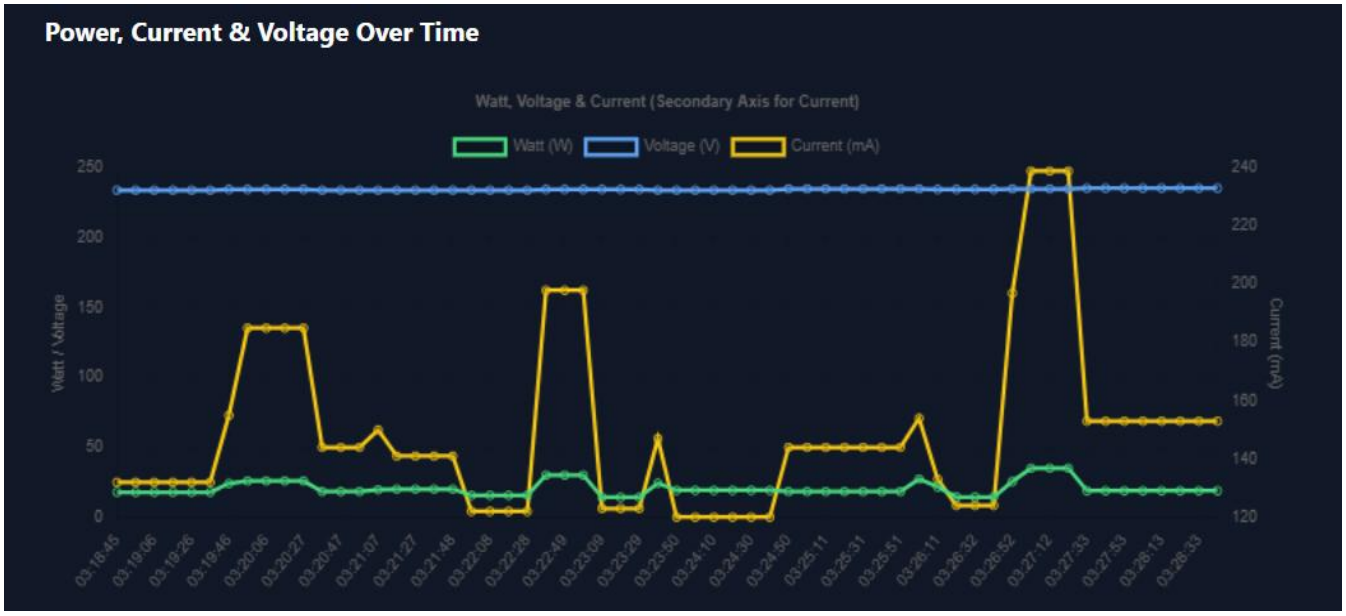
(https://github.com/SudiptaRoyJoy/cse407mid)

- Pictures:


Smart Plug Power Dashboard

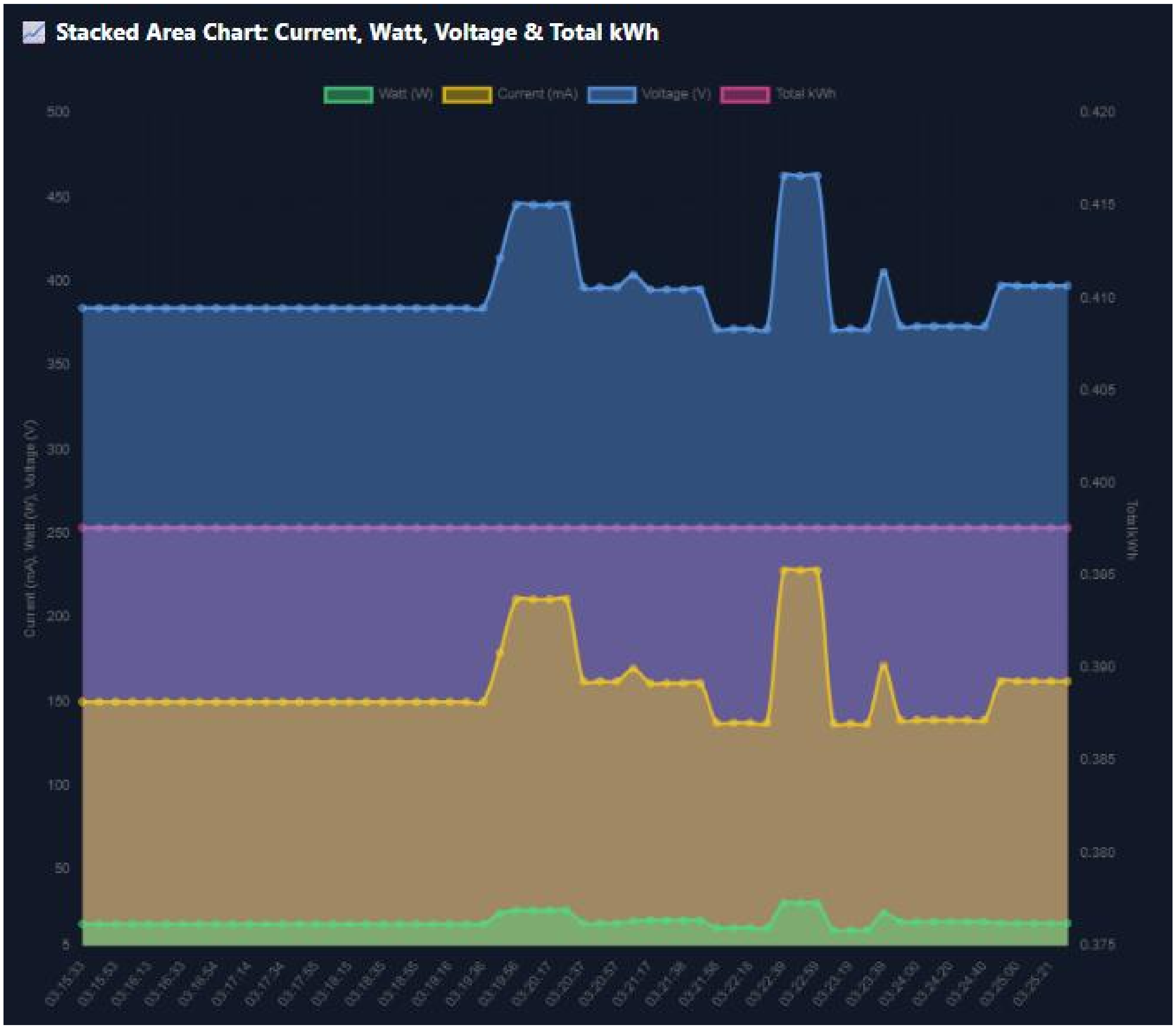
Watt	Voltage	Current	Total Energy
24.5 W	235.2 V	166.000 mA	0.4003 kWh

Picture 1



Power, Current & Voltage Over Time
Watt, Voltage & Current (Secondary Axis for Current)
Watt (W)
Voltage (V)
Current (mA)
Watt / Voltage
Current (mA)
250
200
150
100
50
0
240
220
200
180
160
140
120
03:18:45
03:19:06
03:19:26
03:19:46
03:20:06
03:20:27
03:20:47
03:21:07
03:21:27
03:21:48
03:22:08
03:22:28
03:22:49
03:23:09
03:23:29
03:23:50
03:24:10
03:24:30
03:24:50
03:25:11
03:25:31
03:25:51
03:26:11
03:26:32
03:26:52
03:27:12
03:27:33
03:27:53
03:28:13
03:28:33

Picture :02


Stacked Area Chart: Current, Watt, Voltage & Total kWh
Watt (W)
Current (mA)
Voltage (V)
Total kWh
Current (mA), Watt (W), Voltage (V)
Total kWh

PICTURE:03

Discussion on Issues:

Checklist of Issues	Remarks (if any)
*Planning and researching?	Thorough research on the Tuya WiFi Smart Socket and its compatibility with IoT systems
Data Collected?	Data collected every 10 seconds for voltage, current, and wattage.
Data Stored?	Data stored in a JSON file for analysis.
Data Displayed in the dashboard?	Real-time data displayed using Chart.js in a web dashboard.
*Realtime or stored data?	Real-time data collected and processed.
*Wattage of the chosen device?	Correct wattage calculated based on the PC's energy requirements.
*Wattage of the measuring equipment?	Tuya WiFi Smart Socket rated for the necessary wattage.
*AC Power: Why/not apparent power? Why/not Instantaneous Power? AC power vs. DC power?	Used real-time wattage (apparent power) instead of instantaneous power. AC power was relevant for the device.
*Documentation	Proper documentation of the setup and code for reproducibility.
*Safety: Electrical Insulation and Isolation?	Ensured the Tuya plug met safety standards for insulation and isolation.
*Caution with overclocking/flushing	No overclocking involved; PC settings were kept standard.
*API Issues: Did you get it? How? If not, how did you solve this problem?	API keys obtained successfully after regeneration, solving the authentication issue.

*UI/UX issues? Standard components of an energy dashboard?	Clean, functional dashboard with minimal UI issues.
*User Manual?	Basic guide provided for system setup and dashboard usage.
*Future Extensions and Limitations?	Future extensions could include multi-device monitoring. Limitation - only one device monitored at a time.
*Installation, Operation and Maintenance?	Easy installation; regular monitoring needed. Maintenance required if API keys expire.
*Recurring costs	No recurring costs except electricity consumption.
*Cost Accounting?	Initial setup costs considered (Tuya WiFi Smart Socket purchase).
*Business Aspects? Cost savings and ROI? Value of this product/service? Justification?	Potential for energy savings by optimizing energy use; ROI can be seen in long-term electricity bills.
*Reliability? Never failed? Any fail-safe mechanisms?	Reliable in testing, no failures encountered, no automatic fail-safe built into the system.
*Accuracy? Calibration?	Accuracy within device tolerance; no calibration done but consistent readings observed.
*Data quality? Sampling rate? Crosstalk and interference? Accuracy and calibration?	High-quality data with minimal errors. Data sampled every 10 seconds to ensure tracking of energy trends.
*Scalability?	Scalable to monitor multiple devices with slight adjustments in the code.

*Interoperability?	Interoperable with other IoT devices and platforms like Tuya API.
*Data Security? Important or not in this case?	Data security not a primary concern as no personal data is being collected, only energy consumption data.
*Compliant with regulations?	No specific regulations were applicable, but standard safety practices followed.
*Environmental Impacts? PESTLE analysis?	Conducted a PESTLE analysis covering political, economic, social, technological, legal, and environmental aspects.

Appendices:

- Code: (https://github.com/SudiptaRoyJoy/cse407mid)

```
from flask import Flask, jsonify, render_template
from flask_sqlalchemy import SQLAlchemy
import tinytuya
import threading
import time
from datetime import datetime
from sqlalchemy.sql import func
from sqlalchemy import text

from flask import send_file
import pandas as pd
from sqlalchemy import text
import io

app = Flask(__name__)

# Tuya Device Info
DEVICE_ID = "bf035aef5b8c5240dbykne"
LOCAL_KEY = "}=rHhdU-JWFeL3CB"
DEVICE_IP = "192.168.10.171"
PROTOCOL_VERSION = "3.5"
```

```
device = tinytuya.OutletDevice(DEVICE_ID, DEVICE_IP, LOCAL_KEY)
device.set_version(float(PROTOCOL_VERSION))

# SQLite Config
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///energy_data.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

# Database Model
class EnergyData(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    timestamp = db.Column(db.String(20))
    watt = db.Column(db.Float)
    voltage = db.Column(db.Float)
    current = db.Column(db.Float)
    kwh = db.Column(db.Float)

with app.app_context():
    db.create_all()

# Polling Function (safe with app context)
def poll_device(interval=10):
    while True:
        try:
            data = device.status()
            dp = data.get("dps", {})

            raw_watt = dp.get("19", 0)
            voltage = dp.get("20", 0)
            current = dp.get("18", 0)

            # Corrected watt value
            watt = raw_watt / 10

            timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
            kwh = watt * (interval / 3600) / 1000  # convert to kWh

            with app.app_context():
                entry = EnergyData(
                    timestamp=timestamp,
                    watt=watt,
                    voltage=voltage,
                    current=current,
                    kwh=kwh
                )
                db.session.add(entry)
                db.session.commit()

            print(f"[{timestamp}] W: {watt}W, V: {voltage}V, A: {current}mA, kWh: {kwh:.6f}")
        except Exception as e:
            print("Error fetching data:", e)
        time.sleep(interval)

# Flag to avoid multiple thread start
polling_started = False

# Routes
@app.route('/')
def dashboard():
    global polling_started
    if not polling_started:
        threading.Thread(target=poll_device, daemon=True).start()
        polling_started = True
    return render_template('dashboard.html')

@app.route('/api/data')
def get_data():
    entries = EnergyData.query.order_by(EnergyData.id.desc()).limit(60).all()
    entries.reverse()
    return jsonify([
        {
        "timestamp": e.timestamp,
        "watt": e.watt,
        "voltage": e.voltage,
        "current": e.current
        } for e in entries
    ])

@app.route('/api/total-kwh')
def total_kwh():
    total = db.session.query(db.func.sum(EnergyData.kwh)).scalar() or 0
    return jsonify({"total_kwh": round(total, 4)})
```

```
@app.route('/api/stats')
def energy_stats():
  daily = db.session.execute(
    """
    SELECT SUBSTR(timestamp, 1, 10) AS day, SUM(kwh)
    FROM energy_data
    GROUP BY day
    ORDER BY day DESC
    LIMIT 7
    """
  ).fetchall()

  hourly = db.session.execute(
    """
    SELECT SUBSTR(timestamp, 1, 13) AS hour, SUM(kwh)
    FROM energy_data
    GROUP BY hour
    ORDER BY hour DESC
    LIMIT 24
    """
  ).fetchall()

  return jsonify({
    "daily": [{"day": d[0], "kwh": round(d[1], 4)} for d in daily],
    "hourly": [{"hour": h[0], "kwh": round(h[1], 4)} for h in hourly]
  })

@app.route('/api/stats/minutely')
def minutely_stats():
  # Group by minute (first 16 chars of timestamp: 'YYYY-MM-DD HH:MM')
  results = db.session.execute(text("""
  SELECT SUBSTR(timestamp, 1, 16) AS minute, SUM(kwh) AS total_kwh
  FROM energy_data
  GROUP BY minute
  ORDER BY minute DESC
  LIMIT 60
  """)).fetchall()

  # Reverse so oldest to newest
  results = list(reversed(results))
```

```
    return jsonify([
        {"minute": r[0], "total_kwh": round(r[1], 6)} for r in results
    ])

@app.route('/export/full-energy-report')
def export_full_energy_report():
    # Fetch full raw energy data
    raw_data = db.session.execute(text("""
        SELECT timestamp, watt, current, voltage, kwh
        FROM energy_data
        ORDER BY timestamp ASC
    """)).fetchall()

    # Scale voltage for the export
    scaled_data = []
    for row in raw_data:
        timestamp, watt, current, voltage, kwh = row
        voltage = voltage / 10  # scale down voltage
        scaled_data.append((timestamp, watt, current, voltage, kwh))

    df_raw = pd.DataFrame(scaled_data, columns=['Timestamp', 'Watt', 'Current', 'Voltage', 'kWh'])
    df_raw['kWh'] = df_raw['kWh'].round(6)

    # Group by each minute for minutely total kWh
    df_raw['Minute'] = df_raw['Timestamp'].astype(str).str.slice(0, 16)
    df_minutely = df_raw.groupby('Minute', as_index=False)['kWh'].sum()
    df_minutely.rename(columns={'kWh': 'Total_kWh'}, inplace=True)
    df_minutely['Total_kWh'] = df_minutely['Total_kWh'].round(6)

    # Write both sheets into one Excel file
    output = io.BytesIO()
    with pd.ExcelWriter(output, engine='openpyxl') as writer:
        df_raw.drop(columns='Minute').to_excel(writer, index=False, sheet_name='Raw Data')
        df_minutely.to_excel(writer, index=False, sheet_name='Minutely Report')

    output.seek(0)

    return send_file(output,
                     as_attachment=True,
                     download_name='full_energy_report.xlsx',
                mimetype='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet')

#http://localhost:5000/export/full-energy-report
@app.route('/api/graph-data')
def api_graph_data():
    results = db.session.execute(text("""
        SELECT timestamp, current, watt, voltage
        FROM energy_data
        ORDER BY timestamp DESC
        LIMIT 100
    """)).fetchall()

    data = [{
        'timestamp': row.timestamp.strftime('%H:%M:%S'),
        'current': round(row.current, 2),
        'watt': round(row.watt, 2),
        'voltage': round(row.voltage, 2)
    } for row in results][::-1]  # Reverse to get chronological order

    return jsonify(data)

if __name__ == '__main__':
    app.run(debug=True)
```

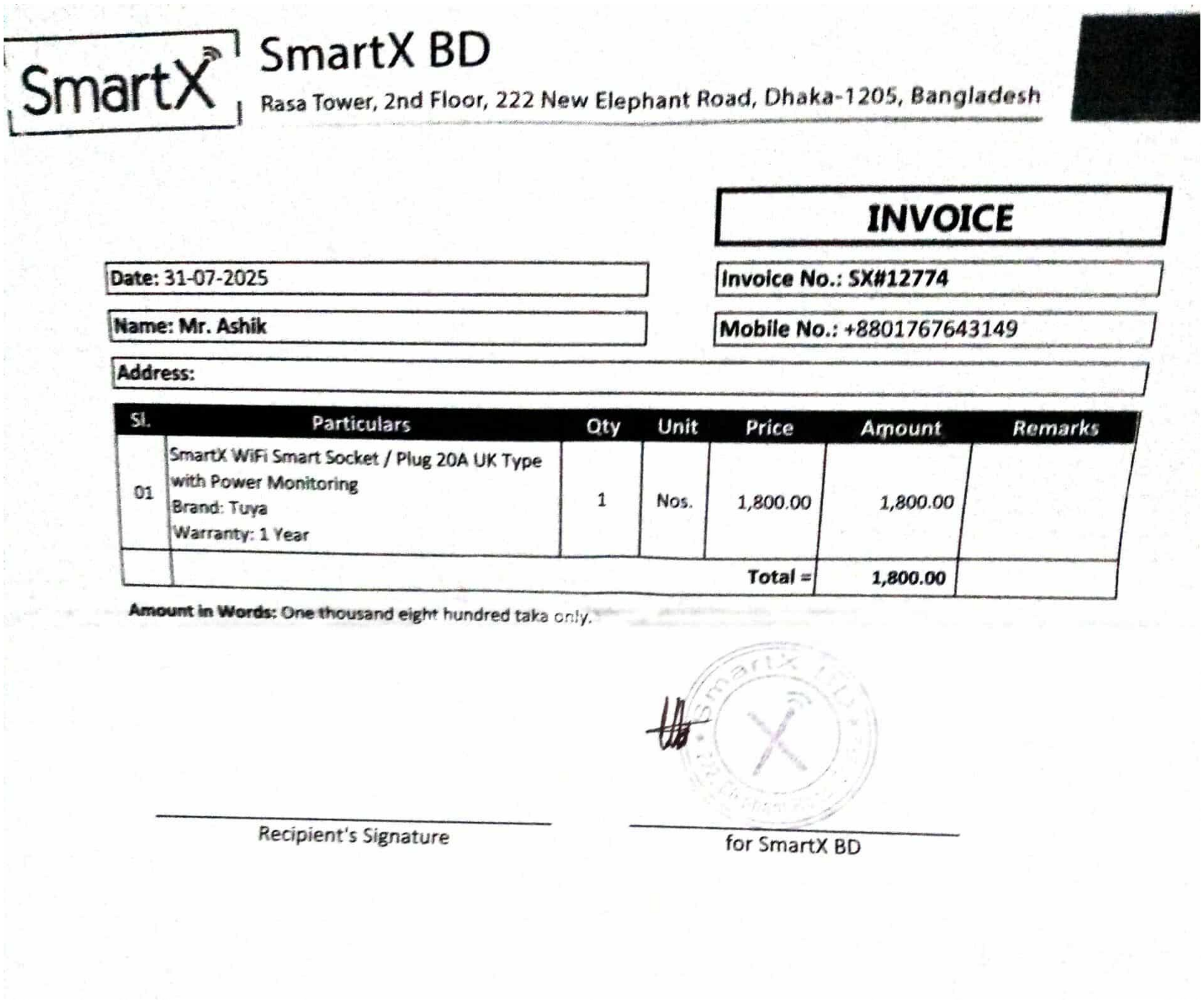
- Datasheets: (https://github.com/SudiptaRoyJoy/cse407mid)

PESTEL Analysis:

- Political: Government regulations around energy consumption and the potential for policy incentives on energy-efficient technology.
- Economic: Cost savings on electricity bills due to optimized energy use, potential market for energy monitoring solutions.
- Social: How society benefits from better energy management, increasing awareness about energy efficiency.

- Technological: The role of IoT in enabling energy management, reliability of the technology, and future improvements.
- Legal: Compliance with data protection laws regarding data collection.
- Environmental: Positive environmental impact due to reduced energy consumption and carbon footprint.

Invoice:



SmartX

SmartX BD

Rasa Tower, 2nd Floor, 222 New Elephant Road, Dhaka-1205, Bangladesh

INVOICE

Date: 31-07-2025	**Invoice No.: SX#12774**
Name: Mr. Ashik	**Mobile No.:** +8801767643149
Address:	

Sl.	Particulars	Qty	Unit	Price	Amount	Remarks
01	SmartX WiFi Smart Socket / Plug 20A UK Type with Power Monitoring Brand: Tuya Warranty: 1 Year	1	Nos.	1,800.00	1,800.00	
				Total =	**1,800.00**	

Amount in Words: One thousand eight hundred taka only.

[illegible]

Recipient's Signature

for SmartX BD

Figure: 03

Conclusion:

This project effectively used IoT technology to track and enhance a personal computer's energy usage with the aid of a Tuya WiFi Smart Socket. The collection and display of voltage, current, and wattage data over a 3-hour period on a web dashboard yielded important information about energy consumption. The system's scalability and flexibility suggested that it might be extended in the future to accommodate additional devices. The project showed both its technical viability and its business benefits, including possible cost savings and improved energy efficiency, after problems with CORS and API authentication were fixed. The program also emphasized the benefits of lowering energy use for the environment and set the stage for future advancements in energy management technologies.