Experiment 2: Implement a micro-website with basic functionalities and measure the following two critical web metrics: 1) Conversion Rate 2) Time on Site

Objective:
Build a micro-website to track and analyze two critical web metrics:

1. **Conversion Rate**: Percentage of sessions ending in newsletter signups
2. **Time on Site**: Duration between session start and end

Key Components:

1. **Session Tracking**:
    - Starts when visiting `/signup`
    - Ends either:
        - After subscription (`/subscribe` route)
        - On cancellation (`/cancel` route)
2. **Database Schema**:

```
CREATE TABLE sessions (
  id TEXT PRIMARY KEY,
  start_time TEXT,
  end_time TEXT,
  converted INTEGER DEFAULT 0
)
```

3. **Metrics Calculated**:

```
# Conversion Rate
(converted_sessions / total_sessions) * 100

# Time on Site
end_time - start_time
```

Program

```
from flask import Flask, request, redirect, session, render_template_string
import sqlite3
import uuid
```

```
from datetime import datetime

app = Flask(__name__)
app.secret_key = 'your-very-secret-key-123'  # Change this for production

# Initialize database
def init_db():
    conn = sqlite3.connect('sessions.db')
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS sessions
                 (id TEXT PRIMARY KEY,
                  start_time TEXT,
                  end_time TEXT,
                  converted INTEGER DEFAULT 0)''')
    conn.commit()
    conn.close()

init_db()

def end_session(session_id, converted=False):
    """Helper function to properly end a session"""
    if session_id:
        conn = sqlite3.connect('sessions.db')
        c = conn.cursor()
        c.execute("UPDATE sessions SET end_time=?, converted=? WHERE id=? AND end_time IS
                  (datetime.now().isoformat(), int(converted), session_id))
        conn.commit()
        conn.close()

@app.route('/')
def home():
    # Terminate any active session when visiting home
    if 'session_id' in session:
        end_session(session['session_id'])
        session.clear()  # Remove all session data

    return '''
    <h1>Welcome to Our Newsletter</h1>
    <p><a href="/signup">Sign Up</a></p>
    <p><a href="/metrics">View Metrics</a></p>
    '''

@app.route('/signup')
def signup():
    # Create new session
    session_id = str(uuid.uuid4())
    session['session_id'] = session_id
```

```
    # Store session in DB
    conn = sqlite3.connect('sessions.db')
    c = conn.cursor()
    c.execute("INSERT INTO sessions VALUES (?, ?, NULL, 0)",
              (session_id, datetime.now().isoformat()))
    conn.commit()
    conn.close()

    return '''
    <h1>Newsletter Signup</h1>
    <form action="/subscribe" method="POST">
        <input type="email" name="email" required placeholder="Your email">
        <button type="submit">Subscribe</button>
    </form>
    <p><a href="/cancel">Cancel Signup</a></p>
    '''

@app.route('/subscribe', methods=['POST'])
def subscribe():
    if 'session_id' not in session:
        return redirect('/')

    email = request.form.get('email', '')
    if email:
        # Mark session as converted
        end_session(session['session_id'], converted=True)
        session.clear()

        return '''
        <h1>Thank You!</h1>
        <p>You've successfully subscribed.</p>
        <p><a href="/">Return Home</a></p>
        '''
    return redirect('/')

@app.route('/cancel')
def cancel():
    if 'session_id' in session:
        end_session(session['session_id'])
        session.clear()
    return redirect('/')

@app.route('/metrics')
def metrics():
    conn = sqlite3.connect('sessions.db')
    c = conn.cursor()

    # Get metrics
```

```
    c.execute("SELECT COUNT(*) FROM sessions WHERE end_time IS NOT NULL")
    total_sessions = c.fetchone()[0]

    c.execute("SELECT COUNT(*) FROM sessions WHERE converted=1")
    converted_sessions = c.fetchone()[0]

    conversion_rate = (converted_sessions / total_sessions * 100) if total_sessions > 0 e

    # Get session data
    c.execute("SELECT id, start_time, end_time FROM sessions ORDER BY start_time DESC")
    sessions = c.fetchall()
    conn.close()

    # Build sessions table
    sessions_table = []
    for s in sessions:
        start = datetime.fromisoformat(s[1])
        end = datetime.fromisoformat(s[2]) if s[2] else None
        duration = str(end - start).split('.')[0] if end else 'N/A'

        sessions_table.append(f'''
        <tr>
            <td>{s[0][:8]}...</td>
            <td>{s[1][11:19]}</td>
            <td>{s[2][11:19] if s[2] else 'N/A'}</td>
            <td>{duration}</td>
        </tr>
        ''')

    return f'''
    <style>
        table {{ border-collapse: collapse; width: 100%; }}
        th, td {{ border: 1px solid #ddd; padding: 8px; text-align: left; }}
        tr:nth-child(even) {{ background-color: #f2f2f2; }}
    </style>
    <h1>Conversion Metrics</h1>
    <p>Total Sessions: {total_sessions}</p>
    <p>Converted Sessions: {converted_sessions}</p>
    <p>Conversion Rate: {conversion_rate:.2f}%</p>

    <h2>Session Durations</h2>
    <table>
        <tr>
            <th>Session ID</th>
            <th>Start Time</th>
            <th>End Time</th>
            <th>Duration</th>
        </tr>
```

```
        {"".join(sessions_table)}
    </table>
    <p><a href="/">Return Home</a></p>
    '''

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080, debug=True)
```