Demeter - Backend Engineer PRD

Java Spring Boot API

Role: Backend Developer
Stack: Java 17 + Spring Boot 3 + PostgreSQL + Africa's Talking

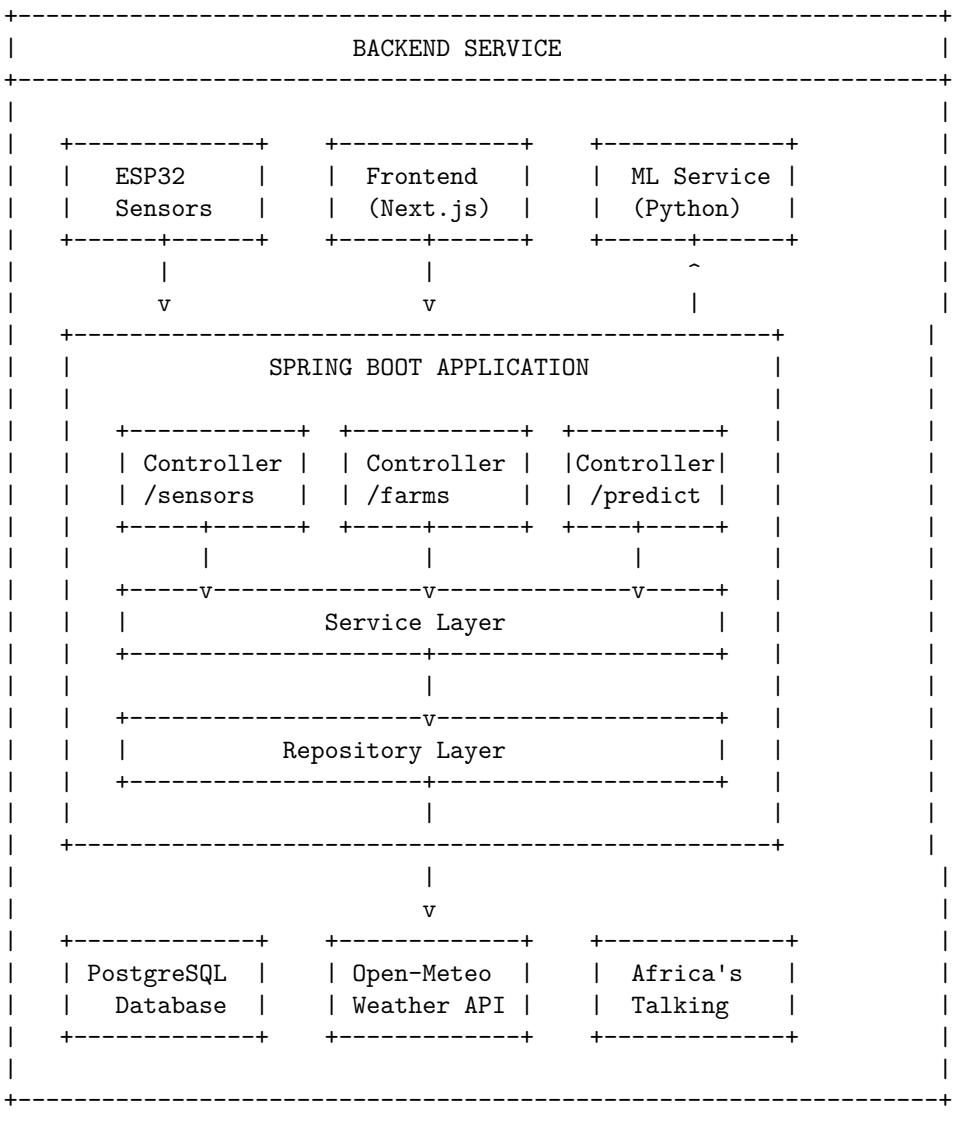
---

1. Overview

Build the REST API that serves as the central hub connecting sensors, ML service, frontend dashboard, and SMS notifications.

---

2. Architecture



---

3. API Endpoints

3.1 Farm Management

```
GET    /api/v1/farms                    # List all farms
POST   /api/v1/farms                    # Create farm
GET    /api/v1/farms/{id}               # Get farm by ID
PUT    /api/v1/farms/{id}               # Update farm
DELETE /api/v1/farms/{id}               # Delete farm
```

Farm Request/Response:

```
{
  "id": "uuid",
  "name": "Amina's Farm",
  "location": "Kaduna State, Nigeria",
  "latitude": 10.5105,
  "longitude": 7.4165,
  "size_hectares": 1.5,
  "crop_type": "MAIZE",
  "planting_date": "2026-02-01",
  "growth_stage": "VEGETATIVE",
  "owner_phone": "+2348012345678",
  "created_at": "2026-02-25T10:00:00Z"
}
```

3.2 Sensor Data

```
POST   /api/v1/sensor-data              # Ingest sensor reading
GET    /api/v1/farms/{farmId}/sensor-data/latest    # Get latest reading
GET    /api/v1/farms/{farmId}/sensor-data?from=&to= # Get historical data
```

Sensor Data Request:

```
{
  "farm_id": "uuid",
  "soil_moisture": 45.2,
  "temperature": 32.5,
  "humidity": 65.0,
  "timestamp": "2026-02-25T10:30:00Z"
}
```

3.3 Predictions (Proxy to ML Service)

```
GET    /api/v1/farms/{farmId}/prediction    # Get current prediction
```

Response:

```
{
  "farm_id": "uuid",
  "stress_index": 72,
  "risk_category": "SEVERE",
  "confidence": 0.85,
  "days_to_critical": 4,
  "recommendation": "Irrigate 25mm within 3 days",
  "forecast": [
    {"day": 1, "stress": 72},
    {"day": 2, "stress": 76},
    {"day": 3, "stress": 79},
    {"day": 4, "stress": 83},
    {"day": 5, "stress": 85},
```

```
    {"day": 6, "stress": 88},
    {"day": 7, "stress": 90}
  ],
  "generated_at": "2026-02-25T10:35:00Z"
}
```

3.4 Simulation (Proxy to ML Service)

```
POST   /api/v1/farms/{farmId}/simulate      # Run what-if simulation
```

Request:

```
{
  "scenario": "DRY_WEEK",
  "parameters": {
    "duration_days": 14,
    "rainfall_mm": 0
  }
}
```

Response:

```
{
  "baseline": {
    "stress_index": 45,
    "yield_impact": 0
  },
  "simulated": {
    "stress_index": 82,
    "yield_impact": -35
  },
  "forecast_comparison": [
    {"day": 1, "baseline": 45, "simulated": 52},
    {"day": 2, "baseline": 46, "simulated": 58}
  ],
  "recommendation": "Irrigate 25mm before day 3 to avoid critical stress"
}
```

3.5 SMS Alerts

```
POST   /api/v1/alerts/sms                   # Send SMS alert
GET    /api/v1/farms/{farmId}/alerts        # Get alert history
```

SMS Request:

```
{
  "farm_id": "uuid",
  "phone": "+2348012345678",
  "language": "en"
}
```

SMS Response:

```
{
  "success": true,
  "message_id": "ATXid_123456",
  "message": "Alert sent successfully"
}
```

3.6 Weather Data

```
GET    /api/v1/farms/{farmId}/weather       # Get weather forecast
```

4. Database Schema

```sql
-- Farms table
CREATE TABLE farms (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(255) NOT NULL,
    location VARCHAR(255),
    latitude DECIMAL(10, 8),
    longitude DECIMAL(11, 8),
    size_hectares DECIMAL(10, 2),
    crop_type VARCHAR(50) DEFAULT 'MAIZE',
    planting_date DATE,
    growth_stage VARCHAR(50),
    owner_phone VARCHAR(20),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Sensor data table
CREATE TABLE sensor_data (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    farm_id UUID REFERENCES farms(id) ON DELETE CASCADE,
    soil_moisture DECIMAL(5, 2),
    temperature DECIMAL(5, 2),
    humidity DECIMAL(5, 2),
    timestamp TIMESTAMP NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_sensor_data_farm_timestamp ON sensor_data(farm_id, timestamp DESC);

-- Predictions cache table
CREATE TABLE predictions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    farm_id UUID REFERENCES farms(id) ON DELETE CASCADE,
    stress_index INTEGER,
    risk_category VARCHAR(20),
    confidence DECIMAL(3, 2),
    days_to_critical INTEGER,
    recommendation TEXT,
    forecast JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Simulations history table
CREATE TABLE simulations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    farm_id UUID REFERENCES farms(id) ON DELETE CASCADE,
    scenario_type VARCHAR(50),
    parameters JSONB,
    baseline JSONB,
    result JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```sql
-- Alerts history table
CREATE TABLE alerts (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    farm_id UUID REFERENCES farms(id) ON DELETE CASCADE,
    alert_type VARCHAR(50),
    phone VARCHAR(20),
    message TEXT,
    message_id VARCHAR(100),
    status VARCHAR(20),
    sent_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

---

5. External Service Integration

5.1 ML Service (Python FastAPI)

```java
@Service
public class MLServiceClient {

    private final WebClient webClient;

    public MLServiceClient(@Value("${ml.service.url}") String mlServiceUrl) {
        this.webClient = WebClient.builder()
            .baseUrl(mlServiceUrl)
            .build();
    }

    public Mono<PredictionResponse> getPrediction(PredictionRequest request) {
        return webClient.post()
            .uri("/predict")
            .bodyValue(request)
            .retrieve()
            .bodyToMono(PredictionResponse.class);
    }

    public Mono<SimulationResponse> runSimulation(SimulationRequest request) {
        return webClient.post()
            .uri("/simulate")
            .bodyValue(request)
            .retrieve()
            .bodyToMono(SimulationResponse.class);
    }
}
```

5.2 Open-Meteo Weather API

```java
@Service
public class WeatherService {

    private static final String OPEN_METEO_URL = "https://api.open-meteo.com/v1/forecast";

    public WeatherForecast getForecast(double latitude, double longitude) {
        String url = String.format(
            "%s?latitude=%f&longitude=%f&daily=temperature_2m_max,temperature_2m_min,precipitation_sum&time
            OPEN_METEO_URL, latitude, longitude
        );
```

```
        // Call API and map response
        return webClient.get()
            .uri(url)
            .retrieve()
            .bodyToMono(OpenMeteoResponse.class)
            .map(this::mapToWeatherForecast)
            .block();
    }
}
```

5.3 Africa's Talking SMS

```
@Service
public class SMSService {

    private final AfricasTalkingGateway gateway;

    public SMSService(
        @Value("${africastalking.username}") String username,
        @Value("${africastalking.apikey}") String apiKey
    ) {
        this.gateway = new AfricasTalkingGateway(username, apiKey);
    }

    public SMSResponse sendAlert(String phone, String message) {
        try {
            JSONArray response = gateway.sendMessage(phone, message, null);
            return new SMSResponse(true, response.getJSONObject(0).getString("messageId"));
        } catch (Exception e) {
            return new SMSResponse(false, e.getMessage());
        }
    }

    public String buildAlertMessage(Farm farm, Prediction prediction, String language) {
        if ("ha".equals(language)) {
            return String.format(
                "DEMETER\nGona: %s\nHadari: %s (%d/100)\nAiki: %s",
                farm.getName(),
                translateRiskCategory(prediction.getRiskCategory(), "ha"),
                prediction.getStressIndex(),
                prediction.getRecommendation()
            );
        }
        return String.format(
            "DEMETER ALERT\nFarm: %s\nRisk: %s (%d/100)\nAction: %s",
            farm.getName(),
            prediction.getRiskCategory(),
            prediction.getStressIndex(),
            prediction.getRecommendation()
        );
    }
}
```

---

6. Project Structure

```
backend/
||| src/main/java/com/demeter/
|   ||| DemeterApplication.java
|   ||| config/
|   |   ||| WebConfig.java
|   |   ||| SecurityConfig.java
|   ||| controller/
|   |   ||| FarmController.java
|   |   ||| SensorDataController.java
|   |   ||| PredictionController.java
|   |   ||| SimulationController.java
|   |   ||| AlertController.java
|   ||| service/
|   |   ||| FarmService.java
|   |   ||| SensorDataService.java
|   |   ||| PredictionService.java
|   |   ||| SimulationService.java
|   |   ||| AlertService.java
|   |   ||| MLServiceClient.java
|   |   ||| WeatherService.java
|   |   ||| SMSService.java
|   ||| repository/
|   |   ||| FarmRepository.java
|   |   ||| SensorDataRepository.java
|   |   ||| PredictionRepository.java
|   |   ||| SimulationRepository.java
|   |   ||| AlertRepository.java
|   ||| model/
|   |   ||| entity/
|   |   |   ||| Farm.java
|   |   |   ||| SensorData.java
|   |   |   ||| Prediction.java
|   |   |   ||| Simulation.java
|   |   |   ||| Alert.java
|   |   ||| dto/
|   |   |   ||| FarmDTO.java
|   |   |   ||| SensorDataDTO.java
|   |   |   ||| PredictionDTO.java
|   |   |   ||| SimulationRequestDTO.java
|   |   |   ||| SimulationResponseDTO.java
|   |   |   ||| AlertDTO.java
|   |   ||| enums/
|   |       ||| CropType.java
|   |       ||| GrowthStage.java
|   |       ||| RiskCategory.java
|   |       ||| ScenarioType.java
|   ||| exception/
|       ||| GlobalExceptionHandler.java
|       ||| ResourceNotFoundException.java
||| src/main/resources/
|   ||| application.yml
|   ||| application-dev.yml
|   ||| db/migration/
|       ||| V1__initial_schema.sql
||| src/test/java/
```

```
||| pom.xml
||| Dockerfile
```

---

7. Configuration

application.yml

```yaml
server:
  port: 8080

spring:
  datasource:
    url: jdbc:postgresql://localhost:5432/demeter
    username: ${DB_USERNAME:demeter}
    password: ${DB_PASSWORD:demeter}
  jpa:
    hibernate:
      ddl-auto: validate
    show-sql: false

ml:
  service:
    url: ${ML_SERVICE_URL:http://localhost:8000}

africastalking:
  username: ${AT_USERNAME:sandbox}
  apikey: ${AT_API_KEY}

cors:
  allowed-origins: http://localhost:3000
```

---

8. Dependencies (pom.xml)

```xml
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-webflux</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>
    <dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <scope>runtime</scope>
    </dependency>
```

```
    <dependency>
        <groupId>org.flywaydb</groupId>
        <artifactId>flyway-core</artifactId>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>com.africastalking</groupId>
        <artifactId>core</artifactId>
        <version>3.4.11</version>
    </dependency>
</dependencies>
```

---

9. Enums

```
public enum GrowthStage {
    EMERGENCE(0.3),
    VEGETATIVE(0.7),
    TASSELING(1.2),
    GRAIN_FILL(1.0),
    MATURITY(0.6);

    private final double cropCoefficient;

    GrowthStage(double kc) {
        this.cropCoefficient = kc;
    }

    public double getCropCoefficient() {
        return cropCoefficient;
    }
}

public enum RiskCategory {
    NONE(0, 20),
    LOW(21, 40),
    MODERATE(41, 60),
    SEVERE(61, 80),
    CRITICAL(81, 100);

    private final int minStress;
    private final int maxStress;

    public static RiskCategory fromStressIndex(int stress) {
        for (RiskCategory cat : values()) {
            if (stress >= cat.minStress && stress <= cat.maxStress) {
                return cat;
            }
        }
        return CRITICAL;
    }
}
```

```
public enum ScenarioType {
    DRY_WEEK,
    DELAYED_PLANTING,
    IRRIGATION_TEST,
    CUSTOM
}
```

---

10. Deliverables Checklist

- ☐ Project setup (Spring Boot + PostgreSQL)
- ☐ Database schema with Flyway migrations
- ☐ Farm CRUD endpoints
- ☐ Sensor data ingestion endpoint
- ☐ ML Service client integration
- ☐ Prediction proxy endpoint
- ☐ Simulation proxy endpoint
- ☐ Weather service (Open-Meteo)
- ☐ Africa's Talking SMS integration
- ☐ Alert history storage
- ☐ CORS configuration
- ☐ Error handling
- ☐ API documentation (OpenAPI/Swagger)
- ☐ Docker setup

---

11. Environment Variables

```
DB_USERNAME=demeter
DB_PASSWORD=demeter
DB_URL=jdbc:postgresql://localhost:5432/demeter
ML_SERVICE_URL=http://localhost:8000
AT_USERNAME=sandbox
AT_API_KEY=your_api_key_here
```

Coordinate with: Frontend (API contracts), AI Engineer (ML service protocol), Hardware (sensor data format)