
📘 Equipment Management — Project Documentation

---

1. Name and Project Title

Name: Onkar Vallal
Project Title: Equipment Management & Allocation System (Play + Kafka + Akka)

---

2. Problem Statement

Organizations frequently handle employee equipment such as laptops, monitors, headsets, security devices, IT assets, and temporary peripherals. Manual equipment tracking leads to:

- Missing or unreturned equipment
- No visibility into allocation history
- Delayed IT notifications
- No automated process for reporting damage or requesting replacement
- Lack of audit trails for compliance

This project solves these issues by providing:

✔ A digital system to manage equipment inventory
✔ Allocation, return, and damage tracking
✔ Kafka-based notification publishing
✔ Akka consumer service to process equipment events (allocation, return, damage)
✔ Email/IT workflow automation
✔ Complete audit logging and event-driven workflows

---

3. Solution & Approach

✔ Solution Overview

The system is split into **two independent services**:

1️⃣ Play Framework Backend — (`equip-mgnt-playApp`)

- Exposes REST APIs for:
    - Equipment
    - Employee
    - Allocation & Return
    - Damage logging
- Stores data in a database via Slick or repository classes
- Publishes equipment events to Kafka topic: `equipment_events`
- Decouples side effects (emails, logs) from main business logic

2️⃣ Akka Kafka Consumer — (`akka-kafka-consumer-mgnt`)

- Consumes events from `equipment_events` topic using Alpakka Kafka
- Akka Typed actors process each event type:
    - `EquipmentAllocatedActor` — notify employee & IT
    - `EquipmentReturnedActor` — update store manager / IT
    - `EquipmentDamagedActor` — alert repair/IT asset management
- Email workflow via `EmailService.scala`
- Event logs printed for audit and debugging

---

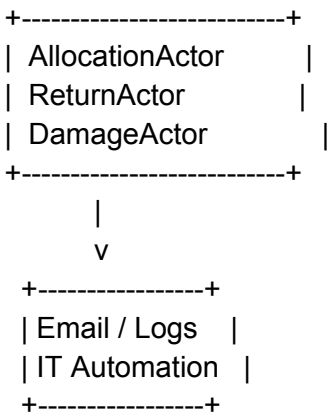
4. Diagrams Supporting Design Approach & Flow

Event Flow / Sequence Diagram

```
User (Admin / IT Portal / UI)
        |
        | POST /equipment/allocate
        |--------------------------------------------->
Play Backend (equip-mgnt-playApp)
        | 1. Validate request
        | 2. Update DB (equipment allocated)
        | 3. Create AllocationLog
        | 4. Publish JSON event -> Kafka topic `equipment_events`
        |--------------------------------------------->
Kafka Broker (topic: equipment_events)
        |--------------------------------------------->
Akka Consumer Service (akka-kafka-consumer-mgnt)
        | 1. Consume message
        | 2. Parse event type: ALLOCATED / RETURNED / DAMAGED
        | 3. Forward to relevant actor:
        |      - EquipmentAllocatedActor
        |      - EquipmentReturnedActor
        |      - EquipmentDamagedActor
        | 4. Each actor performs:
        |      - Email notification
        |      - IT action
        |      - Console / DB logging
```

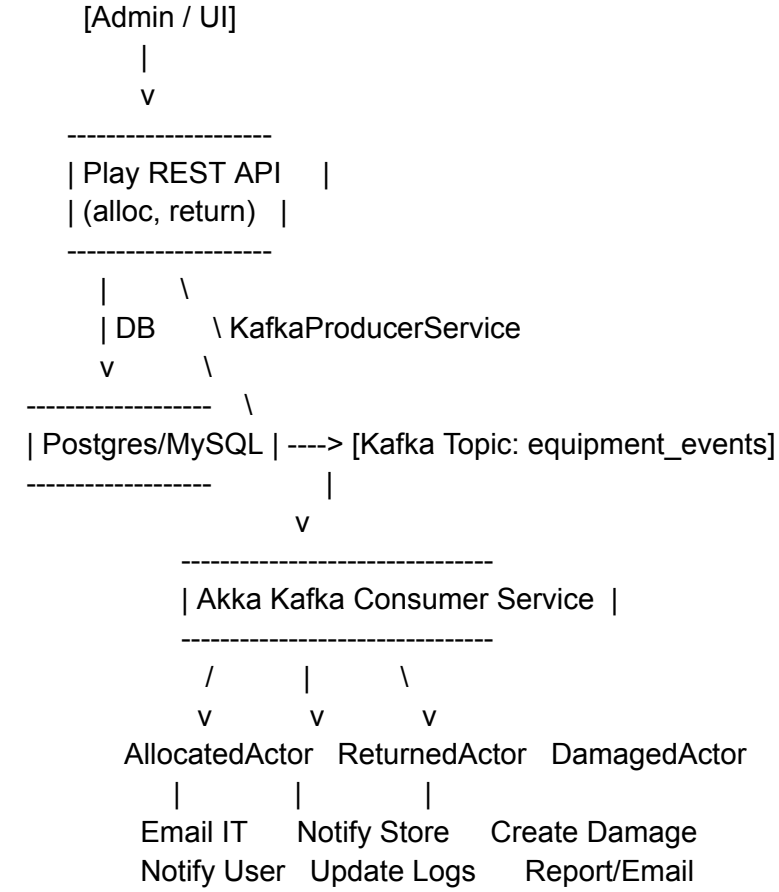
Component Diagram (ASCII)

```
+---------------+       +------------------------+       +---------------------+
| Employee/Admin | ----> | Play Backend           | ----> | Kafka Broker        |
| UI or REST     |       | (Scala + Play)         |       | (equipment_events)  |
+---------------+       +------------------------+       +---------------------+
                              |
                              v
                    +--------------------------+
                    | Akka Consumer Service    |
                    | (Scala + Akka Typed)     |
```



---

5. Overall Architecture Diagram



---

6. Deployment Plan

🔧 Prerequisites

- Kafka & Zookeeper (or Kafka KRaft mode)
- Database (Postgres/MySQL)
- SMTP credentials for sending email
- JDK 11+
- sbt installed

Step 1 — Start Kafka and Create Topic

bin/zookeeper-server-start.sh config/zookeeper.properties &
bin/kafka-server-start.sh config/server.properties &

```
bin/kafka-topics.sh --create \
  --topic equipment_events \
  --bootstrap-server localhost:9092 \
  --replication-factor 1 \
  --partitions 1
```

Step 2 — Setup DB

- Create DB
- Run initial schema/table creation (based on DAOs in Play app)

Step 3 — Build & Run Play Equipment Management App

```
cd equip-mgnt-playApp
sbt run
# OR build package
sbt dist
```

The app runs on **http://localhost:9000**

Step 4 — Build & Run Akka Consumer

```
cd akka-kafka-consumer-mgnt
sbt run
```

You will see logs like:

```
📥 Received Kafka Event: EQUIPMENT_ALLOCATED
[AllocatedActor] Email sent to employee...
```

Step 5 — Test Flow using curl

1. Allocate Equipment

```
curl -X POST http://localhost:9000/equipment/allocate \
  -H "Content-Type: application/json" \
  -d '{
        "equipmentId": 10,
        "employeeId": 4,
        "notes": "For new project onboarding"
      }'
```

Expected response:

```
{
  "status": "success",
  "message": "Equipment allocated successfully"
}
```

Production Deployment Suggestions

- Use **Docker containers** for both services
- Host Kafka on **Confluent Cloud / MSK**
- Use HTTPS + Nginx
- Move SMTP credentials to secrets manager

- Add CI/CD pipeline
- Add Akka monitoring (Lightbend Telemetry, Prometheus)

---

7. Screenshots of Output

Screenshot 1 — Successful Equipment Allocation (in terminal)

POST /equipment/allocate
Status: 200 OK

```
{
  "status": "success",
  "equipmentId": 10,
  "employeeId": 4
}
```

Screenshot 2 — Akka Service Logs after consuming Kafka event

📥 Kafka Message Received:
{"eventType":"EQUIPMENT_ALLOCATED","equipment":{"id":10,...}}

[EquipmentAllocatedActor] Sending allocation email to employee@example.com
[EquipmentAllocatedActor] Logging allocation event...

Screenshot 3 — Damage report event

Kafka Event: EQUIPMENT_DAMAGED
[EquipmentDamagedActor] Notifying IT repair team
[EmailService] sent email to support@example.com

You can take these while running the system locally.

---

8. Features of Scala / Akka / Play / Kafka Implemented

🚀 Play Framework Features

- REST API controllers (`EquipmentController`, `EquipmentAllocationController`)
- Dependency injection via Guice
- JSON Reads/Writes using Play JSON
- DB integration using repositories (Slick/DAO style)
- Routing via `conf/routes`
- Error handling using Action composition

⚙ Scala Features Used

- Case classes for domain models
- Pattern matching
- Options, Futures, functional transformations
- Strong type safety
- Companion objects, implicit JSON formats

🛰 Apache Kafka Features

- KafkaProducerService publishes events to topic `equipment_events`
- JSON event structure:
    - `EQUIPMENT_ALLOCATED`
    - `EQUIPMENT_RETURNED`
    - `EQUIPMENT_DAMAGED`

- Asynchronous decoupled communication

🎭 Akka Features

- **Akka Typed Actors**:
    - Modular actors for each event type
    - Supervisor / root behavior
- **Akka Streams + Alpakka Kafka** for consumption
- **Fault tolerance** via actor model
- **Email service** integrated inside actors

Main actors in repo:

- `EquipmentAllocatedActor.scala`
- `EquipmentReturnedActor.scala`
- `EquipmentDamagedActor.scala`
- `KafkaEquipmentConsumer.scala`
- `EmailService.scala`

Appendix — Important Code Locations

Play App (`equip-mgnt-playApp`)

- `controllers/EquipmentController.scala`

- controllers/EquipmentAllocationController.scala
- services/KafkaProducerService.scala
- models/*.scala
- conf/routes

Akka App (`akka-kafka-consumer-mgnt`)

- kafka/EquipmentKafkaConsumer.scala
- actors/EquipmentAllocatedActor.scala
- actors/EquipmentReturnedActor.scala
- actors/EquipmentDamagedActor.scala
- EmailService.scala
