ДОДАТОК А

Код програми db.py

```
# db.py
"""
Модуль роботи з базою даних для АСОТ.
Використовується SQLAlchemy та SQLite.
"""

from sqlalchemy import create_engine, Column, Integer, Float,
DateTime, Text
from sqlalchemy.orm import declarative_base, sessionmaker
from datetime import datetime
from config import DB_URL

# Створення рушія БД
engine = create_engine(DB_URL, echo=False)

# Фабрика сесій
SessionLocal = sessionmaker(bind=engine)

# Базовий клас для ORM-моделей
Base = declarative_base()



class Measurement(Base):
    """
    Таблиця вимірювань параметрів теплоносія.
    """
    __tablename__ = "measurements"

    id = Column(Integer, primary_key=True, index=True)
```

```
    timestamp = Column(DateTime, default=datetime.utcnow,
index=True)
    g = Column(Float, nullable=False)      # масова витрата, кг/с
    t1 = Column(Float, nullable=False)     # температура подачі, °C
    t2 = Column(Float, nullable=False)      # температура зворотy,
°C
    q_kw = Column(Float, nullable=True)      # теплоспоживання за
інтервал, кВт·год


class Anomaly(Base):
    """
    Таблиця виявлених аномалій / рекомендацій бази знань.
    """
    __tablename__ = "anomalies"

    id = Column(Integer, primary_key=True, index=True)
    timestamp = Column(DateTime, default=datetime.utcnow)
    message = Column(Text)  # текстове пояснення / рекомендація


def init_db():
    """
    Ініціалізація (створення) структури бази даних.
    """
    Base.metadata.create_all(bind=engine)
```

ДОДАТОК Б

Код програми acquisition.py

```
# acquisition.py
"""
Модуль збору (або імітації) даних від вимірювальних приладів АСОТ.
У демонстраційному варіанті використовується генерація випадкових
значень.
"""

import random
from datetime import datetime
from db import SessionLocal, Measurement


def simulate_sensor_values():
    """
    Імітація значень сенсорів.
    G – масова витрата, t1 – температура подачі, t2 – температура
звороту.
    Значення підібрані у типовому робочому діапазоні теплового
вузла.
    """
    g = round(random.uniform(0.9, 1.4), 3)    # кг/с
    t1 = round(random.uniform(75, 85), 1)     # °C
    t2 = round(random.uniform(45, 60), 1)     # °C
    return g, t1, t2


def collect_measurement():
    """
    Один цикл збору (або імітації) даних:
    - отримання значень від сенсорів;
    - створення запису в базі даних.
    Повертає об'єкт Measurement.
    """
    session = SessionLocal()
    try:
        g, t1, t2 = simulate_sensor_values()
        m = Measurement(
            timestamp=datetime.utcnow(),
            g=g,
            t1=t1,
            t2=t2
        )
        session.add(m)
        session.commit()
        session.refresh(m)
```

```
    return m
finally:
    session.close()
```

ДОДАТОК В

Код програми analytics.py

```
# analytics.py
"""
Аналітичний модуль АСОТ.
Реалізує розрахунок теплоспоживання та узагальнені показники.
"""

from db import SessionLocal, Measurement
from config import WATER_HEAT_CAPACITY, DELTA_T_SECONDS


def calculate_instant_heat(measurement: Measurement) -> float:
    """
    Розрахунок теплоспоживання за один інтервал Δt
(DELTA_T_SECONDS).
    Формула:
        Q = G * c * (t1 - t2) * Δt / (3.6·10^6), кВт·год
    де:
        G – масова витрата, кг/с;
        c – питома теплоємність, Дж/(кг·°C);
        t1, t2 – температури подачі та зворотy;
        Δt – тривалість інтервалу, с.
    """
    delta_t = measurement.t1 - measurement.t2
    if delta_t <= 0:
        return 0.0

    q_joules = measurement.g * WATER_HEAT_CAPACITY * delta_t *
DELTA_T_SECONDS
    q_kwh = q_joules / (3.6e6)
    return q_kwh


def update_heat_values():
    """
    Проходить по всіх записах без обчисленого q_kw і заповнює це
поле.
    """
    session = SessionLocal()
    try:
        rows = session.query(Measurement).filter(Measurement.q_kw
== None).all()
        for m in rows:
            m.q_kw = calculate_instant_heat(m)
        session.commit()
```

```
    finally:
        session.close()


def calculate_total_heat() -> float:
    """
    Обчислює сумарне теплоспоживання (сума q_kw) за весь період у
базі.
    Повертає значення у кВт·год.
    """
    session = SessionLocal()
    try:
        rows = session.query(Measurement.q_kw).all()
        total_q = sum(q[0] for q in rows if q[0] is not None)
        return total_q
    finally:
        session.close()
```

ДОДАТОК Г

Код програми knowledge_base.py

```
# knowledge_base.py
"""
Модуль бази знань АСОТ.
Реалізує просту продукційну модель (правила типу ЯКЩО–ТО).
"""

from datetime import datetime
from db import SessionLocal, Measurement, Anomaly
from config import DELTA_T_MIN, EFFICIENCY_MIN


def check_rules_for_measurement(m: Measurement):
    """
    Застосування локальних правил до одного вимірювання.
    Повертає список текстових повідомлень (діагностичних висновків).
    """
    messages = []

    delta_t = m.t1 - m.t2

    # Правило 1: мала різниця температур
    if delta_t < DELTA_T_MIN:
        messages.append(
            "Мала різниця температур подача-зворот (ΔT = {:.1f} °C), "
            "можливі тепловтрати або неправильне налаштування системи.".format(delta_t)
        )

    # Спрощений коефіцієнт ефективності KE ~ ΔT / t1
    if m.t1 > 0:
        ke = delta_t / m.t1
        if ke < EFFICIENCY_MIN:
            messages.append(
                "Низький коефіцієнт ефективності KE = {:.2f}, доцільно перевірити "
                "режим роботи теплообмінного обладнання.".format(ke)
            )

    return messages
```

```
def run_knowledge_base():
    """
    Запуск бази знань:
    - зчитування усіх вимірювань;
    - застосування правил;
    - запис виявлених аномалій/рекомендацій до таблиці anomalies.
    """
    session = SessionLocal()
    try:
        rows = session.query(Measurement).all()
        for m in rows:
            msgs = check_rules_for_measurement(m)
            for msg in msgs:
                an      =      Anomaly(timestamp=datetime.utcnow(),
message=msg)
                session.add(an)
        session.commit()
    finally:
        session.close()
```

ДОДАТОК Д

Код програми reports.py

```
# reports.py
"""
Модуль формування простих текстових звітів для АСОТ.
"""

from db import SessionLocal, Measurement, Anomaly
from analytics import calculate_total_heat


def print_summary():
    """
    Виведення узагальненої інформації:
    - кількість вимірювань;
    - сумарне теплоспоживання.
    """
    session = SessionLocal()
    try:
        count = session.query(Measurement).count()
        total_q = calculate_total_heat()
        print("Кількість вимірювань у БД:", count)
        print("Загальне теплоспоживання, кВт·год:", round(total_q,
3))
    finally:
        session.close()


def print_anomalies():
    """
    Виведення усіх виявлених аномалій / рекомендацій бази знань.
    """
    session = SessionLocal()
    try:
        anomalies = session.query(Anomaly).all()
        print("\nВиявлені аномалії та рекомендації бази знань:")
        if not anomalies:
            print("  (немає зафіксованих подій)")
        for a in anomalies:
            print(f"[{a.timestamp}] {a.message}")
    finally:
        session.close()
```

ДОДАТОК Е

Код програми main.py

```
# main.py
"""
Головний модуль (точка входу) автоматизованої системи обліку
теплоспоживання (АСОТ).
Послідовність:
1) Ініціалізація БД.
2) Збір (імітація) даних.
3) Розрахунок теплоспоживання.
4) Запуск бази знань.
5) Формування звіту.
"""

import time
from db import init_db
from acquisition import collect_measurement
from analytics import update_heat_values
from knowledge_base import run_knowledge_base
from reports import print_summary, print_anomalies


def main():
    # 1. Ініціалізація БД
    print("Ініціалізація бази даних...")
    init_db()

    # 2. Збір даних (у реальній системі цикл буде безперервним)
    print("Старт імітації роботи АСОТ...")
```

```
    num_iterations = 10  # для демонстрації
    for i in range(num_iterations):
        m = collect_measurement()
        print(
            f"#{i+1}: G={m.g} кг/с, t1={m.t1} °C, t2={m.t2} °C"
        )
        # У реальній системі тут було б time.sleep(DELTA_T_SECONDS)
        time.sleep(0.2)

    # 3. Оновлюємо розрахунок теплоспоживання
    print("\nРозрахунок теплоспоживання для кожного інтервалу...")
    update_heat_values()

    # 4. Запускаємо базу знань
    print("Запуск бази знань...")
    run_knowledge_base()

    # 5. Формуємо звіт
    print("\nПІДСУМКОВИЙ ЗВІТ:")
    print_summary()
    print_anomalies()


if __name__ == "__main__":
    main()
```

ДОДАТОК Ж

Демонстраційний матеріал

Формат	Позиція.	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кільк.	Примітки
			Текстові документи		
А4	1	ГЮІК 041929.027 ПЗ	Пояснювальна записка	97	с.
			Інші документи		
А4	2		Код програми db.py	2	с.
А4	3		Код програми acquisition.py	2	с.
А4	4		Код програми analytics.py	2	с.
А4	5		Код програми	2	с.
			knowledge_base.py		
А4	6		Код програми reports.py	1	с.
А4	7		Код програми main.py	2	с.
А4	8		Демонстраційний матеріал	22	с.

					ГЮІК 041929.027 ВД			
Змін.	Арк	Номер докум.	Підпис	Дата				
Розроб.		Трифонов О.М.			Розроблення системи автоматизації обліку теплоспоживанням з використанням бази знань на виробничому підприємстві	Літера	Аркуш	Аркушів
Перевір		Аллахверанов Р.Ю				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр		Стародубцев М.Г						
Затверд.		Невлюдов І.Ш.			Відомість кваліфікаційної роботи			