

LAMPIRAN 1 (Implementasi SQLAlchemy Models)

File: app/models/user.py

```
from sqlalchemy import Column, Integer, String
from app.db.base import Base

class User(Base):
    __tablename__ = "users"

    id = Column(Integer, primary_key=True)
    username = Column(String(50), unique=True, nullable=False)
    password_hash = Column(String, nullable=False)
    role = Column(String(20), nullable=False)
```

File: app/models/zone.py

```
from sqlalchemy import Column, Integer, String, Float, Enum,
DateTime, func
from sqlalchemy.orm import relationship
from app.db.base import Base
import enum

class ZoneStatus(str, enum.Enum):
    aktif = "aktif"
    nonaktif = "nonaktif"
    maintenance = "maintenance"

class Zone(Base):
    __tablename__ = "zones"

    id = Column(Integer, primary_key=True)
    name = Column(String(100), nullable=False)
    location = Column(String(225))
    target_spl = Column(Float, nullable=False)
    min_spl = Column(Float, nullable=False)
    max_spl = Column(Float, nullable=False)
    status = Column(
        Enum(ZoneStatus, name="zone_status"),
        nullable=False,
        default=ZoneStatus.aktif
    )

    # RELATIONSHIPS
    nodes = relationship("Node", back_populates="zone",
        cascade="all, delete")
    alerts = relationship("Alert", back_populates="zone",
        cascade="all, delete")
```

```

        control_actions = relationship("ControlAction",
back_populates="zone", cascade="all, delete")

        created_at = Column(
            DateTime(timezone=True),
            server_default=func.now(),
            nullable=False
        )

        updated_at = Column(
            DateTime(timezone=True),
            server_default=func.now(),
            onupdate=func.now(),
            nullable=False
        )

```

File: app/models/node.py

```

from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from app.db.base import Base

class Node(Base):
    __tablename__ = "nodes"

    id = Column(Integer, primary_key=True, index=True)
    zone_id = Column(Integer, ForeignKey("zones.id"))

    node_id = Column(String(50), unique=True, index=True)
    ip_address = Column(String(50))
    status = Column(String(20), default="offline")

    # RELATIONSHIPS
    zone = relationship("Zone", back_populates="nodes")
    sensor_readings = relationship("SensorReading",
back_populates="node", cascade="all, delete")
    alerts = relationship("Alert", back_populates="node",
cascade="all, delete")

```

File: app/models/sensor_reading.py

```

from sqlalchemy import Column, Integer, Float, ForeignKey,
DateTime
from sqlalchemy.orm import relationship
from datetime import datetime
from app.db.base import Base
from sqlalchemy.sql import func

class SensorReading(Base):
    __tablename__ = "sensor_readings"

    id = Column(Integer, primary_key=True)

```

```

node_id = Column(Integer, ForeignKey("nodes.id"))
spl = Column(Float)
timestamp = Column(DateTime(timezone=True),
                    server_default=func.now(),
                    nullable=False)

node = relationship("Node", back_populates="sensor_readings")

```

File: app/models/control_action.py

```

from sqlalchemy import Column, Integer, Float, String, ForeignKey,
DateTime, func
from sqlalchemy.orm import relationship
from app.db.base import Base

class ControlAction(Base):
    __tablename__ = "control_actions"

    id = Column(Integer, primary_key=True)
    zone_id = Column(Integer, ForeignKey("zones.id"))
    action_type = Column(String(50))
    gain_db = Column(Float)
    timestamp = Column(
        DateTime(timezone=True),
        server_default=func.now(),
        nullable=False
    )

    zone = relationship("Zone", back_populates="control_actions")

```

File: app/models/alert.py

```

from sqlalchemy import Column, Integer, String, Boolean,
ForeignKey, DateTime
from sqlalchemy.orm import relationship
from datetime import datetime
from app.db.base import Base

class Alert(Base):
    __tablename__ = "alerts"

    id = Column(Integer, primary_key=True)
    zone_id = Column(Integer, ForeignKey("zones.id"))
    node_id = Column(Integer, ForeignKey("nodes.id"))
    severity = Column(String(20))
    message = Column(String)
    resolved = Column(Boolean, default=False)
    timestamp = Column(DateTime, default=datetime.utcnow)

    zone = relationship("Zone", back_populates="alerts")

```

LAMPIRAN 2 (Konfigurasi Database Connection)

File: app/core/config.py

```
from pydantic_settings import BaseSettings

class Settings(BaseSettings):
    SECRET_KEY: str = "supersecretkey"
    ALGORITHM: str = "HS256"
    ACCESS_TOKEN_EXPIRE_MINUTES: int = 60

    MQTT_HOST: str
    MQTT_PORT: int
    MQTT_TOPIC_SPL: str

    DATABASE_URL: str

    class Config:
        env_file = ".env"
        extra = "allow"

settings = Settings()
```

File: app/db/session.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker, declarative_base

DATABASE_URL =
"postgresql+psycopg2://smart_user:SmartPassword123@localhost:5432/smart_mosque_audio"

engine = create_engine(DATABASE_URL)

SessionLocal = sessionmaker(
    autocommit=False,
    autoflush=False,
    bind=engine
)

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

Base = declarative_base()
```

File: app/db/base.py

```
from sqlalchemy.orm import declarative_base

Base = declarative_base()
```

File: app/db/debs.py

```
from app.db.session import SessionLocal

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

File: .env

```
# Database Configuration
DATABASE_URL=postgresql+psycopg2://smart_user:SmartPassword123@localhost:5432/smart_mosque_audio

MQTT_HOST=192.168.0.111
MQTT_PORT=1883
MQTT_TOPIC_SPL=smart-mosque/nodes/+/spl
```

LAMPIRAN 3 (Database Migration dengan Alembic)

Menjalankan script:

Inisialisasi alembic:

```
alembic init alembic
```

Generate Migration:

```
alembic revision --autogenerate -m "initial schema"
```

Terapkan ke Database:

```
alembic upgrade head
```

Setelah eksekusi, semua tabel akan dibuat di PostgreSQL database.

LAMPIRAN 4 (Pydantic Schemas (Request/Response Models))

File: app/schemas/auth.py

```
from pydantic import BaseModel

class LoginRequest(BaseModel):
    username: str
    password: str

class LoginResponse(BaseModel):
    access_token: str
    token_type: str
    expires_in: int
```

File: app/schemas/zone.py

```
from pydantic import BaseModel, Field
from enum import Enum
from app.schemas.node import NodeResponse
from datetime import datetime

class ZoneStatus(str, Enum):
    aktif = "aktif"
    nonaktif = "nonaktif"
    maintenance = "maintenance"

class ZoneCreate(BaseModel):
    name: str
    location: str | None = None
    target_spl: float | None = None
    min_spl: float | None = None
    max_spl: float | None = None
    status: ZoneStatus = ZoneStatus.aktif

class ZoneUpdate(BaseModel):
    target_spl: float | None = None
    min_spl: float | None = None
    max_spl: float | None = None

class ZoneUpdateResponse(BaseModel):
    id: int
    name: str
    location: str
    target_spl: float
    min_spl: float
```

```

    max_spl: float
    updated_at: datetime

    class Config:
        from_attributes = True

class ZoneResponse(ZoneCreate):
    id: int
    name: str
    location: str
    target_spl: float
    min_spl: float
    max_spl: float

    created_at: datetime

    nodes: list[NodeResponse] = Field(default_factory=list)

    class Config:
        from_attributes = True

```

File: app/schemas/node.py

```

from pydantic import BaseModel, Field
from datetime import datetime

class NodeCreate(BaseModel):
    zone_id: int
    node_id: str
    ip_address: str | None = None
    status: str | None = "offline"

class NodeResponse(NodeCreate):
    id: int

    class Config:
        from_attributes = True

class ZoneResponse(BaseModel):
    id: int
    name: str
    location: str
    target_spl: float
    min_spl: float
    max_spl: float
    updated_at: datetime

    nodes: list[NodeResponse] = Field(default_factory=list)

    class Config:
        from_attributes = True

```

File: app/schemas/sensor_readings.py

```
from datetime import datetime
from pydantic import BaseModel

class SensorReadingBase(BaseModel):
    spl: float
    timestamp: datetime

class Config:
    from_attributes = True
```

LAMPIRAN 5 (Router Implementation untuk CRUD Zona)

File: *app/api/v1/zones.py*

```
from fastapi import APIRouter, Depends, HTTPException, Query
from sqlalchemy.orm import Session
from sqlalchemy import func
from sqlalchemy import desc
from sqlalchemy.orm import joinedload
from app.core.security import require_roles
from app.db.deps import get_db
from app.models.zones import Zone
from app.models.nodes import Node
from app.models.sensor_readings import SensorReading
from app.schemas.zone import ZoneCreate, ZoneResponse, ZoneUpdate,
ZoneUpdateResponse

router = APIRouter(prefix="/zones", tags=["Zones"])

@router.post("/", response_model=ZoneResponse,
dependencies=[Depends(require_roles("admin"))])
def create_zone(data: ZoneCreate, db: Session = Depends(get_db)):
    zone = Zone(**data.dict())
    db.add(zone)
    db.commit()
    db.refresh(zone)
    return zone

@router.get("/", dependencies=[Depends(require_roles(["admin",
"operator", "maintenance"]))])
def list_zones(db: Session = Depends(get_db)):

    zones = db.query(Zone).all()
    result = []

    for zone in zones:

        # hitung current_spl dari node di zona tersebut
        avg_spl = (
            db.query(func.avg(SensorReading.spl))
            .join(Node, Node.id == SensorReading.node_id)
            .filter(Node.zone_id == zone.id)
            .scalar()
        )

        result.append({
            "id": zone.id,
            "name": zone.name,
            "location": zone.location,
            "target_spl": zone.target_spl,
            "current_spl": round(avg_spl, 2) if avg_spl else 0,
            "status": zone.status
        })
```

```

        return result

    @router.get("/{zone_id}", response_model=ZoneResponse,
dependencies=[Depends(require_roles(["admin", "operator",
"maintenance"])))])
    def get_zone(zone_id: int, db: Session = Depends(get_db)):

        zone = (
            db.query(Zone)
            .options(joinedload(Zone.nodes))
            .filter(Zone.id == zone_id)
            .first()
        )

        if not zone:
            raise HTTPException(status_code=404, detail="Zone not
found")

        return zone

    @router.put(
       ("/{zone_id}",
        response_model=ZoneUpdateResponse,
        dependencies=[Depends(require_roles(["admin"])))])
    )
    def update_zone(
        zone_id: int,
        payload: ZoneUpdate,
        db: Session = Depends(get_db)
    ):
        # Cari zona berdasarkan ID
        zone = db.query(Zone).filter(Zone.id == zone_id).first()

        if not zone:
            raise HTTPException(status_code=404, detail="Zone not
found")

        # Update konfigurasi SPL
        zone.target_spl = payload.target_spl
        zone.min_spl = payload.min_spl
        zone.max_spl = payload.max_spl

        # Simpan perubahan
        db.commit()
        db.refresh(zone)

        return zone

    @router.delete("/{zone_id}",
dependencies=[Depends(require_roles(["admin"])))])
    def delete_zone(zone_id: int, db: Session = Depends(get_db)):
        zone = db.query(Zone).filter(Zone.id == zone_id).first()
        if not zone:
            raise HTTPException(status_code=404, detail="Zone not
found")
        db.delete(zone)
        db.commit()
        return {"message": "Zone deleted"}

```

LAMPIRAN 6 (Router untuk Data Sensor (SPL Reading))

File: app/api/v1/sensor.py

```
from fastapi import APIRouter, Depends, HTTPException, Query
from sqlalchemy.orm import Session
from sqlalchemy import desc
from app.core.security import require_roles

from app.core.redis_client import redis_client
from app.db.deps import get_db
from app.models.nodes import Node
from app.models.sensor_readings import SensorReading

from typing import Optional
from datetime import datetime, timedelta, timezone

# =====
# ROUTER ZONES
# =====
zones_router = APIRouter(
    prefix="/zones",
    tags=["Sensor"]
)

@zones_router.get(
   ("/{zone_id}/spl/latest",
    dependencies=[Depends(require_roles(["admin", "operator",
"maintenance"]))]
)
def get_latest_spl(
    zone_id: int,
    format: str = Query("json"),
    db: Session = Depends(get_db)
):

    redis_key = f"spl:zone:{zone_id}"

    # =====
    # Redis Cache
    # =====
    cached_spl = redis_client.get(redis_key)

    if cached_spl:
        spl_value = float(cached_spl.decode() if
isinstance(cached_spl, bytes) else cached_spl)

        return {
            "zone_id": zone_id,
            "spl_value": spl_value,
            "unit": "dB",
            "source": "redis_cache"
        }
```

```

# =====
# DB Fallback
# =====
latest_spl = (
    db.query(SensorReading)
    .join(Node, SensorReading.node_id == Node.id)
    .filter(Node.zone_id == zone_id)
    .order_by(desc(SensorReading.timestamp))
    .first()
)

if not latest_spl:
    raise HTTPException(status_code=404, detail="SPL data not
found for this zone")

# cache 30 detik
redis_client.set(redis_key, latest_spl.spl, ex=30)

timestamp = latest_spl.timestamp
if timestamp.tzinfo is None:
    timestamp = timestamp.replace(tzinfo=timezone.utc)

return {
    "zone_id": zone_id,
    "spl_value": latest_spl.spl,
    "unit": "dB",
    "timestamp": timestamp,
    "source": "database"
}

@zones_router.get(
   ("/{zone_id}/spl/history",
    dependencies=[Depends(require_roles(["admin", "operator",
"maintenance"]))]
)
def get_spl_history(
    zone_id: int,
    hours: Optional[int] = 24,
    db: Session = Depends(get_db)
):
    # =====
    # TIME SAFE (UTC AWARE)
    # =====
    time_threshold = datetime.now(timezone.utc) -
timedelta(hours=hours)

    history = (
        db.query(SensorReading)
        .join(Node, SensorReading.node_id == Node.id)
        .filter(
            Node.zone_id == zone_id,
            SensorReading.timestamp >= time_threshold
        )
        .order_by(SensorReading.timestamp.asc())
        .all()
    )

    return [

```

```

        {
            "value": data.spl,
            "timestamp": (
                data.timestamp.replace(tzinfo=timezone.utc)
                if data.timestamp.tzinfo is None
                else data.timestamp
            )
        }
        for data in history
    ]

# =====
# ROUTER NODES
# =====
nodes_router = APIRouter(
    prefix="/nodes",
    tags=["Sensor"]
)

@nodes_router.get(
   ("/{node_id}/readings",
    dependencies=[Depends(require_roles(["admin",
    "maintenance"]))]
)
def get_node_readings(node_id: int, db: Session =
    Depends(get_db)):

    node = db.query(Node).filter(Node.id == node_id).first()
    if not node:
        raise HTTPException(status_code=404, detail="Node not
    found")

    readings = (
        db.query(SensorReading)
        .filter(SensorReading.node_id == node_id)
        .order_by(desc(SensorReading.timestamp))
        .all()
    )

    return [
        {
            "value": r.spl,
            "timestamp": (
                r.timestamp.replace(tzinfo=timezone.utc)
                if r.timestamp.tzinfo is None
                else r.timestamp
            )
        }
        for r in readings
    ]

```

LAMPIRAN 7 (Implementasi *Endpoint* Kontrol Audio)

File: app/app/v1/control.py

```
from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy.orm import Session
from app.core.security import require_roles
from app.db.deps import get_db
from app.models.control_actions import ControlAction
from app.core.mqtt_client import publish_control
from app.schemas.control import ControlRequest

router = APIRouter(prefix="/zones", tags=["Control"])

@router.post("/{zone_id}/control",
dependencies=[Depends(require_roles(["admin", "operator"]))])
def control_zone(
    zone_id: int,
    payload: ControlRequest,
    db: Session = Depends(get_db)
):
    # Validasi jika gain digunakan
    if payload.action_type == "GAIN_ADJUST" and payload.gain_db is
None:
        raise HTTPException(status_code=400, detail="gain_db
required for gain adjust")

    action = ControlAction(
        zone_id=zone_id,
        action_type=payload.action_type,
        gain_db=payload.gain_db
    )

    db.add(action)
    db.commit()
    db.refresh(action)

    # MQTT publish
    publish_control(zone_id, payload.action_type, payload.gain_db)

    return {
        "control_id": action.id,
        "zone_id": action.zone_id,
        "action_type": action.action_type,
        "gain_db": action.gain_db,
        "timestamp": action.timestamp
    }

@router.get("/{zone_id}/control/status",
dependencies=[Depends(require_roles(["admin", "operator", ]))])
def get_last_control_status(
    zone_id: int,
```

```

        db: Session = Depends(get_db)
    ):
        action = (
            db.query(ControlAction)
            .filter(ControlAction.zone_id == zone_id)
            .order_by(ControlAction.timestamp.desc())
            .first()
        )

        if not action:
            raise HTTPException(status_code=404, detail="No control
action found")

        return {
            "zone_id": zone_id,
            "last_action": action.action_type,
            "gain_db": action.gain_db,
            "timestamp": action.timestamp
        }

```

LAMPIRAN 8 (Implementasi Autentikasi JWT)

File: *app/api/v1/auth.py*

```
from fastapi import APIRouter, Depends, HTTPException, status
from sqlalchemy.orm import Session
from datetime import timedelta
from app.schemas.auth import LoginRequest, LoginResponse
from app.services.auth_service import authenticate_user
from app.core.security import create_access_token
from app.db.session import get_db

router = APIRouter(prefix="/auth", tags=["Authentication"])

ACCESS_TOKEN_EXPIRE_MINUTES = 60

@router.post("/login", response_model=LoginResponse)
def login(data: LoginRequest, db: Session = Depends(get_db)):
    user = authenticate_user(db, data.username, data.password)

    if not user:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Username atau password salah"
        )

    access_token_expires =
timedelta(minutes=ACCESS_TOKEN_EXPIRE_MINUTES)

    access_token = create_access_token(
        data={
            "sub": user.username,
            "role": user.role
        },
        expires_delta=access_token_expires
    )

    return {
        "access_token": access_token,
        "token_type": "bearer",
        "expires_in": ACCESS_TOKEN_EXPIRE_MINUTES * 60
    }

@router.post("/logout")
def logout():
    return {"message": "Logout successful"}
```

File: app/schemas/auth.py

```
from pydantic import BaseModel

class LoginRequest(BaseModel):
    username: str
    password: str

class LoginResponse(BaseModel):
    access_token: str
    token_type: str
    expires_in: int
```

LAMPIRAN 9 (Konfigurasi MQTT Client (Integrasi IoT))

File: app/core/mqtt_client.py

```
import json
import asyncio
import paho.mqtt.client as mqtt

from datetime import datetime

from app.db.session import SessionLocal
from app.models.sensor_readings import SensorReading
from app.models.nodes import Node

from app.core.redis_client import redis_client
from app.websocket.socket_manager import sio

from app.websocket.socket_manager import (
    emit_spl_update,
    emit_zone_status_changed
)

# =====
# FASTAPI EVENT LOOP
# =====
MAIN_LOOP = None

def set_main_loop(loop):
    global MAIN_LOOP
    MAIN_LOOP = loop

# =====
# MQTT CONFIG
# =====
broker = "192.168.8.72"
port = 1883

topic = "smart-mosque/nodes/+/spl"
topic_app = "smart-mosque/app/spl"

client = mqtt.Client()

# auto reconnect
client.reconnect_delay_set(min_delay=1, max_delay=5)

# =====
# MQTT CONNECT
# =====
def on_connect(client, userdata, flags, rc):

    if rc == 0:

        print(f"☑ Connected to MQTT Broker at {datetime.now()}")
```

```

        client.subscribe(topic)

        print(f"📡 Subscribed to {topic}")

    else:
        print(f"❌ Failed to connect MQTT Broker | code={rc}")

# =====
# MQTT DISCONNECT
# =====
def on_disconnect(client, userdata, rc):

    print(f"❌ Disconnected from MQTT Broker at {datetime.now()}")

# =====
# MQTT MESSAGE
# =====
def on_message(client, userdata, msg):

    db = SessionLocal()

    try:

        payload = json.loads(msg.payload.decode())

        print(f"📧 MQTT Data Received:", payload)

        node_code = payload.get("node_id")
        zone_id = payload.get("zone_id")
        spl_value = payload.get("spl")

        ip = payload.get("ip_address") or payload.get("ip")

        if not node_code:
            print(f"❌ node_id tidak ditemukan")
            return

        # Redis heartbeat
        redis_client.setex(
            f"node:status:{node_code}",
            10,
            "online"
        )

        print(f"💖 Heartbeat node {node_code}")

        # Cari node
        node = (
            db.query(Node)
            .filter(Node.node_id == node_code)
            .first()
        )

        # Node baru
        if not node:

```

```

node = Node(
    node_id=node_code,
    zone_id=zone_id,
    ip_address=ip,
    status="online"
)

db.add(node)
db.commit()
db.refresh(node)

print("🆕 Node baru dibuat")

# Update node
else:

    old_status = node.status

    if zone_id is not None:
        node.zone_id = zone_id

    if ip:
        node.ip_address = ip

    node.status = "online"

    db.commit()

    if old_status != node.status:

        if MAIN_LOOP:

            asyncio.run_coroutine_threadsafe(
                emit_zone_status_changed({
                    "zone_id": node.zone_id,
                    "node_id": node.node_id,
                    "status": node.status
                }),
                MAIN_LOOP
            )

            print(
                f"📡 Zone {node.zone_id} status changed ->
{node.status}"
            )

            print("♻️ Node diperbarui")

# Simpan SPL
if spl_value is not None:

    try:
        spl_value = float(spl_value)

    except Exception:
        print("❌ SPL bukan angka")
        return

```

```

reading = SensorReading(
    node_id=node.id,
    spl=spl_value
)

db.add(reading)
db.commit()

print("✅ Data SPL saved")

# Redis cache SPL terbaru
redis_client.setex(
    f"spl:zone:{node.zone_id}",
    30,
    spl_value
)

print("⚡ Redis cache updated")

# MQTT ke aplikasi
data_app = {
    "node_id": node.node_id,
    "zone_id": node.zone_id,
    "spl": spl_value,
    "status": "online",
    "timestamp": datetime.utcnow().isoformat()
}

client.publish(
    topic_app,
    json.dumps(data_app)
)

print("📡 Data SPL dikirim ke aplikasi")

# Socket.IO ke website
socket_payload = {
    "node_id": node.node_id,
    "zone_id": node.zone_id,
    "spl": spl_value,
    "status": "online",
    "timestamp": datetime.utcnow().isoformat()
}

if MAIN_LOOP:

    asyncio.run_coroutine_threadsafe(
        emit_spl_update(socket_payload),
        MAIN_LOOP
    )

    print(
        f"📡 Socket.IO -> Zone {node.zone_id} |
SPL={spl_value}"
    )

```

```

        else:

            print("⚠️ FastAPI event loop belum tersedia")

except Exception as e:

    db.rollback()

    print("❌ MQTT PROCESS ERROR:", e)

finally:

    db.close()

# =====
# START MQTT
# =====
def start_mqtt():

    client.on_connect = on_connect
    client.on_disconnect = on_disconnect
    client.on_message = on_message

    client.connect(
        broker,
        port,
        60
    )

    client.loop_start()

    print("🚀 MQTT loop started")

# =====
# MQTT CONTROL PUBLISHER
# =====
def publish_control(
    zone_id: int,
    action_type: str,
    gain_db: float | None = None
):

    topic_control = f"smart-mosque/zones/{zone_id}/control"

    message = {
        "action": action_type.lower(),
        "timestamp": datetime.utcnow().isoformat()
    }

    if gain_db is not None:
        message["gain_db"] = gain_db

    try:

        client.publish(
            topic_control,
            json.dumps(message)

```

```
)

print(
    f"🚀 MQTT Control Sent → "
    f"{topic_control} | {message}"
)

except Exception as e:

    print("❌ MQTT publish error:", e)
```

LAMPIRAN 10 (Main Application Setup)

File: app/main.py

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware

import socketio
import asyncio

from app.api.v1 import (
    auth,
    control,
    sensor,
    zones,
    nodes,
    alerts,
    dashboard,
    health
)

from app.core.mqtt_client import (
    start_mqtt,
    set_main_loop
)

from app.websocket.socket_manager import sio
from app.core.node_monitor import monitor_nodes

# WAJIB agar event Socket.IO terdaftar
from app.websocket import socket_events

app = FastAPI(
    title="Smart Mosque Audio Backend",
    description="Backend CPS untuk sistem audio masjid multi-  
zona",
    version="1.0.0"
)

@app.on_event("startup")
async def startup_event():

    # Simpan event loop FastAPI
    loop = asyncio.get_running_loop()

    set_main_loop(loop)

    print("🚀 Starting MQTT...")

    start_mqtt()
```

```

print("☑ MQTT Started")

asyncio.create_task(
    monitor_nodes()
)

print("☑ Node monitor started")

# =====
# ROUTER REST API
# =====

app.include_router(auth.router, prefix="/api/v1")
app.include_router(zones.router, prefix="/api/v1")
app.include_router(nodes.router, prefix="/api/v1")
app.include_router(sensor.zones_router, prefix="/api/v1")
app.include_router(sensor.nodes_router, prefix="/api/v1")
app.include_router(control.router, prefix="/api/v1")
app.include_router(alerts.router, prefix="/api/v1")
app.include_router/dashboard.router, prefix="/api/v1")
app.include_router(health.router, prefix="/api/v1")

# =====
# CORS
# =====

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],

    # Jika frontend React memakai credentials,
    # ganti "*" dengan daftar domain frontend
    allow_credentials=True,

    allow_methods=["*"],
    allow_headers=["*"],
)

# =====
# SOCKET.IO ASGI WRAPPER
# =====

socket_app = socketio.ASGIApp(
    sio,
    other_asgi_app=app
)

```

LAMPIRAN 11 (Rekaman Data *Sensor Readings*)

pi@raspberrypi: ~				
id	node_id	spl	timestamp	
1	2	85.2	2026-06-18	16:27:47.155734+07
2	3	73.9	2026-06-18	16:27:47.21657+07
3	2	56.4	2026-06-18	16:27:48.154935+07
4	3	60.4	2026-06-18	16:27:48.668188+07
5	2	55.5	2026-06-18	16:27:50.155863+07
6	3	59.6	2026-06-18	16:27:50.669782+07
7	2	59	2026-06-18	16:27:52.168235+07
8	3	60.4	2026-06-18	16:27:52.67389+07
9	2	63.9	2026-06-18	16:27:54.163655+07
10	3	63.1	2026-06-18	16:27:54.679561+07
11	2	68.9	2026-06-18	16:27:56.164213+07
12	3	64.3	2026-06-18	16:27:56.679116+07
13	2	66.8	2026-06-18	16:27:58.155188+07
14	3	73.1	2026-06-18	16:27:58.668955+07
15	2	55.1	2026-06-18	16:28:00.153676+07
16	3	54.2	2026-06-18	16:28:00.669181+07
17	2	62	2026-06-18	16:28:02.164662+07
18	3	57.2	2026-06-18	16:28:02.679009+07
19	2	66.7	2026-06-18	16:28:04.164878+07
20	3	58	2026-06-18	16:28:04.680746+07
21	2	55	2026-06-18	16:28:06.154111+07
22	3	62.9	2026-06-18	16:28:06.666987+07
23	2	55.8	2026-06-18	16:28:08.154754+07
24	3	55.7	2026-06-18	16:28:08.669368+07
25	2	57.2	2026-06-18	16:28:10.1582+07
26	3	56.7	2026-06-18	16:28:10.678135+07
27	2	57.6	2026-06-18	16:28:12.165372+07
28	3	54.8	2026-06-18	16:28:12.679238+07
29	2	60.3	2026-06-18	16:28:14.165066+07
30	3	61.9	2026-06-18	16:28:14.675895+07
31	2	56.8	2026-06-18	16:28:16.16573+07
32	3	63.4	2026-06-18	16:28:16.677606+07
33	2	73.3	2026-06-18	16:28:18.173068+07
34	3	67.1	2026-06-18	16:28:18.677338+07
35	2	69.3	2026-06-18	16:28:20.164768+07
36	3	69.7	2026-06-18	16:28:20.679938+07
37	2	63.9	2026-06-18	16:28:22.16471+07
38	3	59.2	2026-06-18	16:28:22.67951+07
39	2	62.3	2026-06-18	16:28:24.166724+07
40	3	52.6	2026-06-18	16:28:24.679064+07
41	2	55.4	2026-06-18	16:28:26.164392+07
42	3	55.8	2026-06-18	16:28:26.67818+07
43	2	67.5	2026-06-18	16:28:28.168343+07
44	3	64	2026-06-18	16:28:28.677631+07
45	2	63.2	2026-06-18	16:28:30.163774+07
46	3	68.3	2026-06-18	16:28:30.67674+07
47	2	75.2	2026-06-18	16:28:32.164693+07
48	3	58.5	2026-06-18	16:28:32.681993+07
49	2	70.2	2026-06-18	16:28:34.164603+07
50	3	68	2026-06-18	16:28:34.67707+07
51	2	50.7	2026-06-18	16:28:36.165564+07
52	3	60.5	2026-06-18	16:28:36.678723+07
53	2	63.5	2026-06-18	16:28:38.154325+07
54	3	65.6	2026-06-18	16:28:38.668439+07
55	2	68.7	2026-06-18	16:28:40.15502+07

LAMPIRAN 12 (Data Pengujian Waktu *Respons Endpoint*)

No	Response Time	Status
1	114	200
2	69	200
3	79	200
4	67	200
5	72	200
6	75	200
7	68	200
8	64	200
9	72	200
10	68	200
11	74	200
12	64	200
13	70	200
14	72	200
15	75	200
16	76	200
17	134	200
18	62	200
19	76	200
20	70	200
21	79	200
22	65	200
23	62	200
24	75	200
25	78	200
26	79	200
27	64	200
28	74	200
29	117	200
30	84	200
31	98	200
32	85	200
33	107	200
34	88	200
35	110	200
36	63	200
37	59	200
38	79	200
39	73	200
40	68	200
41	76	200
42	68	200
43	73	200
44	73	200
45	70	200
46	79	200
47	79	200
48	74	200
49	70	200
50	69	200

Rata-rata	77,18
Min	59
Max	134
Std Dev	15,27929

No	Response Time	Status
1	258	200
2	22	200
3	24	200
4	23	200
5	24	200
6	23	200
7	22	200
8	23	200
9	25	200
10	23	200
11	21	200
12	24	200
13	24	200
14	21	200
15	24	200
16	23	200
17	23	200
18	24	200
19	25	200
20	24	200
21	24	200
22	21	200
23	24	200
24	24	200
25	23	200
26	22	200
27	24	200
28	21	200
29	24	200
30	21	200
31	22	200
32	23	200
33	25	200
34	24	200
35	25	200
36	20	200
37	24	200
38	24	200
39	24	200
40	24	200
41	23	200
42	25	200
43	25	200
44	24	200
45	21	200
46	24	200
47	25	200
48	20	200
49	24	200
50	23	200

Rata-rata	27,94
Min	20
Max	258
Std Dev	33,22736148

No	Response Time	Status
1	179	200
2	57	200
3	49	200
4	48	200
5	42	200
6	37	200
7	37	200
8	37	200
9	36	200
10	41	200
11	34	200
12	36	200
13	37	200
14	35	200
15	37	200
16	39	200
17	34	200
18	36	200
19	34	200
20	37	200
21	36	200
22	36	200
23	36	200
24	36	200
25	36	200
26	36	200
27	35	200
28	40	200
29	33	200
30	36	200

Rata-rata	42,73333333
Min	33
Max	179
Std Dev	26,2336101

No	Response Time	Status
1	288	200
2	121	200
3	122	200
4	120	200
5	121	200
6	121	200
7	123	200
8	122	200
9	122	200
10	121	200
11	121	200
12	120	200
13	122	200
14	118	200
15	121	200
16	120	200
17	120	200
18	121	200
19	121	200
20	121	200
21	121	200
22	122	200
23	125	200
24	123	200
25	124	200
26	122	200
27	156	200
28	122	200
29	122	200
30	125	200
31	122	200
32	124	200
33	122	200
34	125	200
35	121	200
36	120	200
37	137	200
38	119	200
39	120	200
40	124	200
41	134	200
42	119	200
43	121	200
44	121	200
45	122	200
46	119	200
47	116	200
48	121	200
49	128	200
50	117	200

Rata-rata	126
Min	116
Max	288
Std Dev	24,11706

LAMPIRAN 13 (Data Pengujian Latensi Komunikasi MQTT *End-to-End*)

No	T1	T2	T3
1	12:20:09.010	12:20:09.024	12:20:09.071
2	12:20:11.008	12:20:11.026	12:20:11.076
3	12:20:13.009	12:20:13.026	12:20:13.074
4	12:20:15.009	12:20:15.027	12:20:15.076
5	12:20:17.009	12:20:17.027	12:20:17.070
6	12:20:19.009	12:20:19.028	12:20:19.069
7	12:20:21.009	12:20:21.027	12:20:21.070
8	12:20:23.008	12:20:23.027	12:20:23.068
9	12:20:25.008	12:20:25.027	12:20:25.069
10	12:20:27.008	12:20:27.024	12:20:27.066
11	12:20:29.008	12:20:29.023	12:20:29.066
12	12:20:31.009	12:20:31.026	12:20:31.053
13	12:20:33.009	12:20:33.027	12:20:33.054
14	12:20:35.009	12:20:35.027	12:20:35.067
15	12:20:37.009	12:20:37.027	12:20:37.070
16	12:20:39.009	12:20:39.024	12:20:39.067
17	12:20:41.008	12:20:41.027	12:20:41.069
18	12:20:43.008	12:20:43.027	12:20:43.066
19	12:20:45.008	12:20:45.028	12:20:45.067
20	12:20:47.009	12:20:47.028	12:20:47.071
21	12:20:49.009	12:20:49.036	12:20:49.090
22	12:20:51.009	12:20:51.027	12:20:51.053
23	12:20:53.009	12:20:53.026	12:20:53.052
24	12:20:55.009	12:20:55.027	12:20:55.069
25	12:20:57.008	12:20:57.027	12:20:57.067
26	12:20:59.008	12:20:59.027	12:20:59.068
27	12:21:01.008	12:21:01.027	12:21:01.076
28	12:21:03.008	12:21:03.027	12:21:03.069
29	12:21:05.009	12:21:05.025	12:21:05.067
30	12:21:07.009	12:21:07.030	12:21:07.069
31	12:21:09.009	12:21:09.028	12:21:09.069
32	12:21:11.009	12:21:11.028	12:21:11.069
33	12:21:13.009	12:21:13.027	12:21:13.069
34	12:21:15.008	12:21:15.027	12:21:15.070
35	12:21:17.008	12:21:17.028	12:21:17.070
36	12:21:19.008	12:21:19.027	12:21:19.053
37	12:21:21.009	12:21:21.027	12:21:21.053
38	12:21:23.009	12:21:23.029	12:21:23.069
39	12:21:25.009	12:21:25.028	12:21:25.070
40	12:21:27.009	12:21:27.029	12:21:27.071
41	12:21:29.009	12:21:29.028	12:21:29.071
42	12:21:31.008	12:21:31.028	12:21:31.070
43	12:21:33.008	12:21:33.028	12:21:33.071
44	12:21:35.008	12:21:35.028	12:21:35.069
45	12:21:37.008	12:21:37.027	12:21:37.068
46	12:21:39.009	12:21:39.029	12:21:39.071
47	12:21:41.009	12:21:41.029	12:21:41.071
48	12:21:43.009	12:21:43.029	12:21:43.077
49	12:21:45.009	12:21:45.029	12:21:45.072
50	12:21:47.009	12:21:47.029	12:21:47.072

Latensi MQTT (T2 - T1)	Latensi DB (T3 - T2)	Latensi API (Response Time <i>Endpoint</i>)	Total End-to-End (MQTT + Database + API)
14	47	166	227
18	50	16	84
17	48	19	84
18	49	27	94
18	43	24	85
19	41	18	78
18	43	20	81
19	41	27	87
19	42	26	87
16	42	27	85
15	43	28	86
17	27	41	85
18	27	36	81
18	40	32	90
18	43	16	77
15	43	55	113
19	42	16	77
19	39	24	82
20	39	25	84
19	43	28	90
27	54	25	106
18	26	30	74
17	26	27	70
18	42	26	86
19	40	24	83
19	41	27	87
19	49	59	127
19	42	49	110
16	42	26	84
21	39	23	83
19	41	88	148
19	41	16	76
18	42	16	76
19	43	27	89
20	42	18	80
19	34	15	68
18	26	24	68
20	40	27	87
19	42	29	90
20	42	17	79
19	43	15	77
20	42	27	89
20	43	27	90
20	41	44	105
19	41	44	104
20	42	92	154
20	42	18	80
20	48	19	87
20	43	24	87
20	43	28	91

Kategori Latensi	Rata-Rata (ms)	Min (ms)	Max (ms)	Std Dev (ms)
MQTT	18,72	14	27	1,884848286
DB	41,08	26	54	5,889596501
API	32,04	15	166	25,01832798
End-to-End	91,84	68	227	25,82149081

LAMPIRAN 14 (Data Hasil Pengujian Integrasi MQTT dan Penyimpanan Data)

```
pi@raspberrypi: ~  
31 | 1 | 74.6 | 2026-05-18 11:54:51.168883+07  
32 | 1 | 75.8 | 2026-05-18 11:54:56.185724+07  
33 | 1 | 77.8 | 2026-05-18 11:55:01.202257+07  
34 | 1 | 53.4 | 2026-05-18 11:55:06.117415+07  
35 | 1 | 76.5 | 2026-05-18 11:55:11.090891+07  
36 | 1 | 84.6 | 2026-05-18 11:55:16.09534+07  
37 | 1 | 77.5 | 2026-05-18 11:55:21.170218+07  
38 | 1 | 75.5 | 2026-05-18 11:55:26.100476+07  
39 | 1 | 69.3 | 2026-05-18 11:55:31.102025+07  
40 | 1 | 59.6 | 2026-05-18 11:55:36.120560+07  
41 | 1 | 64.4 | 2026-05-18 11:55:41.138467+07  
42 | 1 | 69.1 | 2026-05-18 11:55:46.147684+07  
43 | 1 | 60.8 | 2026-05-18 11:55:51.173987+07  
44 | 1 | 67.4 | 2026-05-18 11:55:56.192206+07  
45 | 1 | 60.7 | 2026-05-18 11:56:01.107366+07  
46 | 1 | 69 | 2026-05-18 11:56:06.124688+07  
47 | 1 | 69.5 | 2026-05-18 11:56:11.091363+07  
48 | 1 | 66.9 | 2026-05-18 11:56:16.102501+07  
49 | 1 | 72.8 | 2026-05-18 11:56:21.101541+07  
50 | 1 | 69.1 | 2026-05-18 11:56:26.101906+07  
51 | 1 | 70.8 | 2026-05-18 11:56:31.113085+07  
52 | 1 | 72.5 | 2026-05-18 11:56:36.129333+07  
53 | 1 | 77.5 | 2026-05-18 11:56:41.148098+07  
54 | 1 | 69.2 | 2026-05-18 11:56:46.157366+07  
55 | 1 | 65 | 2026-05-18 11:56:51.183114+07  
56 | 1 | 63.2 | 2026-05-18 11:56:56.202741+07  
57 | 1 | 71.3 | 2026-05-18 11:57:01.092826+07  
58 | 1 | 65.4 | 2026-05-18 11:57:06.135559+07  
59 | 1 | 69 | 2026-05-18 11:57:11.096465+07  
60 | 1 | 70.5 | 2026-05-18 11:57:16.106662+07  
(60 rows)  
(END)
```

```
pi@raspberrypi: ~  
151 | 1 | 61.1 | 2026-05-18 13:38:16.300246+07  
152 | 3 | 61.9 | 2026-05-18 13:38:16.712773+07  
153 | 2 | 56.2 | 2026-05-18 13:38:17.168656+07  
154 | 1 | 58.7 | 2026-05-18 13:38:18.301925+07  
155 | 3 | 58.4 | 2026-05-18 13:38:18.712527+07  
156 | 2 | 57.1 | 2026-05-18 13:38:19.167447+07  
157 | 1 | 61.8 | 2026-05-18 13:38:20.303087+07  
158 | 3 | 62.1 | 2026-05-18 13:38:20.710078+07  
163 | 1 | 61.9 | 2026-05-18 13:38:24.303814+07  
168 | 2 | 64.7 | 2026-05-18 13:38:27.168114+07  
173 | 3 | 77.6 | 2026-05-18 13:38:30.712204+07  
159 | 2 | 55.6 | 2026-05-18 13:38:21.166229+07  
164 | 3 | 61.9 | 2026-05-18 13:38:24.713265+07  
169 | 1 | 61.9 | 2026-05-18 13:38:28.305033+07  
174 | 2 | 74.8 | 2026-05-18 13:38:31.168943+07  
160 | 1 | 57.7 | 2026-05-18 13:38:22.302774+07  
165 | 2 | 61.2 | 2026-05-18 13:38:25.166767+07  
170 | 3 | 65.4 | 2026-05-18 13:38:28.714274+07  
175 | 1 | 61.9 | 2026-05-18 13:38:32.302305+07  
161 | 3 | 63.3 | 2026-05-18 13:38:22.711139+07  
166 | 1 | 60.6 | 2026-05-18 13:38:26.302836+07  
171 | 2 | 66.9 | 2026-05-18 13:38:29.168052+07  
176 | 3 | 65.6 | 2026-05-18 13:38:32.712494+07  
162 | 2 | 57.5 | 2026-05-18 13:38:23.168289+07  
167 | 3 | 63.5 | 2026-05-18 13:38:26.711821+07  
172 | 1 | 73.8 | 2026-05-18 13:38:30.302782+07  
177 | 2 | 70.4 | 2026-05-18 13:38:33.168554+07  
178 | 1 | 68.5 | 2026-05-18 13:38:34.301825+07  
179 | 3 | 65.8 | 2026-05-18 13:38:34.712195+07  
180 | 2 | 61.5 | 2026-05-18 13:38:35.167738+07  
(180 rows)  
(END)
```

LAMPIRAN 15 (Master Data Validasi dan Evaluasi)

1. Pengujian Fungsional - Prosedur Detail

Pengujian 1: Manajemen Zona Audio (CRUD Zones)

Memastikan *endpoint* zona dapat menambah, membaca, mengupdate, dan menghapus data zona audio dengan benar.

Langkah Pengujian:

1. Test Case 1.1: Tambah Zona Baru (POST)

- Request: POST `http://192.168.8.72:8000/api/v1/zones`
- Headers: Content-Type: application/json
- Body:

```
{
  "name": "ZONA Sholat",
  "location": "Ruang Utama",
  "target_spl": 70,
  "min_spl": 60,
  "max_spl": 80,
  "status": "aktif"
}
```
- Expected Result:
 - Status Code: 200 Created
 - Response Body: Data zona dengan ID baru
 - Database: Record baru tercatat di tabel zones
- Kriteria Sukses: ID zona ter-generate otomatis, timestamp `created_at` tercatat

2. Test Case 1.2: Ambil Semua Zona (GET)

- Request: GET `http://192.168.8.72:8000/api/v1/zones`
- Expected Result:
 - Status Code: 200 OK
 - Response Body: Array zona dengan minimal 1 zona
 - Format: `[{id, name, location, target_spl, current_spl, status}]`
- Kriteria Sukses: Semua zona yang ditambahkan sebelumnya muncul dalam list

3. Test Case 1.3: Ambil Detail Zona (GET by ID)

- Request: GET `http://192.168.8.72:8000/api/v1/zones/1`
- Expected Result:
 - Status Code: 200 OK

- Response Body: Detail lengkap zona ID=1 yang mencakup informasi zona dan daftar node yang terhubung pada zona tersebut.
 - Kriteria Sukses: Data zona sesuai dengan yang disimpan di database
- 4. Test Case 1.4: Update Target SPL Zona (PUT)
 - Request: PUT `http://192.168.8.72:8000/api/v1/zones/1`
 - Body:


```
{
    "target_spl": 60,
    "min_spl": 50,
    "max_spl": 70
}
```
 - Expected Result:
 - Status Code: 200 OK
 - Response Body: Data zona ter-update
 - Database: Nilai target_spl berubah menjadi 78
 - Kriteria Sukses: Update tercatat dengan timestamp updated_at
- 5. Test Case 1.5: Hapus Zona (DELETE)
 - Request: DELETE `http://192.168.8.72:8000/api/v1/zones/1`
 - Expected Result:
 - Status Code: 200 OK
 - Database: Record zona ID=1 dihapus atau di-soft delete
 - Kriteria Sukses: Zona tidak muncul lagi di GET /zones

Dokumentasi Hasil: Catat hasil pengujian di Tabel 2 (lihat Bagian 2).

Tabel 2: Hasil Pengujian Fungsional - Manajemen Zona Audio

No	Test Case	Metode	Endpoint	Input Data (Dummy)	Expected Status	Expected Output	Actual Status	Actual Output	Hasil	Catatan
1.1	Tambah Zona Baru	POST	/api/v1/zones	{ "name": "Zona Sholat", "location": "Ruang Utama", "target_spl": 70, "min_spl": 60, "max_spl": 80, "status": "aktif" }	200 Created	{ "name": "Zona Sholat", "location": "Ruang Utama", "target_spl": 70.0, "min_spl": 60.0, "max_spl": 80.0, "status": "aktif", "id": 7, "created_at": "2026-05-14T13:46:26.981575+07:00", "nodes": [] }	200	{ "id": 1, ... }	✓ Pass	Zona berhasil dibuat
1.2	Ambil Semua Zona	GET	/api/v1/zones	-	200 OK	[{ "id": 1, "name": "Zona Sholat", "location": "Ruang Utama", "target_spl": 70.0, "current_spl": 0, "status": "aktif" }, { "id": 2, "name": "Zona Serambi Kanan", "location": "Serambi Kanan", "target_spl": 80.0, "current_spl": 0, "status": "aktif" }, { "id": 3, "name": "Zona Serambi Kiri", "location": "Serambi Kiri", "target_spl": 80.0, "current_spl": 0, "status": "aktif" }, { "id": 4, "name": "Zona Serambi Depan", "location": "Serambi Depan", "target_spl": 80.0, "current_spl": 0, "status": "aktif" }, { "id": 5, "name": "Zona Mikhrab", "location": "Tempat Imam Sholat", "target_spl": 60.0, "current_spl": 0, "status": "aktif" }]	200	Array 2 zona	✓ Pass	Semua zona tampil

1.3	Ambil Detail Zona	GET	/api/v1/zones/1	-	200 OK	{ "name": "Zona Sholat", "location": "Ruang Utama", "target_spl": 70.0, "min_spl": 60.0, "max_spl": 80.0, "status": "aktif", "id": 1, "created_at": "2026-05-14T14:07:55.367328+07:00", "nodes": [{"zone_id": 1, "node_id": "node01", "ip_address": "192.168.8.238", "status": "online", "id": 1}]} }	200	Detail lengkap	✓ Pass	Data sesuai DB
1.4	Update Target SPL	PUT	/api/v1/zones/1	{ "target_spl": 60, "min_spl": 50, "max_spl": 70} }	200 OK	{ "id": 1, "name": "Zona Sholat", "location": "Ruang Utama", "target_spl": 60.0, "min_spl": 50.0, "max_spl": 70.0, "updated_at": "2026-05-14T14:24:19.880947+07:00"} }	200	Target updated	✓ Pass	Update berhasil
1.5	Hapus Zona	DELETE	/api/v1/zones/1	-	200 OK	{ "message": "Zone deleted" }	200	Zone deleted	✓ Pass	Zona terhapus
1.6	Validasi Input SPL	POST	/api/v1/zones	{ "name": "Test", "target_spl": "abc" }	422 Unprocessable	Error validation	422	Field error	✓ Pass	Validasi berfungsi

Catatan Pengujian:

- Tanggal Pengujian: 2026-05-14
- Penguji: Jauharatun Nafiisah
- Lingkungan: Local Development (FastAPI + PostgreSQL)
- Jumlah Zona Awal: 0
- Jumlah Zona Akhir: 5

Pengujian 2: Telemetri Sensor SPL (Sensor Readings)

Memastikan data SPL dari node MQTT tercatat dengan benar di database dan dapat diakses via API.

Langkah Pengujian:

1. Test Case 2.1: Kirim Data SPL via MQTT

- Aksi: Publish pesan MQTT dari simulator atau node fisik
- Topic: `smart-mosque/nodes/+/spl`
- Payload:

```
{
  "node_id": "node01",
  "zone_id": 1,
  "ip_address": 192.168.8.72,
  "status": "online",
  "spl": 55.1
}
```
- Expected Result:
 - Backend log menunjukkan pesan diterima
 - Database: Record baru di tabel `sensor_readings` dan table nodes untuk registrasi node
- Kriteria Sukses: Data SPL tercatat

2. Test Case 2.2: Ambil SPL Terkini per Zona (GET)

- Request: GET
`http://192.168.8.72:8000/api/v1/sensor/1/spl/latest?format=json`
- Expected Result:
 - Status Code: 200 OK
 - Response Body:

```
{
  "zone_id": 1,
  "spl_value": 55.1,
  "unit": "dB",
  "timestamp": "2026-05-14T14:27:53.383369+07:00"
  "source": "database",
}
```
- Kriteria Sukses: Nilai SPL sesuai dengan data data SPL terakhir yang diterima dari node sensor melalui MQTT dan tersimpan pada database sistem.

3. Test Case 2.3: Ambil Riwayat SPL per Zona (GET with Time Range)

- Request: GET
`http://192.168.8.72:8000/api/v1/sensor/1/spl/history?hours=24`
- Expected Result:

- Status Code: 200 OK
 - Response Body: Array data SPL dalam rentang waktu
 - Kriteria Sukses: Data terurut berdasarkan timestamp, minimal 10 record jika simulasi berjalan 20 detik
4. Test Case 2.4: Validasi Format Data
- Aksi: Kirim data MQTT dengan format salah (SPL = "abc")
 - Expected Result:
 - *Backend* log menunjukkan error validasi
 - Data tidak tercatat di database
 - Kriteria Sukses: Sistem reject data invalid tanpa crash

Dokumentasi Hasil: Catat hasil pengujian di Tabel 3 (lihat Bagian 2).

Tabel 3: Hasil Pengujian Fungsional - Telemetri Sensor SPL

No	Test Case	Aksi	MQTT Topic	Payload (Dummy)	Expected Database	Actual Database	Hasil	Catatan
2.1	Kirim Data SPL	Publish MQTT	smart-mosque/nodes/node01/spl	{"node_id": "node01", "zone_id": 1, "ip_address": 192.168.8.238, "status": "online", "spl": 55.1}	1 record baru di sensor_readings dan registrasi node di table nodes	✓ Record created	✓ Pass	Data tercatat
2.2	Ambil SPL Terkini	GET API	/api/v1/zones/1/spl/latest	-	SPL = 55.1	SPL = 55.1	✓ Pass	Sesuai data MQTT
2.3	Riwayat SPL Zona	GET API	api/v1/sensor/1/spl/history?hours=24	-	Array 10 record	10 records	✓ Pass	Time range benar
2.4	Validasi Format	Publish MQTT	smart-mosque/nodes/+spl	{"node_id": "node01", "zone_id": 1, "ip_address": 192.168.8.238, "spl": "ABC"}	No record created	✓ Rejected	✓ Pass	Error handling OK
2.5	Multi-Node Concurrent	3 nodes publish	smart-mosque/nodes/+spl	3 payloads simultan	3 records	✓ 3 records	✓ Pass	No collision

Catatan Pengujian:

- Tanggal Pengujian: 2026-05-14
- Broker MQTT: Mosquitto 2.0.18
- Interval Publikasi: 2 detik
- Total Data Point: 50 per zona

Pengujian 3: Manajemen Alert & Peringatan Sistem

Memastikan sistem dapat mendeteksi kondisi abnormal dan mencatat peringatan.

Langkah Pengujian:

1. Test Case 4.1: Deteksi SPL Melebihi Threshold
 - Aksi: Kirim data MQTT dengan SPL = 95.4 dB (melebihi max_spl = 90 dB)
 - Expected Result:
 - Backend membuat record alert di tabel alerts
 - Alert type: "SPL_OVER_THRESHOLD"
 - Severity: "HIGH"
 - Kriteria Sukses: Alert tercatat dengan timestamp dan status resolved = false
2. Test Case 4.2: Ambil Daftar Alert Aktif (GET)
 - Request: GET `http://192.168.8.72:8000/api/v1/alerts?resolved=false`
 - Expected Result:
 - Status Code: 200 OK
 - Response Body: Array alert dengan status unresolved
 - Kriteria Sukses: Alert SPL_OVER_THRESHOLD dari Test 4.1 muncul
3. Test Case 4.3: Resolve Alert (POST)
 - Request: POST `http://192.168.8.72:8000/api/v1/alerts/1/resolve`
 - Body:

```
{
  "resolution_notes": "SPL adjusted back to normal range"
}
```
 - Expected Result:
 - Status Code: 200 OK
 - Database: Alert ID=1 status berubah menjadi resolved = true
 - Kriteria Sukses: Alert tidak muncul lagi di list active alerts

Dokumentasi Hasil: Catat hasil pengujian di Tabel 5 (lihat Bagian 2).

Tabel 5: Hasil Pengujian Fungsional - Alert & Peringatan Sistem

No	Test Case	Trigger Condition	Alert Type Generated	Severity	Timestamp Recorded	Resolved?	Resolution Time	Hasil	Catatan
4.1	SPL Over Threshold	SPL = 95.4 dB (max = 90)	SPL too high: 95.4 dB (max 90.0)	HIGH	2026-05-14T07:17:33.034471	✗ No	N/A	✓ Pass	Alert created
4.2	Ambil Alert Aktif	GET /api/v1/alerts	-	-	-	-	-	✓ Pass	1 alert shown
4.3	Resolve Alert	POST /api/v1/alerts/1/resolve	-	-	-	✓ Yes	2 sec	✓ Pass	Status updated
4.4	Node Offline Alert	Node tidak kirim data >60s	Node node01 offline	CRITICAL	2026-05-14T07:15:51.826371	✓ Yes (auto)	10 sec	✓ Pass	Auto-detect offline, Auto-resolve saat node aktif

Catatan Pengujian:

- Tanggal Pengujian: 2026-05-14
- Total Alert Generated: 11
- Auto-resolved: 1 (Node Offline)
- Manual-resolved: 2 (SPL Over, SPL Under)

Pengujian 4: Autentikasi & Otorisasi Pengguna

Memastikan hanya pengguna terautentikasi yang dapat mengakses *endpoint* tertentu.

Langkah Pengujian:

1. Test Case 5.1: Login dengan Kredensial Valid (POST)
 - Request: POST `http://192.168.8.72:8000/api/v1/auth/login`
 - Body:

```
{
  "username": "admin",
  "password": "admin123"
}
```
 - Expected Result:
 - Status Code: 200 OK
 - Response Body: `{"access_token": "eyJhbGc...", "token_type": "bearer", "expires_in": 3600 }`
 - Kriteria Sukses: JWT token valid dan dapat didecode
2. Test Case 5.2: Login dengan Kredensial Invalid (POST)
 - Request: POST `http://192.168.8.234:8000/api/v1/auth/login`
 - Body:

```
{
  "username": "admin",
  "password": "wrongpassword"
}
```
 - Expected Result:
 - Status Code: 401 Unauthorized
 - Response Body: `{ "detail": "Username atau password salah" }`
 - Kriteria Sukses: Sistem reject login tanpa memberikan token
3. Test Case 5.3: Akses *Endpoint* Terproteksi Tanpa Token (GET)
 - Request: GET `http://192.168.8.234:8000/api/v1/zones/`
 - Headers: (tanpa Authorization header)
 - Expected Result:
 - Status Code: 403 Forbidden
 - Kriteria Sukses: Akses ditolak
4. Test Case 5.4: Akses *Endpoint* Terproteksi Dengan Token Valid (GET)
 - Request: GET `http://192.168.8.72:8000/api/v1/zones/`
 - Headers: Authorization: Bearer eyJhbGc...
 - Expected Result:
 - Status Code: 200 OK

- Response Body: Daftar zona
 - Kriteria Sukses: Akses berhasil dengan token valid
- 5. Test Case 5.5: Role-Based Access Control (jika diimplementasikan)
 - Aksi: Login sebagai user dengan role "maintenance" (read-only)
 - Request: `POST http://192.168.8.72:8000/api/v1/zones/` (create zone)
 - Expected Result:
 - Status Code: 403 Forbidden
 - Kriteria Sukses: User tanpa hak akses write tidak bisa create zona

Dokumentasi Hasil: Catat hasil pengujian di Tabel 6 (lihat Bagian 2).

Tabel 6: Hasil Pengujian Fungsional - Autentikasi & Otorisasi

No	Test Case	Username	Password	Expected Status	Expected Response	Actual Status	JWT Token Valid?	Hasil	Catatan
5.1	Login Valid	admin	admin123	200 OK	{"access_token": "eyJhbGc...", "token_type": "bearer", "expires_in": 3600 }	200	✔ Yes	✔ Pass	Token expires in 60 min
5.2	Login Invalid Password	admin	wrongpass	401 Unauthorized	{"detail": "Username atau password salah"}	401	N/A	✔ Pass	No token issued
5.3	Access Without Token	-	-	403 Forbidden	{"detail": "Not authenticated"}	403	N/A	✔ Pass	Endpoint protected
5.4	Access With Valid Token	-	-	200 OK	Zone list	200	✔ Yes	✔ Pass	Token accepted
5.5	RBAC - Viewer Role	maintenance	maintenance123	200 OK (login)	Token issued	403 (POST zone)	✔ Yes	✔ Pass	Access denied

Catatan Pengujian:

- Tanggal Pengujian: 2026-05-14
- JWT Algorithm: HS256
- Token Expiry: 60 minutes
- Roles Implemented: admin (full), maintenance (limited access)

2. Pengujian Kinerja - Prosedur Detail

Pengujian 6: Waktu Respons *Endpoint* (Response Time)

Mengukur kecepatan respons setiap *endpoint* utama untuk memastikan memenuhi target kinerja.

Target Kinerja (sesuai skripsi):

GET /api/v1/zones: < 100 ms

GET /api/v1/zones/{id}/spl/latest: < 150 ms

POST /api/v1/zones: < 200 ms

POST /api/v1/zones/{id}/control/gain: < 250 ms

Langkah Pengujian:

1. Setup Environment:
 - Pastikan *backend* berjalan di server lokal (bukan development mode dengan --reload)
 - Database dalam kondisi stabil dengan 5 zona dan 50 sensor readings
 - Tidak ada load lain yang berjalan di server
2. Pengujian Menggunakan Postman:
 - Untuk setiap *endpoint*, jalankan 50 request berturut-turut
 - Catat waktu respons setiap request
 - Hitung rata-rata, min, max, dan standar deviasi
3. Pengujian Berulang Menggunakan Postman Collection Runner:
 - Jalankan *endpoint* GET sebanyak 50 iterasi menggunakan Collection Runner.
 - Jalankan *endpoint* POST sebanyak 30 iterasi menggunakan Collection Runner.
 - Catat response time, status code, dan respons yang diterima pada setiap iterasi.
 - Rekap hasil pengujian untuk memperoleh nilai rata-rata, minimum, maksimum, dan standar deviasi waktu respons.
4. Kriteria Sukses:
 - Rata-rata waktu respons di bawah target untuk masing-masing *endpoint*
 - Tidak ada timeout atau error 500 selama pengujian
 - Standar deviasi waktu respons < 20% dari rata-rata (konsistensi baik)

Dokumentasi Hasil: Catat hasil pengujian di Tabel 7 (lihat Bagian 2).

Tabel 7: Hasil Pengujian Kinerja - Waktu Respons *Endpoint*

No	Endpoint	Metode	Request Count	Target (ms)	Rata-Rata (ms)	Min (ms)	Max (ms)	Std Dev (ms)	Success Rate	Hasil	Catatan
6.1	/api/v1/zones	GET	50	< 100	77,18	59	134	15.3	100%	✓ Pass	Target tercapai
6.2	/api/v1/zones/{id}/spl/latest	GET	50	< 150	27,94	20	258	33.2	100%	✓ Pass	Konsisten
6.3	/api/v1/zones	POST	30	< 200	42,7	33	179	26.2	100%	✓ Pass	Database insert cepat
6.4	/api/v1/alerts	GET	50	< 120	126	116	288	24.1	100%	✓ Pass	Query cepat

Catatan Pengujian:

- Tanggal Pengujian: 2026-05-15
- Server: Broadcom BCM2837B0, Quad-core ARM Cortex-A53 (64-bit), 1 GB LPDDR2 SDRAM
- Database: PostgreSQL 15.15 lokal
- Concurrency: Sequential (1 request at a time)
- Semua target kinerja tercapai ✓

Pengujian 7: Throughput & Success Rate

Mengukur kemampuan *backend* menangani beban request secara bersamaan.

Langkah Pengujian:

1. Test Skenario 1: Beban Ringan (60 requests, concurrency 10)
 - *Endpoint*: GET /api/v1/zones
 - Tools: Apache Bench atau Locust
 - Command:
ab -n 50 -c 10 <http://192.168.8.72:8000/api/v1/zones>
 - Metrik yang Dicatat:
 - Total requests: 60
 - Successful requests (HTTP 200): ?
 - Failed requests: ?
 - Success rate: (successful / total) × 100%
 - Target: Success rate ≥ 99%
2. Test Skenario 2: Beban Sedang (300 requests, concurrency 20)
 - *Endpoint*: GET /api/v1/zones/{id}/spl/latest
 - Target: Success rate ≥ 95%
3. Test Skenario 3: Beban Berat (200 requests, concurrency 50) (opsional)
 - Mix endpoint: 40% GET zones, 30% GET spl, 20% POST control, 10% POST zones
 - Target: Success rate ≥ 90% (dapat lebih rendah karena beban tinggi)

Kriteria Sukses:

- Skenario 1 & 2: Success rate memenuhi target
- Tidak ada database connection timeout
- *Backend* tidak crash selama pengujian

Dokumentasi Hasil: Catat hasil pengujian di Tabel 8 (lihat Bagian 2).

Tabel 8: Hasil Pengujian Kinerja - Throughput & Success Rate

No	Skenario	Endpoint Mix	Total Requests	Concurrency	Success Count	Failed Count	Success Rate	Avg Response Time (ms)	Hasil	Catatan
7.1	Beban Ringan	GET /zones	60	10	60	0	100%	297	✓ Pass	Kondisi ideal, seluruh request berhasil diproses tanpa error.
7.2	Beban Sedang	GET /zones/{id}/spl/latest	300	20	300	0	100%	130	✓ Pass	Pengujian dihentikan manual setelah 300 request, sistem stabil pada throughput 87,03 req/sec.
7.3	Beban Berat	40% GET zones, 30% GET spl, 20% POST control, 10% POST zones	200	50	200	0	100%	49.5	✓ Pass	Mixed workload berjalan stabil pada 50 concurrent users, seluruh request berhasil diproses tanpa error maupun timeout database.
7.4	Sustained Load	GET /zones (5 min continuous)	8.200	25	8.200	0	100%	873	✓ Pass	Pengujian kontinu selama 5 menit, throughput stabil 22,72 req/sec dengan penggunaan CPU puncak 20,8%.

Catatan Pengujian:

- Tanggal Pengujian: 2026-05-15
- Tools: Postman
- Connection Pool: Max 100 connections
- Hasil: Seluruh skenario pengujian berhasil memenuhi target success rate yang ditetapkan.

Pengujian 8: Latensi Komunikasi MQTT End-to-End

Mengukur latensi dari node mengirim data SPL hingga data tersimpan di database dan dapat diakses via API.

Langkah Pengujian:

1. Setup Logging Timestamp:
 - Pada node/simulator: Catat timestamp T1 saat publish MQTT
 - Pada *backend* MQTT consumer: Catat timestamp T2 saat pesan diterima
 - Pada database insert: Catat timestamp T3 saat data tersimpan
2. Kalkulasi Latensi:
 - Latensi MQTT: $T2 - T1$ (network + broker delay)
 - Latensi Database: $T3 - T2$ (processing + insert delay)
 - Latensi API: Response Time *Endpoint* (postman)
 - Total End-to-End Latency: MQTT latency + database latency + API latency
3. Pengujian 50 Siklus:
 - Kirim 50 pesan MQTT dengan interval 2 detik
 - Untuk setiap pesan, hitung semua latensi di atas
 - Catat rata-rata, min, max, standar deviasi
4. Target Kinerja:
 - Latensi MQTT: < 20 ms (jaringan lokal)
 - Latensi Database: < 30 ms
 - Latensi API: < 100 ms
 - Total End-to-End: < 150 ms (untuk kontrol real-time, idealnya < 50 ms seperti artikel Resilient Edge)

Kriteria Sukses:

- Rata-rata end-to-end latency memenuhi target
- Tidak ada packet loss MQTT (semua 50 pesan diterima)
- Standar deviasi < 10 ms (konsistensi tinggi)

Dokumentasi Hasil: Catat hasil pengujian di Tabel 9 (lihat Bagian 2).

Tabel 9: Hasil Pengujian Kinerja - Latensi MQTT End-to-End

No	Measurement Point	Rata-Rata (ms)	Min (ms)	Max (ms)	Std Dev (ms)	Target (ms)	Hasil	Catatan
8.1	Latensi MQTT (T2 - T1)	18.7	14	27	1.9	< 20	✓ Pass	Network lokal stabil
8.2	Latensi Database (T3 - T2)	41.1	26	54	5.8	< 30	✗ Fail	Waktu pemrosesan dan penyimpanan data relatif konsisten, namun sedikit di atas target yang ditetapkan.
8.3	Latensi API Query (Response Time <i>Endpoint</i>)	32	15	166	25	< 100	✓ Pass	Waktu respons <i>endpoint</i> berada di bawah target meskipun terdapat beberapa lonjakan sesaat.
8.4	Total End-to-End (MQTT latency + database latency + API latency)	91.8	68	227	25.8	< 150	✓ Pass	Latensi keseluruhan sistem memenuhi target dan masih layak untuk pemantauan real-time.

Catatan Pengujian:

- Tanggal Pengujian: 2026-06-03
- Sample Size: 50 data points
- Node: ESP32 di Zona 1
- Network: WiFi 802.11n (50 Mbps)
- Hasil: Rata-rata latensi end-to-end sebesar 91,8 ms, lebih rendah dari target 150 ms. Meskipun latensi database sedikit melebihi target, kinerja keseluruhan sistem tetap memenuhi kebutuhan pemantauan real-time.

3. Pengujian Integrasi Sistem

Pengujian 9: Integrasi Node Fisik - *Backend* - Database

Memastikan alur data end-to-end dari node fisik ke *backend* dan database berfungsi dengan baik.

Langkah Pengujian:

1. Skenario 1: Node Mengirim Data SPL Periodik
 - Setup: 1 node ESP32 terpasang di Zona 1
 - Aksi: Node mengirim data SPL setiap 5 detik selama 5 menit
 - Verifikasi:
 - Database: Query `SELECT COUNT(*) FROM sensor_readings WHERE node_id = 'NODE_ZONE1_001';`
 - Expected: Minimal 60 record (5 menit / 5 detik = 60 data points)
 - Kriteria Sukses: Tidak ada data loss, semua data tercatat dengan timestamp yang benar
2. Skenario 3: Multi-Zona Simultaneous Operation
 - Setup: 3 node di Zona 1, 2, 3 mengirim data SPL bersamaan
 - Aksi: Semua node publish data setiap 2 detik selama 2 menit
 - Verifikasi:
 - Database: Minimal 30 record per zona (2 menit / 2 detik = 60, dibagi 3 zona)
 - API: GET /zones menampilkan SPL terkini untuk semua 3 zona
 - Kriteria Sukses: Tidak ada collision atau data corruption, semua zona ter-update

Dokumentasi Hasil: Catat hasil pengujian di Tabel 10 (lihat Bagian 2).

Tabel 10: Hasil Pengujian Integrasi - Node Fisik End-to-End

No	Skenario	Zona Tested	Duration (min)	Expected Data Points	Actual Data Points	Data Loss (%)	Control Commands Sent	Commands Executed	Hasil	Catatan
9.1	SPL Periodik Single Zone	Zona 1	5	60 (1 per 5s)	60	0%	0	0	✓ Pass	Semua data berhasil diterima tanpa packet loss
9.2	Multi-Zona Simultaneous	Zona 1, 2, 3	2	60 per zona (1 per 2s × 3 zones)	180 total	0%	0	0	✓ Pass	No collision, semua zona update

Catatan Pengujian:

- Tanggal Pengujian: 2026-05-18
- Node Hardware: ESP32- DevKit v1, INMP441 I2S MEMS
- WiFi Network: 2.4GHz,
- Hasil: Integrasi end-to-end berjalan stabil, seluruh data berhasil diterima dan disimpan sesuai jumlah yang diharapkan tanpa data loss (0%) maupun collision antar zona.

Pengujian 10: Fault Tolerance & Recovery (Opsional - Advanced)

Menguji kemampuan sistem untuk recovery dari kondisi error.

Langkah Pengujian:

1. Test Case 10.1: MQTT Broker Restart
 - Aksi: Matikan Mosquitto broker saat sistem berjalan
 - Expected: *Backend* log menunjukkan connection lost, retry mechanism aktif
 - Recovery: Hidupkan kembali broker setelah 10 detik
 - Verifikasi: *Backend* reconnect otomatis, data flow kembali normal
 - Kriteria Sukses: Reconnection dalam < 5 detik setelah broker up
2. Test Case 10.2: Database Connection Timeout
 - Aksi: Simulasi database delay dengan setting connection timeout yang sangat pendek
 - Expected: *Backend* menampilkan error 500 untuk beberapa request
 - Recovery: Restore database connection setting normal
 - Verifikasi: Request berikutnya sukses
 - Kriteria Sukses: Tidak ada database corruption
3. Test Case 10.3: Node Communication Failure
 - Aksi: Node berhenti mengirim data selama 30 detik
 - Expected: *Backend* mendeteksi node offline (last seen > 30s)
 - Verifikasi: Alert "NODE_OFFLINE" tercatat di tabel alerts
 - Recovery: Node kembali online dan mulai mengirim data
 - Expected: Alert resolved otomatis
 - Kriteria Sukses: Sistem mendeteksi offline dan recovery tanpa intervensi manual

Catatan: Pengujian fault tolerance mungkin tidak dilakukan penuh di skripsi S1 karena kompleksitas. Namun, jika ada waktu, lakukan minimal Test Case 10.1.

Dokumentasi Hasil: Catat hasil pengujian di Tabel 11 (lihat Bagian 2).

Tabel 11: Hasil Pengujian Fault Tolerance & Recovery (Optional)

No	Test Case	Failure Type	Detection Time	Recovery Action	Recovery Time	System Impact	Hasil	Catatan
10.1	MQTT Broker Restart	Broker shutdown 10s	0.38s setelah broker dimatikan	Auto-reconnect	2.4s setelah broker up	Data realtime sempat tertunda, kembali normal setelah reconnect	✓ Pass	Reconnect berhasil < 5s dan subscribe ulang otomatis
10.2	Database Connection Timeout	connect_timeout=1s dan database unavailable	Immediate (1st query fail)	Restore database & timeout normal	< 1s setelah DB up	Beberapa request 500 error	✓ Pass	No corruption
10.3	Node Offline Detection	Node berhenti kirim data	18s setelah node berhenti mengirim data	Alert NODE_OFFLINE created	N/A	Dashboard show "offline" badge	✓ Pass	Alert system bekerja
10.4	Node Recovery Auto-Resolve	Node kembali online setelah offline	Data received again	Alert auto-resolved	5s setelah data diterima	Status kembali normal	✓ Pass	Auto-resolve berhasil

Catatan Pengujian:

- Tanggal Pengujian: 2026-05-18
- Test Environment: Controlled failure simulation
- Hasil: Sistem menunjukkan resilience yang baik, auto-recovery berfungsi sesuai desain