LAB ASSIGNMENTS

ADVANCE PYTHON PROGRAMMING

Name: Harshini Supriya J

Reg no: 22mid0278

Q) web application using flask

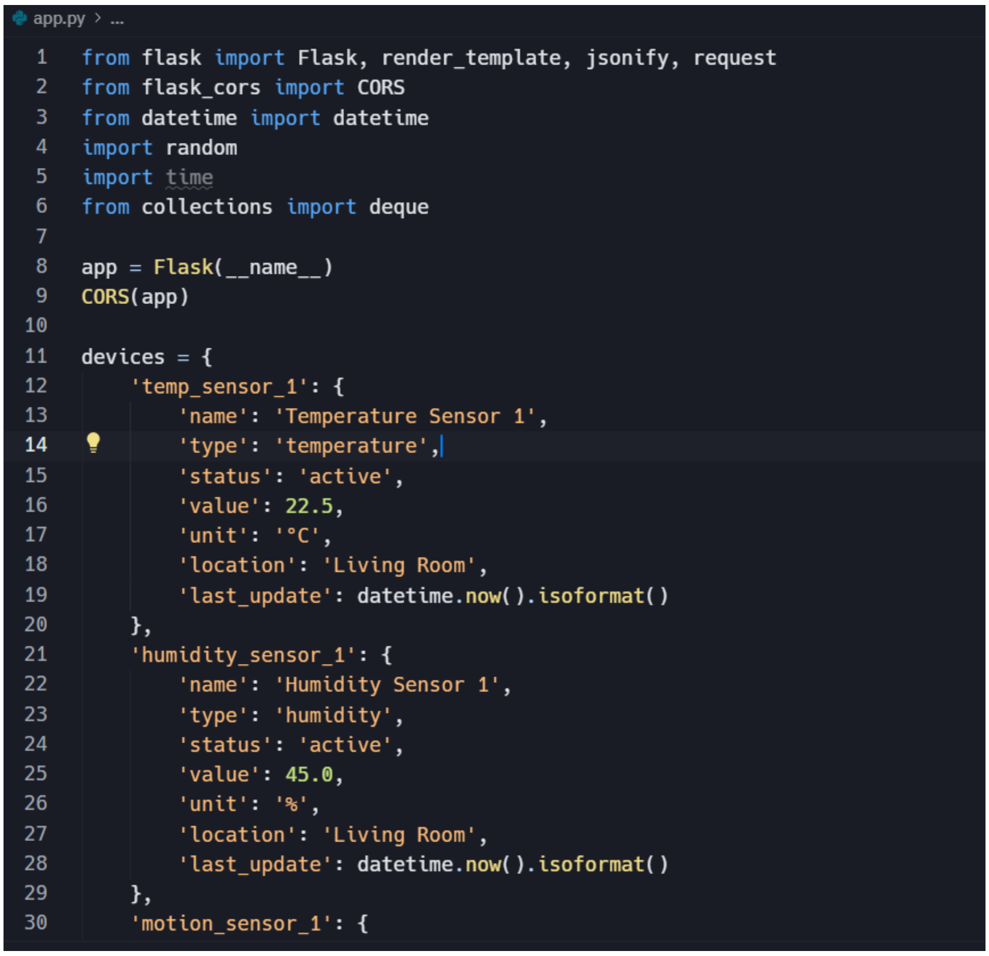
Flask application structure.

1. app.py - Main Flask application
2. requirements.txt - Dependencies
3. templates/index.html - Dashboard UI
4. static/css/style.css - Styling
5. static/js/app.js - Frontend JavaScript
6. .gitignore - Python gitignore

app.py

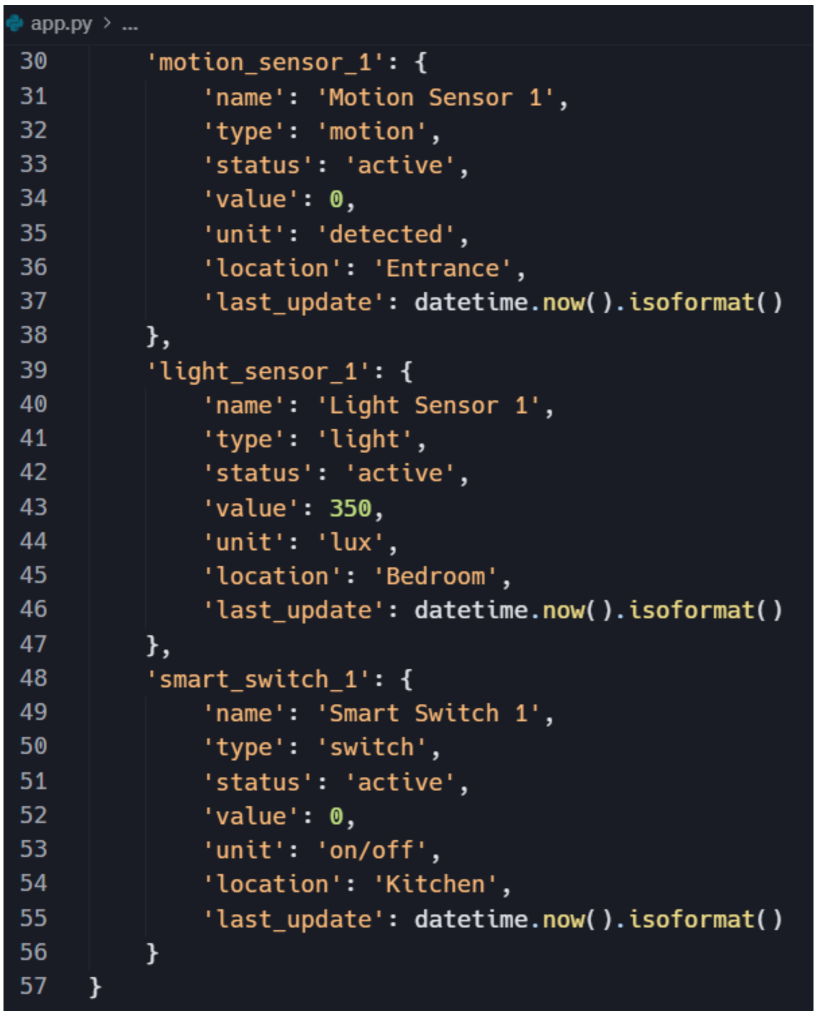
```
from flask import Flask, render_template, jsonify, request

from flask_cors import CORS

from datetime import datetime

import random

import time

from collections import deque


app = Flask(__name__)

CORS(app)
```

```
devices = {
    'temp_sensor_1': {
        'name': 'Temperature Sensor 1',
        'type': 'temperature',
        'status': 'active',
        'value': 22.5,
        'unit': '°C',
        'location': 'Living Room',
        'last_update': datetime.now().isoformat()
    },
    'humidity_sensor_1': {
        'name': 'Humidity Sensor 1',
        'type': 'humidity',
        'status': 'active',
        'value': 45.0,
        'unit': '%',
        'location': 'Living Room',
        'last_update': datetime.now().isoformat()
    },
    'motion_sensor_1': {
        'name': 'Motion Sensor 1',
        'type': 'motion',
        'status': 'active',
        'value': 0,
        'unit': 'detected',
        'location': 'Entrance',
        'last_update': datetime.now().isoformat()
    },
    'light_sensor_1': {
        'name': 'Light Sensor 1',
        'type': 'light',
        'status': 'active',
        'value': 350,
        'unit': 'lux',
        'location': 'Bedroom',
        'last_update': datetime.now().isoformat()
    },
    'smart_switch_1': {
        'name': 'Smart Switch 1',
        'type': 'switch',
        'status': 'active',
        'value': 0,
        'unit': 'on/off',
        'location': 'Kitchen',
        'last_update': datetime.now().isoformat()
    }
}

sensor_history = {device_id: deque(maxlen=20) for device_id in devices.keys()}

def update_sensor_data():
    current_time = datetime.now().isoformat()
    if devices['temp_sensor_1']['status'] == 'active':
        devices['temp_sensor_1']['value'] = round(random.uniform(18.0, 28.0), 1)
        devices['temp_sensor_1']['last_update'] = current_time
        sensor_history['temp_sensor_1'].append({
            'timestamp': current_time,
            'value': devices['temp_sensor_1']['value']
        })

    if devices['humidity_sensor_1']['status'] == 'active':
        devices['humidity_sensor_1']['value'] = round(random.uniform(30.0, 70.0),
1)
        devices['humidity_sensor_1']['last_update'] = current_time
        sensor_history['humidity_sensor_1'].append({
            'timestamp': current_time,
            'value': devices['humidity_sensor_1']['value']
        })

    if devices['motion_sensor_1']['status'] == 'active':
        devices['motion_sensor_1']['value'] = random.choice([0, 0, 0, 1])
        devices['motion_sensor_1']['last_update'] = current_time
        sensor_history['motion_sensor_1'].append({
            'timestamp': current_time,
            'value': devices['motion_sensor_1']['value']
        })

    if devices['light_sensor_1']['status'] == 'active':
        devices['light_sensor_1']['value'] = random.randint(100, 800)
        devices['light_sensor_1']['last_update'] = current_time
        sensor_history['light_sensor_1'].append({
            'timestamp': current_time,
            'value': devices['light_sensor_1']['value']
        })

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/api/devices', methods=['GET'])
def get_devices():
    update_sensor_data()
    return jsonify(devices)

@app.route('/api/devices/<device_id>', methods=['GET'])
def get_device(device_id):
    if device_id in devices:
        return jsonify(devices[device_id])
    return jsonify({'error': 'Device not found'}), 404

@app.route('/api/devices/<device_id>/control', methods=['POST'])
def control_device(device_id):
    if device_id not in devices:
        return jsonify({'error': 'Device not found'}), 404
    data = request.json
    action = data.get('action')

    if action == 'toggle_status':
        current_status = devices[device_id]['status']
        devices[device_id]['status'] = 'inactive' if current_status == 'active' else
'active'
        devices[device_id]['last_update'] = datetime.now().isoformat()
        return jsonify({'success': True, 'new_status': devices[device_id]['status']})

    if action == 'set_value' and 'value' in data:
        devices[device_id]['value'] = data['value']
        devices[device_id]['last_update'] = datetime.now().isoformat()
        return jsonify({'success': True, 'new_value': devices[device_id]['value']})

    return jsonify({'error': 'Invalid action'}), 400

@app.route('/api/history/<device_id>', methods=['GET'])
def get_history(device_id):
    if device_id not in sensor_history:
        return jsonify({'error': 'Device not found'}), 404

    return jsonify(list(sensor_history[device_id]))

@app.route('/api/stats', methods=['GET'])
def get_stats():
    active_devices = sum(1 for device in devices.values() if device['status'] ==
'active')
    total_devices = len(devices)


    return jsonify({
        'total_devices': total_devices,
        'active_devices': active_devices,
        'inactive_devices': total_devices - active_devices
    })


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

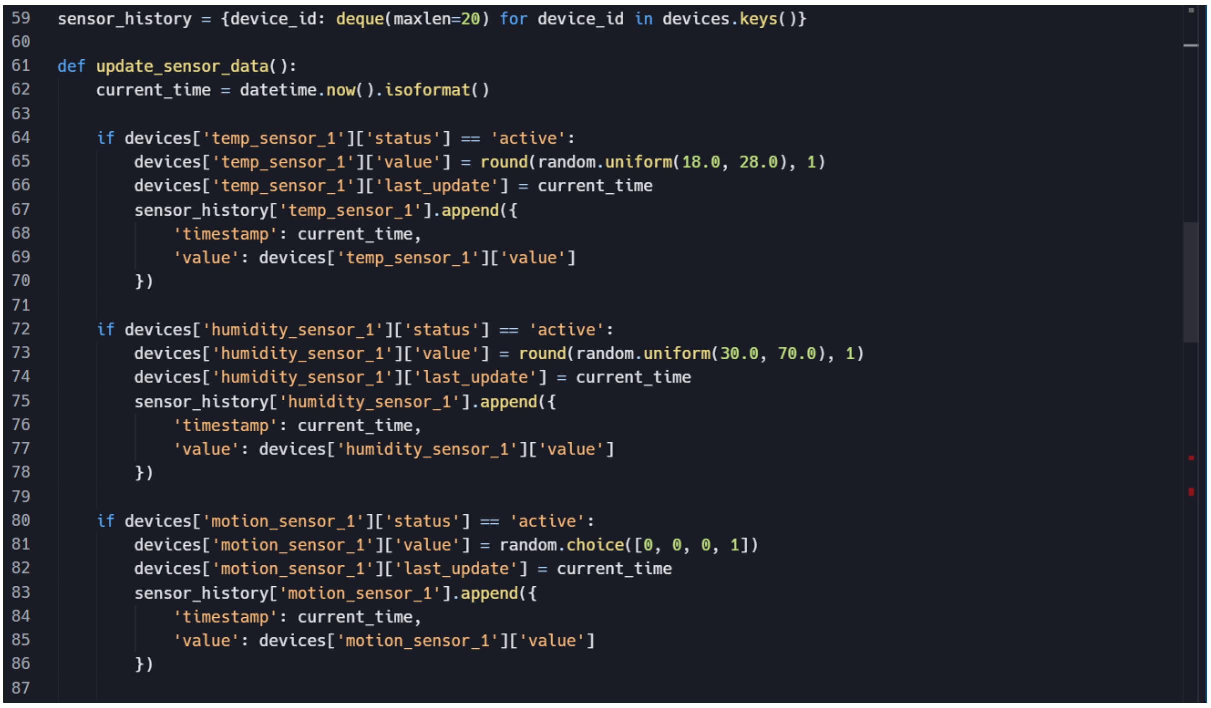


app.py > ...

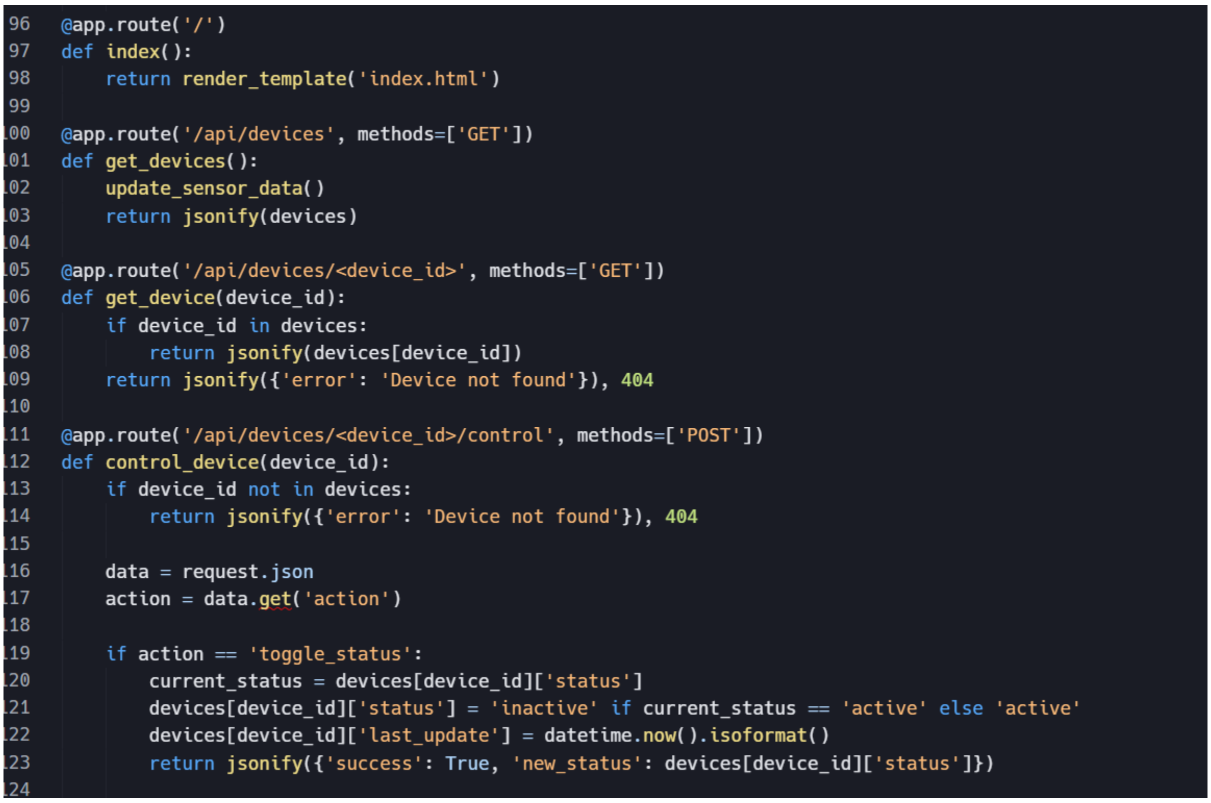
```python
from flask import Flask, render_template, jsonify, request
from flask_cors import CORS
from datetime import datetime
import random
import time
from collections import deque

app = Flask(__name__)
CORS(app)

devices = {
    'temp_sensor_1': {
        'name': 'Temperature Sensor 1',
        'type': 'temperature',
        'status': 'active',
        'value': 22.5,
        'unit': '°C',
        'location': 'Living Room',
        'last_update': datetime.now().isoformat()
    },
    'humidity_sensor_1': {
        'name': 'Humidity Sensor 1',
        'type': 'humidity',
        'status': 'active',
        'value': 45.0,
        'unit': '%',
        'location': 'Living Room',
        'last_update': datetime.now().isoformat()
    },
    'motion_sensor_1': {
```



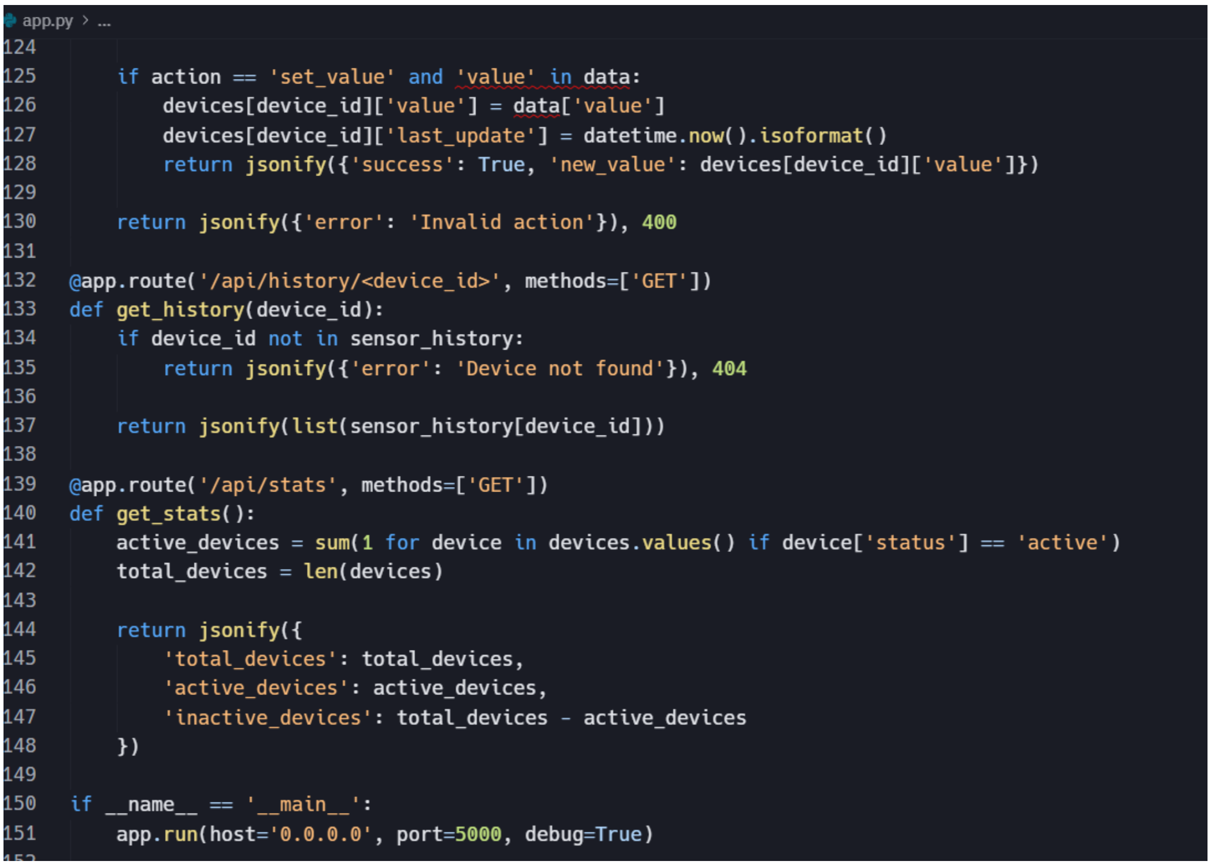
```python
        'motion_sensor_1': {
            'name': 'Motion Sensor 1',
            'type': 'motion',
            'status': 'active',
            'value': 0,
            'unit': 'detected',
            'location': 'Entrance',
            'last_update': datetime.now().isoformat()
        },
        'light_sensor_1': {
            'name': 'Light Sensor 1',
            'type': 'light',
            'status': 'active',
            'value': 350,
            'unit': 'lux',
            'location': 'Bedroom',
            'last_update': datetime.now().isoformat()
        },
        'smart_switch_1': {
            'name': 'Smart Switch 1',
            'type': 'switch',
            'status': 'active',
            'value': 0,
            'unit': 'on/off',
            'location': 'Kitchen',
            'last_update': datetime.now().isoformat()
        }
    }
```



```python
sensor_history = {device_id: deque(maxlen=20) for device_id in devices.keys()}

def update_sensor_data():
    current_time = datetime.now().isoformat()

    if devices['temp_sensor_1']['status'] == 'active':
        devices['temp_sensor_1']['value'] = round(random.uniform(18.0, 28.0), 1)
        devices['temp_sensor_1']['last_update'] = current_time
        sensor_history['temp_sensor_1'].append({
            'timestamp': current_time,
            'value': devices['temp_sensor_1']['value']
        })

    if devices['humidity_sensor_1']['status'] == 'active':
        devices['humidity_sensor_1']['value'] = round(random.uniform(30.0, 70.0), 1)
        devices['humidity_sensor_1']['last_update'] = current_time
        sensor_history['humidity_sensor_1'].append({
            'timestamp': current_time,
            'value': devices['humidity_sensor_1']['value']
        })

    if devices['motion_sensor_1']['status'] == 'active':
        devices['motion_sensor_1']['value'] = random.choice([0, 0, 0, 1])
        devices['motion_sensor_1']['last_update'] = current_time
        sensor_history['motion_sensor_1'].append({
            'timestamp': current_time,
            'value': devices['motion_sensor_1']['value']
        })

```

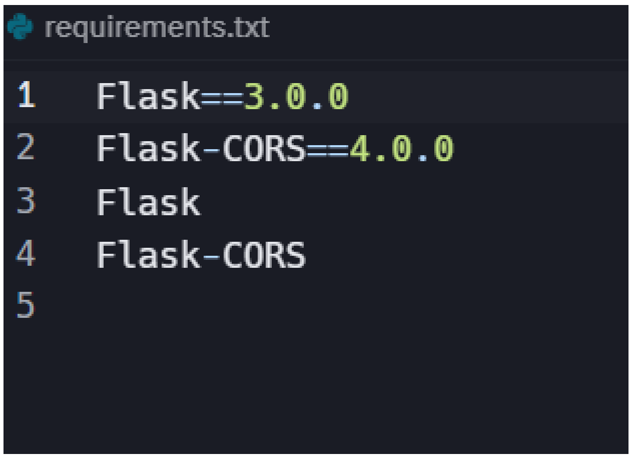


```python
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/api/devices', methods=['GET'])
def get_devices():
    update_sensor_data()
    return jsonify(devices)

@app.route('/api/devices/<device_id>', methods=['GET'])
def get_device(device_id):
    if device_id in devices:
        return jsonify(devices[device_id])
    return jsonify({'error': 'Device not found'}), 404

@app.route('/api/devices/<device_id>/control', methods=['POST'])
def control_device(device_id):
    if device_id not in devices:
        return jsonify({'error': 'Device not found'}), 404

    data = request.json
    action = data.get('action')

    if action == 'toggle_status':
        current_status = devices[device_id]['status']
        devices[device_id]['status'] = 'inactive' if current_status == 'active' else 'active'
        devices[device_id]['last_update'] = datetime.now().isoformat()
        return jsonify({'success': True, 'new_status': devices[device_id]['status']})

```



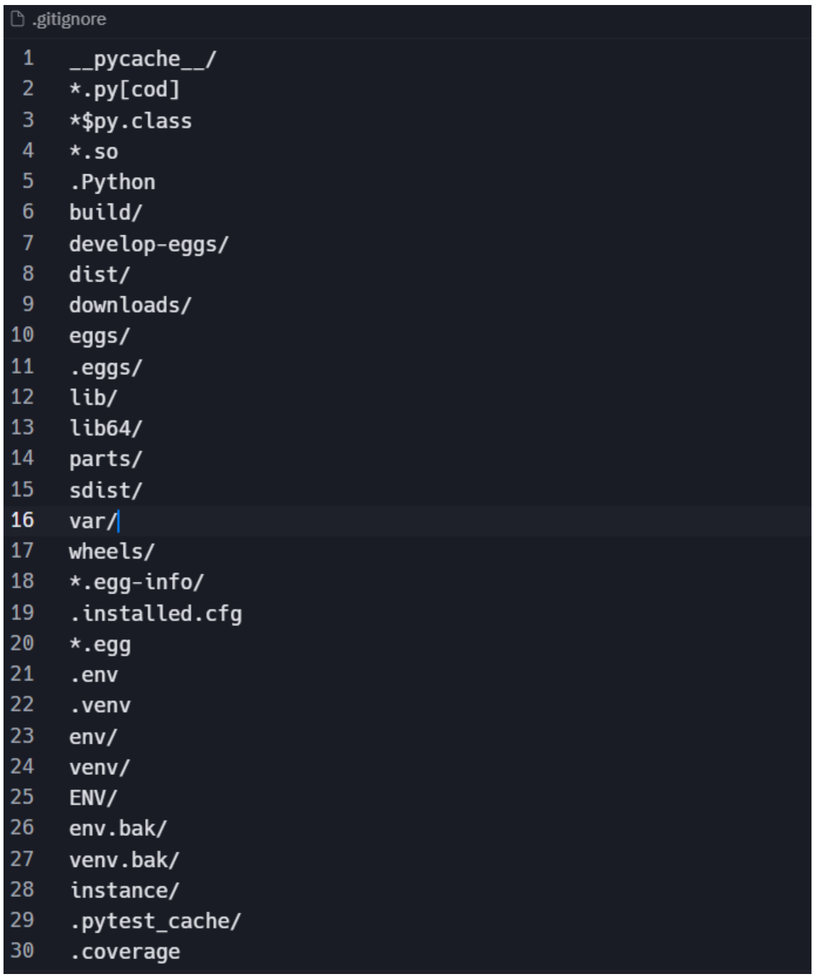
app.py > ...

```python

    if action == 'set_value' and 'value' in data:
        devices[device_id]['value'] = data['value']
        devices[device_id]['last_update'] = datetime.now().isoformat()
        return jsonify({'success': True, 'new_value': devices[device_id]['value']})

    return jsonify({'error': 'Invalid action'}), 400

@app.route('/api/history/<device_id>', methods=['GET'])
def get_history(device_id):
    if device_id not in sensor_history:
        return jsonify({'error': 'Device not found'}), 404

    return jsonify(list(sensor_history[device_id]))

@app.route('/api/stats', methods=['GET'])
def get_stats():
    active_devices = sum(1 for device in devices.values() if device['status'] == 'active')
    total_devices = len(devices)

    return jsonify({
        'total_devices': total_devices,
        'active_devices': active_devices,
        'inactive_devices': total_devices - active_devices
    })

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```



requirements.txt

```
Flask==3.0.0
Flask-CORS==4.0.0
Flask
Flask-CORS

```



.gitignore

```
__pycache__/
*.py[cod]
*$py.class
*.so
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
*.egg-info/
.installed.cfg
*.egg
.env
.venv
env/
venv/
ENV/
env.bak/
venv.bak/
instance/
.pytest_cache/
.coverage
```

Index.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>IoT Dashboard</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.cs
s" rel="stylesheet">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.11.0/font/bootstrap-icons.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
    <nav class="navbar navbar-dark bg-primary">
        <div class="container-fluid">
            <span class="navbar-brand mb-0 h1">
                <i class="bi bi-router"></i> IoT Dashboard
            </span>
            <span class="navbar-text text-white" id="current-time"></span>
        </div>
    </nav>

    <div class="container-fluid mt-4">
        <div class="row mb-4">
            <div class="col-md-4 mb-3">
                <div class="card text-center shadow-sm">
                    <div class="card-body">
                        <i class="bi bi-hdd-network fs-1 text-primary"></i>
                        <h5 class="card-title mt-2">Total Devices</h5>
                        <p class="card-text fs-3" id="total-devices">0</p>
                    </div>
                </div>
            </div>
            <div class="col-md-4 mb-3">
                <div class="card text-center shadow-sm">
                    <div class="card-body">
                        <i class="bi bi-check-circle fs-1 text-success"></i>
                        <h5 class="card-title mt-2">Active Devices</h5>
                        <p class="card-text fs-3" id="active-devices">0</p>
                    </div>
                </div>
            </div>
            <div class="col-md-4 mb-3">
                <div class="card text-center shadow-sm">
                    <div class="card-body">
                        <i class="bi bi-x-circle fs-1 text-danger"></i>
                        <h5 class="card-title mt-2">Inactive Devices</h5>
                        <p class="card-text fs-3" id="inactive-devices">0</p>
                    </div>
                </div>
            </div>
        </div>

        <div class="row">
            <div class="col-lg-8 mb-4">
                <div class="card shadow-sm">
```

```
            <div class="card-header bg-primary text-white">
                <i class="bi bi-graph-up"></i> Sensor Data Trends
            </div>
            <div class="card-body">
                <canvas id="sensorChart"></canvas>
            </div>
        </div>
    </div>

    <div class="col-lg-4 mb-4">
        <div class="card shadow-sm">
            <div class="card-header bg-primary text-white">
                <i class="bi bi-speedometer2"></i> Real-time Monitoring
            </div>
            <div class="card-body" id="realtime-data">
                <div class="text-center text-muted">
                    <p>Loading data...</p>
                </div>
            </div>
        </div>
    </div>
</div>

<div class="row">
    <div class="col-12">
        <div class="card shadow-sm">
```

```
                    <div class="card-header bg-primary text-white">
                        <i class="bi bi-sliders"></i> Device Control Panel
                    </div>
                    <div class="card-body">
                        <div class="table-responsive">
                            <table class="table table-hover">
                                <thead>
                                    <tr>
                                        <th>Device Name</th>
                                        <th>Type</th>
                                        <th>Location</th>
                                        <th>Current Value</th>
                                        <th>Status</th>
                                        <th>Last Update</th>
                                        <th>Action</th>
                                    </tr>
                                </thead>
                                <tbody id="device-list">
                                    <tr>
                                        <td colspan="7" class="text-center text-muted">Loading
devices...</td>
                                    </tr>
                                </tbody>
                            </table>
                        </div>
                    </div>
                </div>
```

```
            </div>
        </div>
    </div>

    <footer class="mt-5 py-3 bg-light">
        <div class="container text-center">
            <p class="text-muted mb-0">IoT Dashboard &copy; 2025 | Real-time
monitoring and control</p>
        </div>
    </footer>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.m
in.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/chart.js@4.4.0/dist/chart.umd.min.js"></scri
pt>
    <script src="{{ url_for('static', filename='js/app.js') }}"></script>
</body>
</html>
```

<u>Styles.css:</u>

```
body {
    background-color: #f8f9fa;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

.navbar {
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
```

```
}

.card {
   border: none;
   border-radius: 10px;
   transition: transform 0.2s, box-shadow 0.2s;
}

.card:hover {
   transform: translateY(-5px);
   box-shadow: 0 4px 12px rgba(0,0,0,0.15);
}

.card-header {
   border-radius: 10px 10px 0 0 !important;
   font-weight: 600;
}

.device-value {
   font-size: 1.5rem;
   font-weight: bold;
   color: #0d6efd;
}

.status-badge {
   font-size: 0.9rem;
```

```
    padding: 0.5rem 1rem;
}

.realtime-item {
    padding: 1rem;
    margin-bottom: 1rem;
    border-radius: 8px;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: white;
    box-shadow: 0 2px 8px rgba(0,0,0,0.1);
}

.realtime-item h6 {
    margin: 0;
    font-size: 0.85rem;
    opacity: 0.9;
}

.realtime-item .value {
    font-size: 2rem;
    font-weight: bold;
    margin: 0.5rem 0;
}

.realtime-item .location {
    font-size: 0.8rem;
    opacity: 0.8;
}

.temp-gradient {
    background: linear-gradient(135deg, #f093fb 0%, #f5576c 100%);
}

.humidity-gradient {
    background: linear-gradient(135deg, #4facfe 0%, #00f2fe 100%);
}

.light-gradient {
    background: linear-gradient(135deg, #fa709a 0%, #fee140 100%);
}

.motion-gradient {
    background: linear-gradient(135deg, #30cfd0 0%, #330867 100%);
}

.btn-control {
    min-width: 100px;
}

footer {
    margin-top: 3rem;
    border-top: 1px solid #dee2e6;
```

```
}

@media (max-width: 768px) {
    .realtime-item .value {
        font-size: 1.5rem;
    }
}

#sensorChart {
    max-height: 300px;
}

.table th {
    background-color: #f8f9fa;
    font-weight: 600;
}

.status-indicator {
    display: inline-block;
    width: 10px;
    height: 10px;
    border-radius: 50%;
    margin-right: 5px;
}

.status-active {
    background-color: #28a745;
    box-shadow: 0 0 5px #28a745;
}

.status-inactive {
    background-color: #dc3545;
}
```

app.js

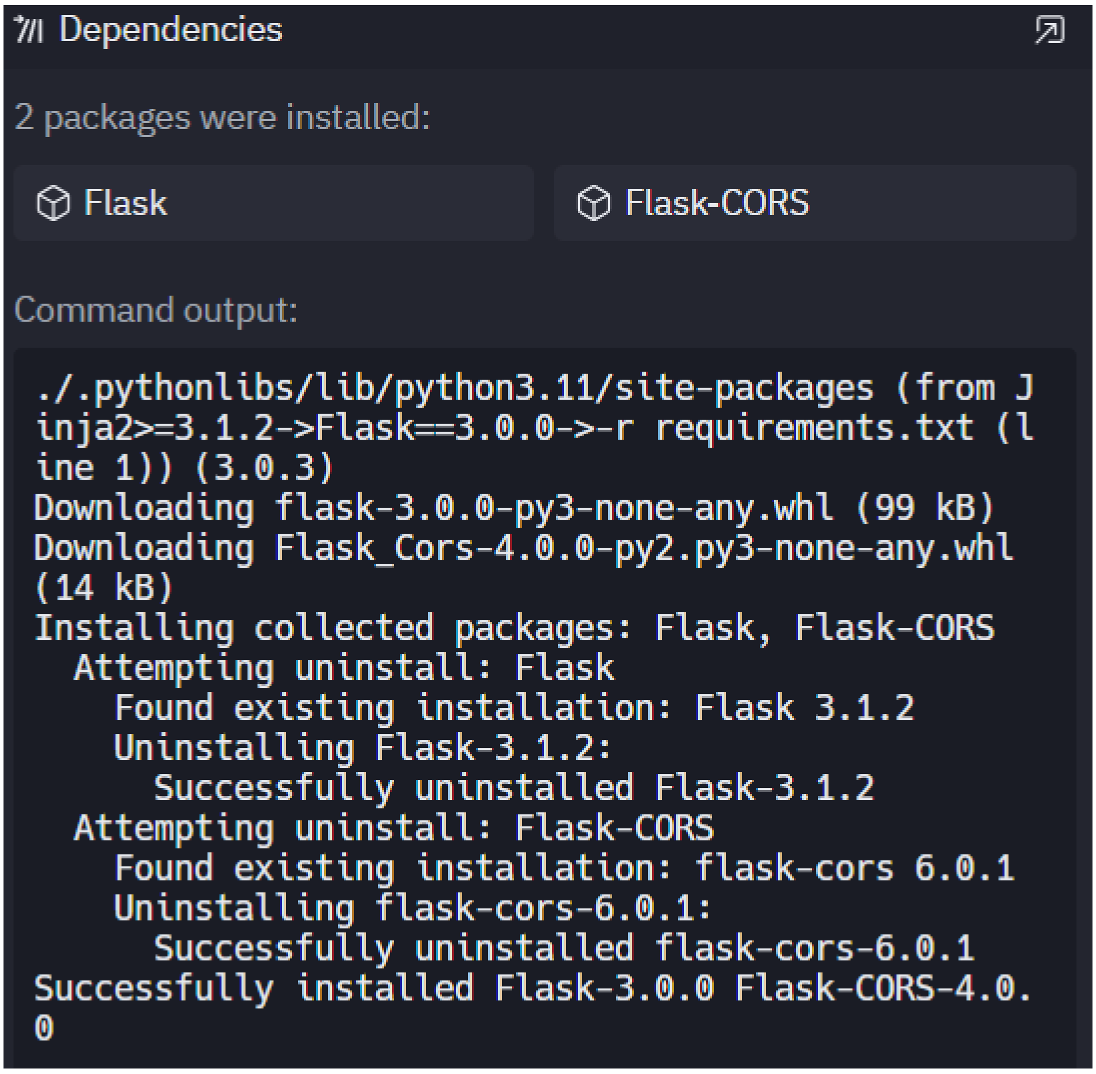
```
let sensorChart;
const API_BASE = window.location.origin;

function updateCurrentTime() {
    const now = new Date();
    const timeString = now.toLocaleString('en-US', {
        hour: '2-digit',
        minute: '2-digit',
        second: '2-digit',
        hour12: true
    });
    document.getElementById('current-time').textContent = timeString;
}

async function fetchStats() {
    try {
        const response = await fetch(`${API_BASE}/api/stats`);
        const stats = await response.json();
        document.getElementById('total-devices').textContent =
stats.total_devices;
        document.getElementById('active-devices').textContent =
stats.active_devices;
        document.getElementById('inactive-devices').textContent =
stats.inactive_devices;
    } catch (error) {
        console.error('Error fetching stats:', error);
    }
}

async function fetchDevices() {
    try {
        const response = await fetch(`${API_BASE}/api/devices`);
        const devices = await response.json();

        updateDeviceList(devices);
        updateRealtimeData(devices);
        updateChart(devices);
    } catch (error) {
        console.error('Error fetching devices:', error);
    }
}

function updateDeviceList(devices) {
    const tbody = document.getElementById('device-list');
    tbody.innerHTML = '';

    Object.entries(devices).forEach(([id, device]) => {
        const row = document.createElement('tr');

        const statusClass = device.status === 'active' ? 'success' : 'danger';
        const statusIndicator = device.status === 'active' ? 'status-active' : 'status-
inactive';

        const lastUpdate = new Date(device.last_update).toLocaleTimeString();

        let valueDisplay = device.value;
        if (device.type === 'motion') {
            valueDisplay = device.value === 1 ? 'Detected' : 'Clear';
        } else if (device.type === 'switch') {
            valueDisplay = device.value === 1 ? 'ON' : 'OFF';
        } else {
            valueDisplay = `${device.value} ${device.unit}`;
        }

        let actionButtons = '';
        if (device.type === 'switch') {
            const switchBtnClass = device.value === 1 ? 'btn-success' : 'btn-
secondary';
            const switchBtnText = device.value === 1 ? 'Turn OFF' : 'Turn ON';
            actionButtons = `
            <button class="btn btn-sm ${switchBtnClass} btn-control me-1"
onclick="toggleSwitchValue('${id}', ${device.value})">
              <i class="bi bi-power"></i> ${switchBtnText}
            </button>
            <button class="btn btn-sm btn-outline-primary btn-control"
onclick="toggleDevice('${id}')">
              ${device.status === 'active' ? 'Deactivate' : 'Activate'}
            </button>
          `;
        } else {
          actionButtons = `
            <button class="btn btn-sm btn-outline-primary btn-control"
onclick="toggleDevice('${id}')">
              ${device.status === 'active' ? 'Deactivate' : 'Activate'}
            </button>
          `;
        }

        row.innerHTML = `
          <td><i class="bi bi-cpu"></i> ${device.name}</td>
          <td><span class="badge bg-secondary">${device.type}</span></td>
          <td><i class="bi bi-geo-alt"></i> ${device.location}</td>
          <td class="device-value">${valueDisplay}</td>
          <td>
            <span class="status-indicator ${statusIndicator}"></span>
            <span class="badge bg-${statusClass}">${device.status}</span>
          </td>
```

```
            <td>${lastUpdate}</td>
            <td>${actionButtons}</td>
        `;

        tbody.appendChild(row);
    });
}

function updateRealtimeData(devices) {
    const container = document.getElementById('realtime-data');
    container.innerHTML = '';

    const realtimeDevices = ['temp_sensor_1', 'humidity_sensor_1',
'light_sensor_1', 'motion_sensor_1'];
    const gradients = {
        'temp_sensor_1': 'temp-gradient',
        'humidity_sensor_1': 'humidity-gradient',
        'light_sensor_1': 'light-gradient',
        'motion_sensor_1': 'motion-gradient'
    };

    realtimeDevices.forEach(deviceId => {
        if (devices[deviceId]) {
            const device = devices[deviceId];
            const div = document.createElement('div');
            div.className = `realtime-item ${gradients[deviceId]}`;
            let valueDisplay = device.value;
            if (device.type === 'motion') {
                valueDisplay = device.value === 1 ? 'Detected!' : 'Clear';
            } else {
                valueDisplay = `${device.value} ${device.unit}`;
            }

            div.innerHTML = `
                <h6>${device.name}</h6>
                <div class="value">${valueDisplay}</div>
                <div class="location"><i class="bi bi-geo-alt"></i>
${device.location}</div>
            `;

            container.appendChild(div);
        }
    });
}

async function updateChart(devices) {
    const tempDevice = devices['temp_sensor_1'];
    const humidityDevice = devices['humidity_sensor_1'];

    try {
        const [tempHistory, humidityHistory] = await Promise.all([
            fetch(`${API_BASE}/api/history/temp_sensor_1`).then(r => r.json()),
            fetch(`${API_BASE}/api/history/humidity_sensor_1`).then(r => r.json())
```

```
]);

const labels = tempHistory.map((_, index) => index + 1);
const tempData = tempHistory.map(item => item.value);
const humidityData = humidityHistory.map(item => item.value);

const ctx = document.getElementById('sensorChart').getContext('2d');

if (sensorChart) {
  sensorChart.data.labels = labels;
  sensorChart.data.datasets[0].data = tempData;
  sensorChart.data.datasets[1].data = humidityData;
  sensorChart.update();
} else {
  sensorChart = new Chart(ctx, {
    type: 'line',
    data: {
      labels: labels,
      datasets: [
        {
          label: 'Temperature (°C)',
          data: tempData,
          borderColor: 'rgb(255, 99, 132)',
          backgroundColor: 'rgba(255, 99, 132, 0.1)',
          tension: 0.4,
          fill: true
        },
        {
          label: 'Humidity (%)',
          data: humidityData,
          borderColor: 'rgb(54, 162, 235)',
          backgroundColor: 'rgba(54, 162, 235, 0.1)',
          tension: 0.4,
          fill: true
        }
      ]
    },
    options: {
      responsive: true,
      maintainAspectRatio: true,
      plugins: {
        legend: {
          position: 'top',
        },
        title: {
          display: false
        }
      },
      scales: {
        y: {
          beginAtZero: false
        }
```

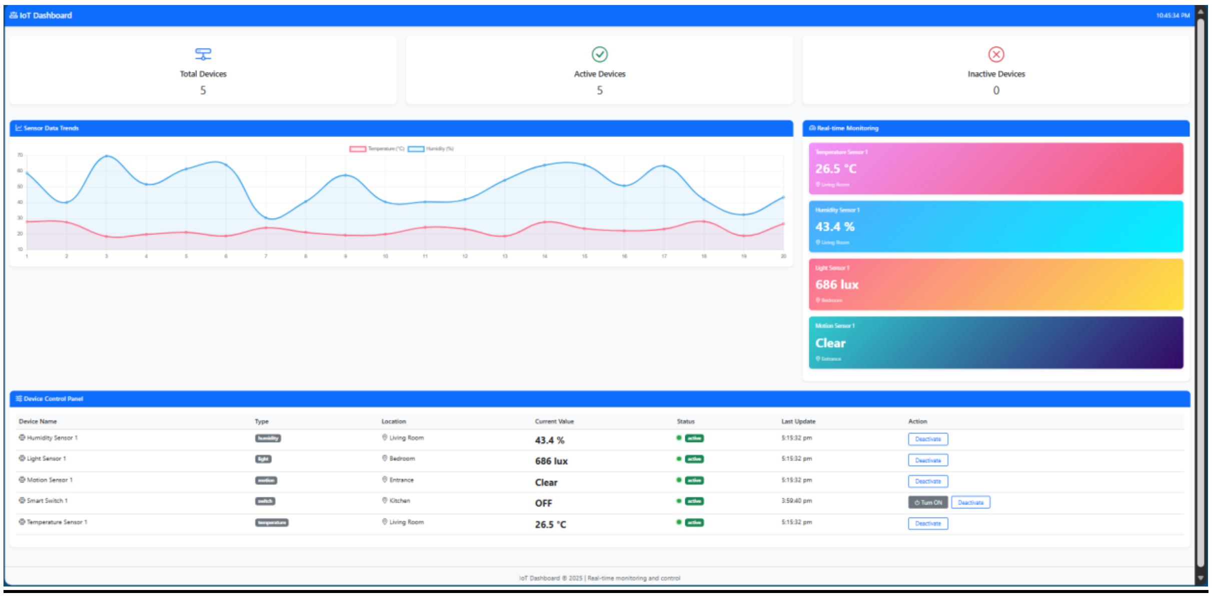
```
                }
            }
        });
    }
  } catch (error) {
      console.error('Error updating chart:', error);
  }
}

async function toggleDevice(deviceId) {
  try {
      const response = await
fetch(`${API_BASE}/api/devices/${deviceId}/control`, {
          method: 'POST',
          headers: {
              'Content-Type': 'application/json'
          },
          body: JSON.stringify({ action: 'toggle_status' })
      });

      const result = await response.json();
      if (result.success) {
          fetchDevices();
      }
  } catch (error) {
      console.error('Error toggling device:', error);
  }
```

```
}

async function toggleSwitchValue(deviceId, currentValue) {
    try {
        const newValue = currentValue === 1 ? 0 : 1;
        const response = await
fetch(`${API_BASE}/api/devices/${deviceId}/control`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({
                action: 'set_value',
                value: newValue
            })
        });

        const result = await response.json();
        if (result.success) {
            fetchDevices();
        }
    } catch (error) {
        console.error('Error toggling switch:', error);
    }
}

function init() {
    updateCurrentTime();
    setInterval(updateCurrentTime, 1000);


    fetchStats();
    fetchDevices();


    setInterval(fetchStats, 5000);
    setInterval(fetchDevices, 3000);
}

document.addEventListener('DOMContentLoaded', init);
```



```
./.pythonlibs/lib/python3.11/site-packages (from J
inja2>=3.1.2->Flask==3.0.0->-r requirements.txt (l
ine 1)) (3.0.3)
Downloading flask-3.0.0-py3-none-any.whl (99 kB)
Downloading Flask_Cors-4.0.0-py2.py3-none-any.whl
(14 kB)
Installing collected packages: Flask, Flask-CORS
  Attempting uninstall: Flask
    Found existing installation: Flask 3.1.2
    Uninstalling Flask-3.1.2:
      Successfully uninstalled Flask-3.1.2
  Attempting uninstall: Flask-CORS
    Found existing installation: flask-cors 6.0.1
    Uninstalling flask-cors-6.0.1:
      Successfully uninstalled flask-cors-6.0.1
Successfully installed Flask-3.0.0 Flask-CORS-4.0.
0
```

Output:



IoT Dashboard 10:45:34 PM

Total Devices 5 | Active Devices 5 | Inactive Devices 0

Sensor Data Trends

Real-time Monitoring

Temperature Sensor 1
26.5 °C
Living Room

Humidity Sensor 1
43.4 %
Living Room

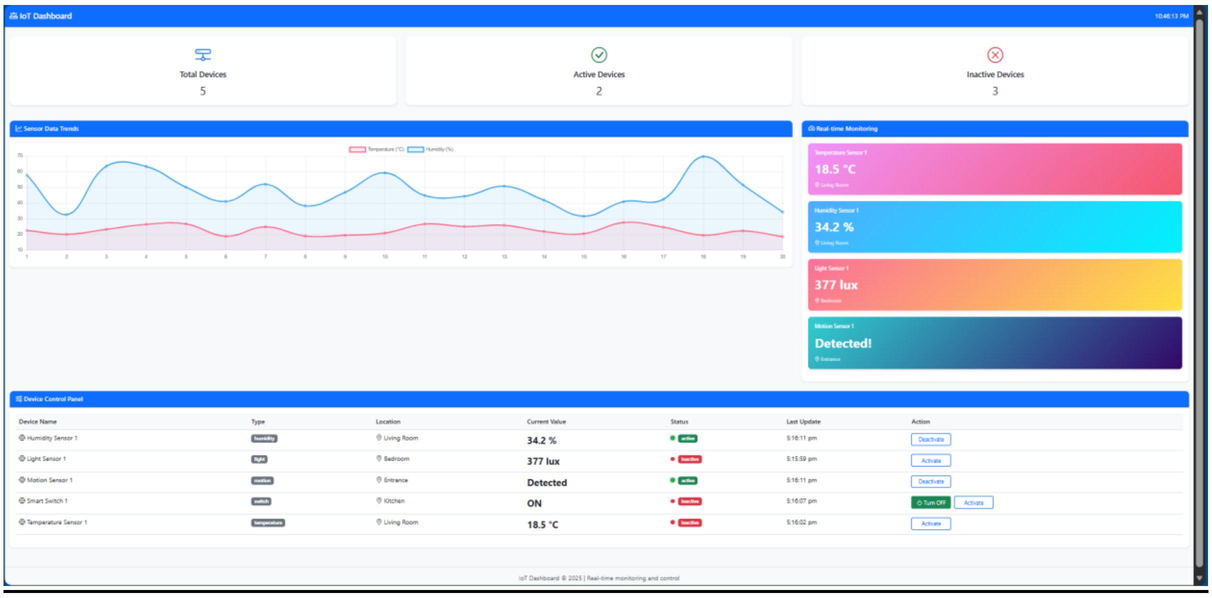
Light Sensor 1
686 lux
Bedroom

Motion Sensor 1
Clear
Entrance

Device Control Panel

Device Name	Type	Location	Current Value	Status	Last Update	Action
Humidity Sensor 1	humidity	Living Room	43.4 %	● active	5:15:32 pm	Deactivate
Light Sensor 1	light	Bedroom	686 lux	● active	5:15:32 pm	Deactivate
Motion Sensor 1	motion	Entrance	Clear	● active	5:15:32 pm	Deactivate
Smart Switch 1	switch	Kitchen	OFF	● active	3:59:40 pm	Turn ON Deactivate
Temperature Sensor 1	temperature	Living Room	26.5 °C	● active	5:15:32 pm	Deactivate

IoT Dashboard © 2025 | Real-time monitoring and control



IoT Dashboard 10:46:13 PM

Total Devices 5 | Active Devices 2 | Inactive Devices 3

Sensor Data Trends

Real-time Monitoring

Temperature Sensor 1
18.5 °C
Living Room

Humidity Sensor 1
34.2 %
Living Room

Light Sensor 1
377 lux
Bedroom

Motion Sensor 1
Detected!
Entrance

Device Control Panel

Device Name	Type	Location	Current Value	Status	Last Update	Action
Humidity Sensor 1	humidity	Living Room	34.2 %	● active	5:16:11 pm	Deactivate
Light Sensor 1	light	Bedroom	377 lux	● inactive	5:15:59 pm	Activate
Motion Sensor 1	motion	Entrance	Detected	● active	5:16:11 pm	Deactivate
Smart Switch 1	switch	Kitchen	ON	● inactive	5:16:07 pm	Turn OFF Activate
Temperature Sensor 1	temperature	Living Room	18.5 °C	● inactive	5:16:02 pm	Activate

IoT Dashboard © 2025 | Real-time monitoring and control