Додаток А
Код програми

base.py
```
from abc import ABC, abstractmethod
from typing import Any, Dict
from app.core.models import SensorTestResult
from app.core.logic import evaluate

class SensorAdapter(ABC):
    def __init__(self, cfg: Dict[str, Any]):
        self.cfg = cfg

    @abstractmethod
    async def _read(self) -> float:
        """Асинхронно зчитати сире числове значення."""
        ...

    async def test(self) -> SensorTestResult:
        try:
            value = await self._read()
            return evaluate(
                raw_value=value,
                mn=self.cfg["expect_min"],
                mx=self.cfg["expect_max"],
                sensor_id=self.cfg["id"],
            )
        except Exception as exc:
            return evaluate(
                raw_value=None,
                mn=0,
                mx=0,
                sensor_id=self.cfg["id"],
                error=str(exc),
            )
```

modbus.py
```
from abc import ABC, abstractmethod
from typing import Any, Dict
from app.core.models import SensorTestResult
from app.core.logic import evaluate


class SensorAdapter(ABC):
    def __init__(self, cfg: Dict[str, Any]):
        self.cfg = cfg
```

```
    @abstractmethod
    async def _read(self) -> float:
        """Асинхронно зчитати сире числове значення."""
        ...

    async def test(self) -> SensorTestResult:
        try:
            value = await self._read()
            return evaluate(
                raw_value=value,
                mn=self.cfg["expect_min"],
                mx=self.cfg["expect_max"],
                sensor_id=self.cfg["id"],
            )
        except Exception as exc:
            return evaluate(
                raw_value=None,
                mn=0,
                mx=0,
                sensor_id=self.cfg["id"],
                error=str(exc),
            )
```

endpoints.py

```
from abc import ABC, abstractmethod
from typing import Any, Dict
from app.core.models import SensorTestResult
from app.core.logic import evaluate

class SensorAdapter(ABC):
    def __init__(self, cfg: Dict[str, Any]):
        self.cfg = cfg

    @abstractmethod
    async def _read(self) -> float:
        """Асинхронно зчитати сире числове значення."""
        ...

    async def test(self) -> SensorTestResult:
        try:
            value = await self._read()
            return evaluate(
                raw_value=value,
                mn=self.cfg["expect_min"],
                mx=self.cfg["expect_max"],
```

```
                sensor_id=self.cfg["id"],
            )
        except Exception as exc:
            return evaluate(
                raw_value=None,
                mn=0,
                mx=0,
                sensor_id=self.cfg["id"],
                error=str(exc),
            )
```

logic.py

```
from datetime import datetime
from app.core.models import SensorTestResult

__all__ = ["evaluate"]

def evaluate(raw_value: float | None,
             mn: float,
             mx: float,
             sensor_id: str,
             error: str | None = None) -> SensorTestResult:
    status = raw_value is not None and mn <= raw_value <= mx and error
is None
    return SensorTestResult(
        id=sensor_id,
        timestamp=datetime.utcnow(),
        value=raw_value,
        status=status,
        error=error,
    )
```

models.py

```
from datetime import datetime
from pydantic import BaseModel, Field

class SensorTestResult(BaseModel):
    id: str = Field(..., description="Унікальний ідентифікатор
датчика")
    timestamp: datetime
    value: float | None
    status: bool
    error: str | None = None
```

settings.py

```
from functools import lru_cache
from pydantic_settings import BaseSettings, SettingsConfigDict

class Settings(BaseSettings):
    app_host: str = "0.0.0.0"
    app_port: int = 8000
    db_url: str = "sqlite:///results.db"
    poll_interval_sec: int = 30

    model_config = SettingsConfigDict(env_file=".env", extra="ignore")

@lru_cache
def get_settings() -> Settings:
    return Settings()
```

modbus_sim.py

```
"""
Modbus-TCP Simulator
"""

import threading
import time
from random import uniform

from pymodbus.datastore import (
    ModbusSparseDataBlock,
    ModbusSlaveContext,
    ModbusServerContext,
)

try:
    # старі версії pymodbus ≤3.5
    from pymodbus.server.sync import StartTcpServer   # type: ignore
except (ModuleNotFoundError, ImportError):
    # нові pymodbus ≥3.9
    from pymodbus.server import StartTcpServer         # type: ignore

class DynBlock(ModbusSparseDataBlock):
    def __init__(self, addr: int):
        super().__init__({addr: 0})
        self.addr = addr
        threading.Thread(target=self._gen, daemon=True).start()

    def _gen(self):
        while True:
```

```
            v = int(uniform(15.0, 30.0) * 10)
            self.setValues(self.addr, [v])
            print(f"[SIM] UPDATE addr={self.addr} <- {v}")
            time.sleep(5)

    def getValues(self, address, count=1):
        vals = super().getValues(address, count)
        print(f"[SIM] READ   addr={address} -> {vals}")
        return vals

def main() -> None:
    slave_ctx = ModbusSlaveContext(hr=DynBlock(100))
    ctx = ModbusServerContext(slaves={1: slave_ctx}, single=False)

    print("[SIM] Modbus-TCP listening on 0.0.0.0:5020 (Slave ID 1, Reg
100)")
    StartTcpServer(context=ctx, address=("0.0.0.0", 5020))

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

__init__.py

```
from fastapi import FastAPI
from app.api.endpoints import router

app = FastAPI(title="Sensor Checker")
app.include_router(router)
```

config.yaml

```
sensors:
  - id: temp_room_1
    proto: modbus
    host: ::1
    port: 5002
    unit: 1
    register: 100
    expect_min: 15.0
    expect_max: 30.0
```

Додаток Б
Демонстраційний матеріал

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК. ХХХХХХ.021 ПЗ	Пояснювальна записка	А4, с.
		Додаткові матеріали	
2		Демонстраційний матеріал у	А4, с.
		вигляді презентації	

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК ХХХХХХ.021 ВД			
Розроб.		Дубина С. С.			Розроблення програмного модуля для отримання результатів перевірки датчиків автоматизованих систем за стандартними протоколами	Літера	Аркуш	Аркушів
Перевір.		Іванов Л.С.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Демська Н. П.						
Затв.		Невлюдов І.Ш.						