

Corporate Equipment Allocation and Tracking System

Owner: Sanjeev Kumar V

Date: Nov 22, 2025

The Problem

In a busy corporate office environment, the management of shared equipment—such as laptops, projectors, and tablets—presents a significant logistical challenge. The current reliance on manual tracking methods, like spreadsheets or physical logbooks, is inefficient and prone to human error.

This leads to several key problems:

- **Poor Availability Tracking:** Staff cannot easily determine what equipment is available or when a currently allocated item is scheduled to be returned.
- **Lack of Accountability:** There is no formal process for tracking the condition of equipment, leading to items being returned damaged without a clear maintenance flag. This also contributes to equipment loss.
- **Scheduling Conflicts:** Without an automated system to enforce return dates, employees often return items late, causing cascading scheduling conflicts for other users.
- **Siloed Information:** The Inventory and Maintenance teams are disconnected from the daily allocation process. They are not automatically updated when an item's status changes (e.g., "allocated," "returned," "damaged").
- Inefficient equipment distribution
- Missing or lost devices due to lack of audits
- No real-time view of availability
- No automated follow-up for overdue returns
- No proper maintenance workflow
- Lack of notifications to teams responsible for inventory and repairs

1.2. The Solution & System Goals

To solve these challenges, we are proposing a centralized, automated **Corporate Equipment Allocation and Tracking System**. This system will streamline the entire lifecycle of equipment management, from allocation to return and maintenance.

The primary goals of this solution are:

- **Goal 1: Simplify Allocation:** Empower reception staff to easily allocate and log equipment returns in seconds through a simple user interface.
- **Goal 2: Ensure Accountability:** Track every item's lifecycle, logging who has what, when it's due back, and its condition upon return.

- **Goal 3: Automate Communication:** Proactively notify all relevant stakeholders (Employees, Maintenance, Inventory) in real-time about equipment status.
- **Goal 4: Build a Resilient System:** Implement a modern, scalable, and fault-tolerant backend architecture that can handle failures gracefully and scale as the company grows.

The **Corporate Equipment Allocation and Tracking System** provides an automated, centralized backend that manages the full lifecycle of equipment assets.

1. Equipment Allocation & Return Workflow

- Reception staff assigns equipment to employees.
- The system checks availability before allocating.
- Return workflow logs condition and creates maintenance alerts for damaged items.

2. Automated Notifications using Kafka

- Overdue reminders sent automatically.
- Maintenance alerts for damaged items.
- Inventory updates when items are allocated/returned.
- Kafka ensures reliable event-driven communication.

3. Role-Based Access Control (RBAC)

- Admin
- Reception Staff
- Inventory Team
- Maintenance Team

Each role has separate access privileges.

4. Akka/Pekko Scheduler for Automated Overdue Checks

- Periodically scans for overdue items.
- Publishes events to Kafka or sends notifications.

5. Play Framework REST API Backend

- Fully RESTful endpoints for allocation, returns, maintenance, users, inventory, and notifications.

Core Features

The system's functionality is broken down into the following core features:

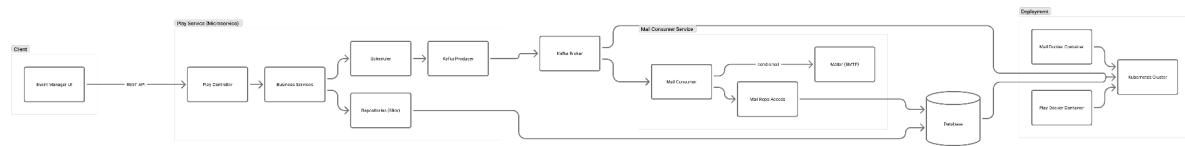
Feature	Description
Equipment Allocation	Reception staff logs employee details (name, dept, purpose) and expected return date.
Equipment Return	Staff logs the return, verifies the item's condition, and flags it for maintenance if needed.
Overdue Reminders	Automated: Sends email reminders to employees who have not returned equipment on time.
Maintenance Alerts	Automated: Notifies the maintenance team instantly if an item is returned damaged or due for service.
Inventory Sync	Automated: Notifies the inventory team on every allocation and return to keep availability status up-to-date.

User Roles & Stakeholders

The system will serve several key groups:

- **Reception Staff (Primary User):** Manages the day-to-day allocation and return process via the system's web interface.
- **Employees (End User):** The individuals who borrow equipment. Their primary interaction is receiving automated email notifications (e.g., overdue reminders).
- **Maintenance Team (Stakeholder):** A downstream consumer of the system. They receive automated alerts to schedule repairs for equipment flagged as damaged.
- **Inventory Team (Stakeholder):** A downstream consumer of the system. They receive real-time data to update master inventory lists and availability status.

System Architecture



4.1. Architectural Overview

We will implement a **Reactive Microservices Architecture** to separate concerns and ensure high availability and resilience. This design decouples the core request-handling logic from asynchronous background tasks and notifications.

The architecture consists of four primary components:

1. **Equipment Service (Play Framework):** The core of the system and the main "front door." This service exposes a REST API that handles all synchronous requests from the reception staff's web application (e.g., "Allocate this item," "Return this item").
2. **Notification Service (Akka Microservice):** A separate, asynchronous service. Its sole responsibility is to handle scheduled, internal tasks. For this project, it manages checking for and sending overdue reminders.
3. **Message Broker (Kafka):** This acts as the "central nervous system" for all event-driven, asynchronous communication. It decouples the Equipment Service from all other teams and services.
4. **Downstream Consumers:** These are independent services that listen for events from Kafka. In this system, they represent the Maintenance and Inventory teams, who consume alerts and updates.

Equipment Service (Play Framework)

- **Role:** The main "front door" of our system. It handles all direct user interactions.
- **Technology:** Play Framework (Java/Scala)
- **Justification:** The Play Framework is ideal for this role. It is built on Akka, making it non-blocking and asynchronous by default, which allows it to handle thousands of concurrent requests efficiently. Its REST-first design and hot-reloading capabilities also allow for rapid development and clean, stateless APIs.

Notification Service (Akka Actors)

- **Role:** The "background worker" for internal, stateful, and scheduled tasks.
- **Technology:** Akka Actors (in a standalone microservice)
- **Justification:** The Actor Model is perfect for our overdue reminder requirement. We can create a lightweight `OverdueSchedulerActor` with a built-in `Scheduler` to wake up periodically (e.g., once an hour). This actor then sends a message to a `ReminderWorkerActor` to perform the work

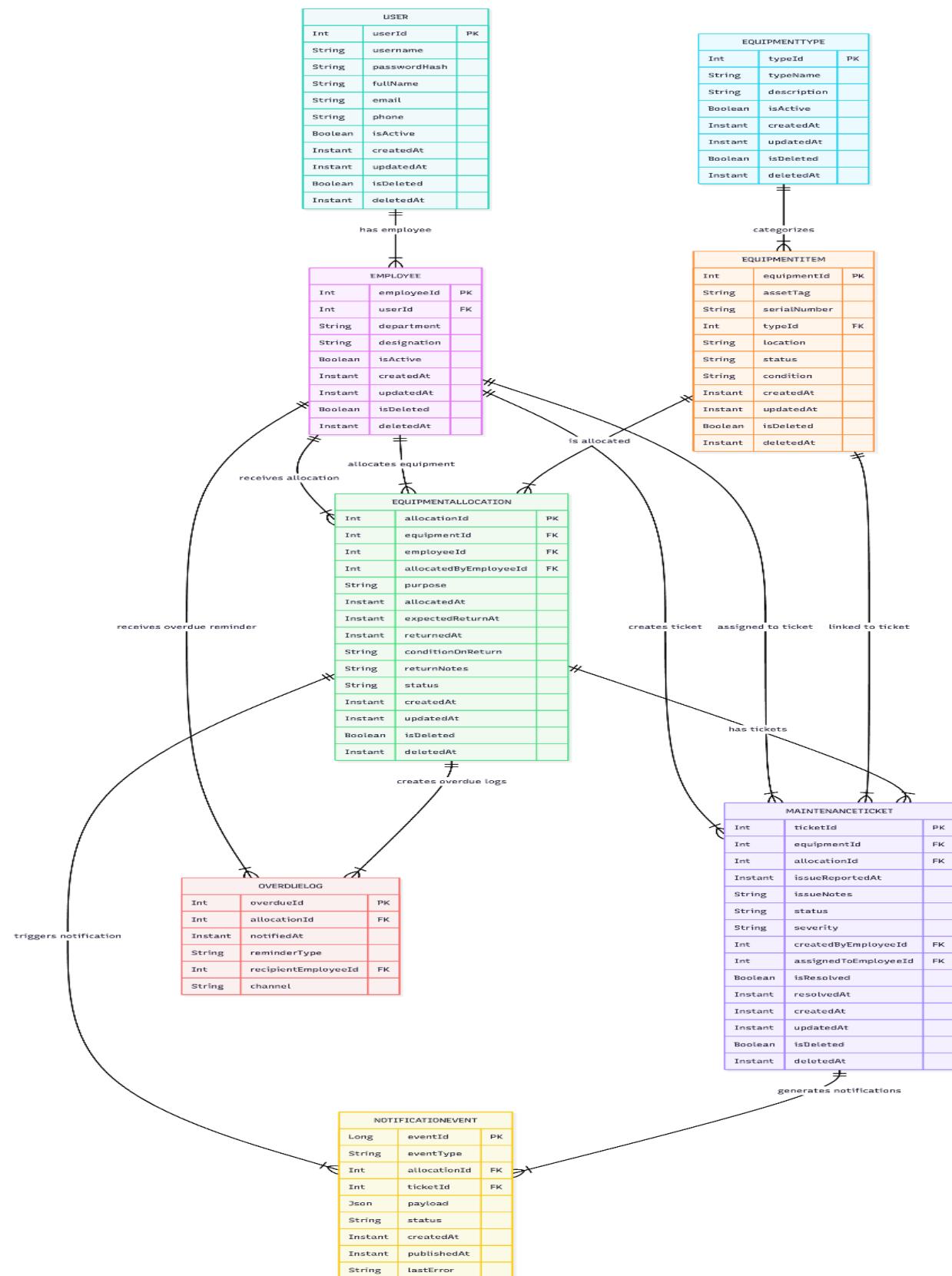
(querying the database and sending emails). This isolates the task and, by using Akka's supervision strategies, creates a self-healing, fault-tolerant system.

Apache Kafka (Message Queue)

Kafka acts as a durable, high-throughput buffer for all inter-service communication.

- **Decoupling:** Play doesn't need to know how notifications are sent; it just sends the data to Akka, which uses Kafka.
- **Reliable Delivery:** If the final Notification Dispatcher fails, the messages remain in the Kafka topic, ensuring delivery once the dispatcher recovers.
- **Notification Queueing:** All reminders and alerts are queued, smoothing out traffic spikes and preventing the Akka service from blocking on external HTTP calls (Email/SMS gateways).
- **Usage Example:**
 - **Topic:** equipment-event
 - **Allocation Message:** When an item is allocated, the Play service publishes: {
 "event_type": "CREATE", "item_id": "laptop-123", "user": "..." }
 - **Damage Message:** When an item is returned damaged, the Play service publishes: {
 "event_type": "DAMAGE", "item_id": "proj-456", "notes": "..." }

Database:



Sample Data:

Use:

```
{  
  "userId": 1,  
  "username": "john.doe",  
  "passwordHash": "$2b$12$xxxxhashedpasswordxxxx",  
  "fullName": "John Doe",  
  "email": "john.doe@example.com",  
  "phone": "9876543210",  
  "isActive": true,  
  "createdAt": "2025-01-10T10:00:00Z",  
  "updatedAt": "2025-01-10T10:00:00Z",  
  "roles": ["EMPLOYEE"],  
  "isDeleted": false,  
  "deletedAt": null  
}
```

Employee:

```
{  
  "employeeId": 101,  
  "userId": 1,  
  "department": "IT",  
  "designation": "Technician",  
  "isActive": true,  
  "createdAt": "2025-01-10T10:05:00Z",  
  "updatedAt": "2025-01-10T10:05:00Z",  
  "isDeleted": false,  
  "deletedAt": null  
}
```

EquipmentType:

```
{  
  "typeId": 10,  
  "typeName": "Laptop",  
  "description": "Portable computer devices",  
  "status": "Active",  
  "createdBy": "John Doe",  
  "modifiedBy": "Jane Doe",  
  "lastModified": "2025-01-10T10:00:00Z",  
  "deletedAt": null  
}
```

```
"isActive": true,  
"createdAt": "2025-01-12T09:00:00Z",  
"updatedAt": "2025-01-12T09:00:00Z",  
"isDeleted": false,  
"deletedAt": null  
}
```

EquipmentItem:

```
{  
    "equipmentId": 501,  
    "assetTag": "ASSET-LAP-001",  
    "serialNumber": "SN12345LAP",  
    "typeId": 10,  
    "location": "Storage Room A",  
    "status": "AVAILABLE",  
    "condition": "GOOD",  
    "createdAt": "2025-01-12T10:00:00Z",  
    "updatedAt": "2025-01-12T10:00:00Z",  
    "isDeleted": false,  
    "deletedAt": null  
}
```

EquipmentAllocation:

```
{  
    "allocationId": 3001,  
    "equipmentId": 501,  
    "employeeId": 101,  
    "allocatedByEmployeeId": 102,  
    "purpose": "For remote project work",  
    "allocatedAt": "2025-01-15T09:30:00Z",  
    "expectedReturnAt": "2025-02-15T09:30:00Z",  
    "returnedAt": null,  
    "conditionOnReturn": null,  
    "returnNotes": null,  
    "status": "ACTIVE",  
    "createdAt": "2025-01-15T09:30:00Z",  
    "updatedAt": "2025-01-15T09:30:00Z",  
    "isDeleted": false,  
}
```

```
    "deletedAt": null  
}
```

MaintenanceTicket:

```
{  
    "ticketId": 9001,  
    "equipmentId": 501,  
    "allocationId": 3001,  
    "issueReportedAt": "2025-01-20T14:00:00Z",  
    "issueNotes": "Laptop overheating frequently.",  
    "status": "OPEN",  
    "severity": "HIGH",  
    "createdByEmployeeId": 101,  
    "assignedToEmployeeId": 103,  
    "isResolved": false,  
    "resolvedAt": null,  
    "createdAt": "2025-01-20T14:00:00Z",  
    "updatedAt": "2025-01-20T14:00:00Z",  
    "isDeleted": false,  
    "deletedAt": null  
}
```

OverdueLog:

```
{  
    "overdueId": 1,  
    "allocationId": 3001,  
    "notifiedAt": "2025-02-16T09:00:00Z",  
    "reminderType": "FIRST",  
    "recipientEmployeeId": 101,  
    "channel": "EMAIL"  
}
```

Deployment Plan:

Environment Setup:

Component	Technology
Backend	Scala + Play
Actor System	Akka
Event Bus	Kafka
Database	MySQL
Build Tool	SBT
Deployment Container	Docker

Steps

1. Build the application

sbt clean compile stage

2. Containerize

Dockerfile

docker build -t event-coordination.

3. Deploy Kafka

Using docker-compose or managed Kafka.

docker-compose up -d kafka zookeeper

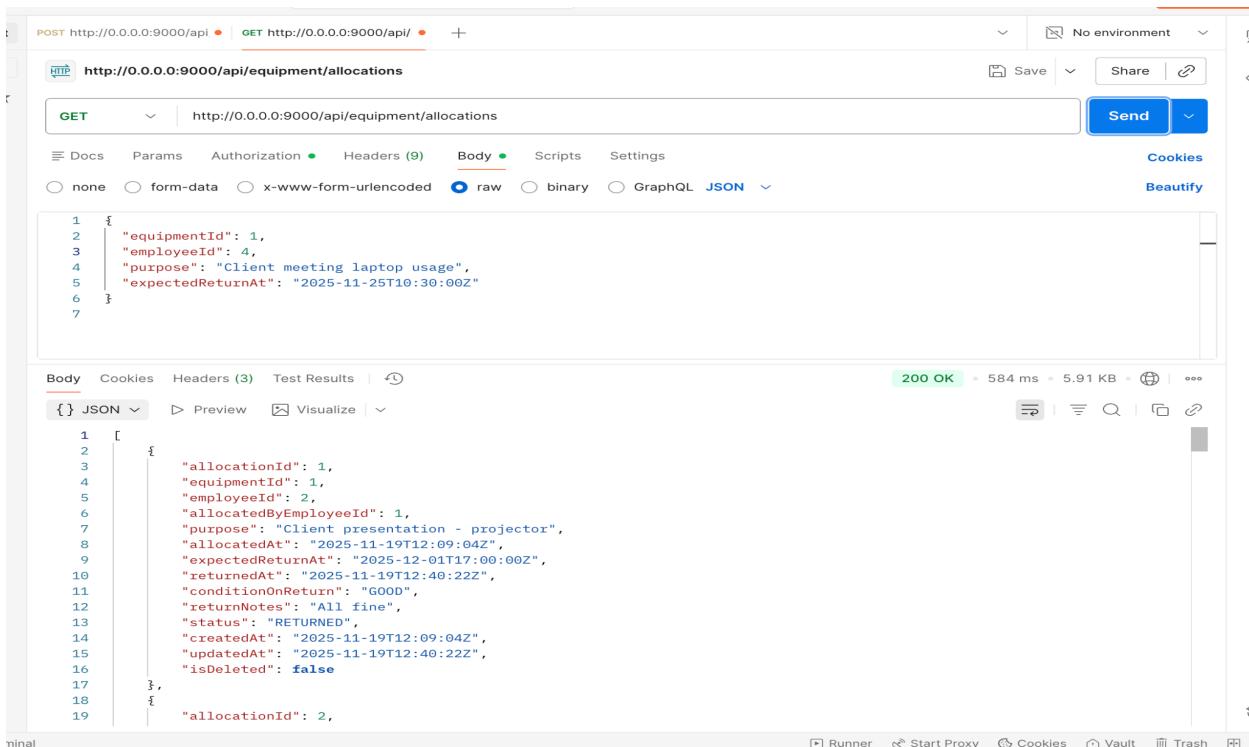
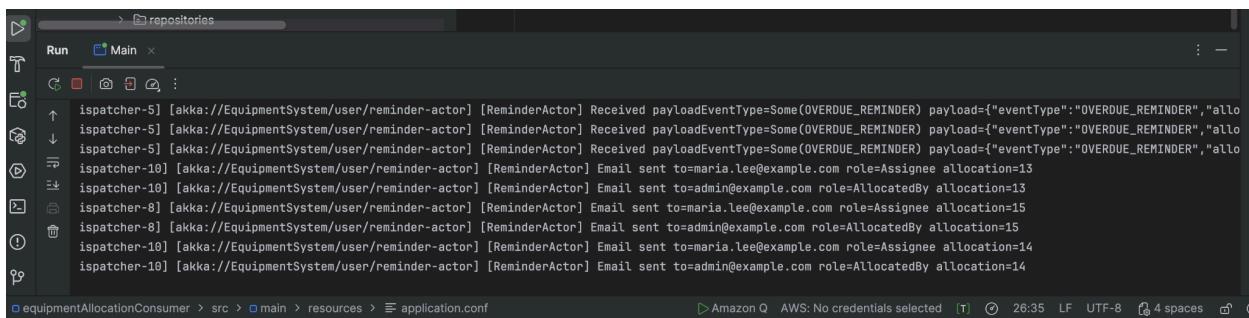
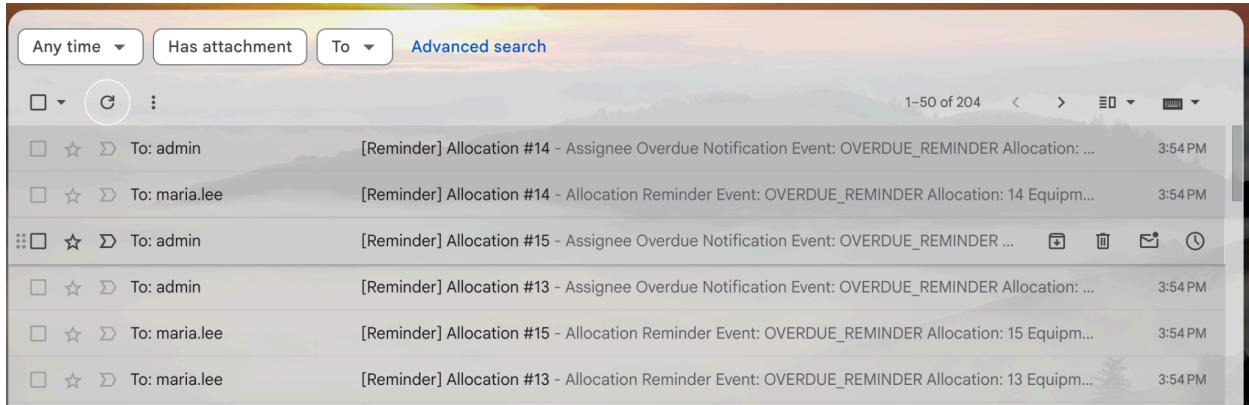
4. Deploy Database

docker run -p 3306:3306 --name events-db mysql

5. Run the Application

```
docker run -p 9000:9000 event-coordination
```

Output:



POST http://0.0.0.0:9000/api | GET http://0.0.0.0:9000/api/ | +

HTTP <http://0.0.0.0:9000/api/equipment/allocations/2>

GET <http://0.0.0.0:9000/api/equipment/allocations/2> Send

Docs Params Authorization Headers (9) Body Scripts Settings Cookies Beautify

Body none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "equipmentId": 1,
3   "employeeId": 4,
4   "purpose": "Client meeting laptop usage",
5   "expectedReturnAt": "2025-11-25T10:30:00Z"
6 }
7
```

Body Cookies Headers (3) Test Results | ↻

200 OK 1.60 s 524 B 🔗 🔍 🔍 🔍 🔍 🔍 🔍 🔍 🔍

{ } JSON ▾ ▶ Preview Visualize ▾

```
1 {
2   "allocationId": 2,
3   "equipmentId": 1,
4   "employeeId": 1,
5   "allocatedByEmployeeId": 1,
6   "purpose": "Client Presentation",
7   "allocatedAt": "2025-11-19T13:30:56Z",
8   "expectedReturnAt": "2025-01-17T17:00:00Z",
9   "returnedAt": "2025-11-20T09:37:48Z",
10  "conditionOnReturn": "GOOD",
11  "returnNotes": "Device working normally, no issues found.",
12  "status": "RETURNED",
13  "createdAt": "2025-11-19T13:30:56Z",
14  "updatedAt": "2025-11-20T09:37:49Z",
15  "isDeleted": false
16 }
```

POST http://0.0.0.0:9000/api | POST http://0.0.0.0:9000/api/ | +

HTTP <http://0.0.0.0:9000/api/equipment/allocations>

POST <http://0.0.0.0:9000/api/equipment/allocations> Send

Docs Params Authorization Headers (9) Body Scripts Settings Cookies Beautify

Body none form-data x-www-form-urlencoded raw binary GraphQL JSON

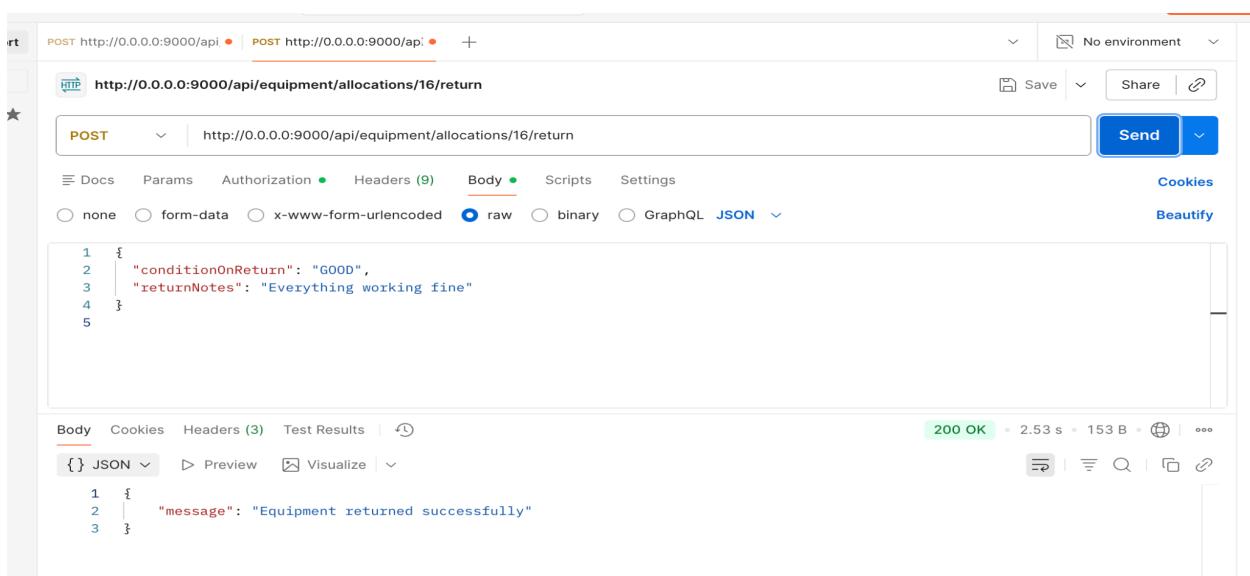
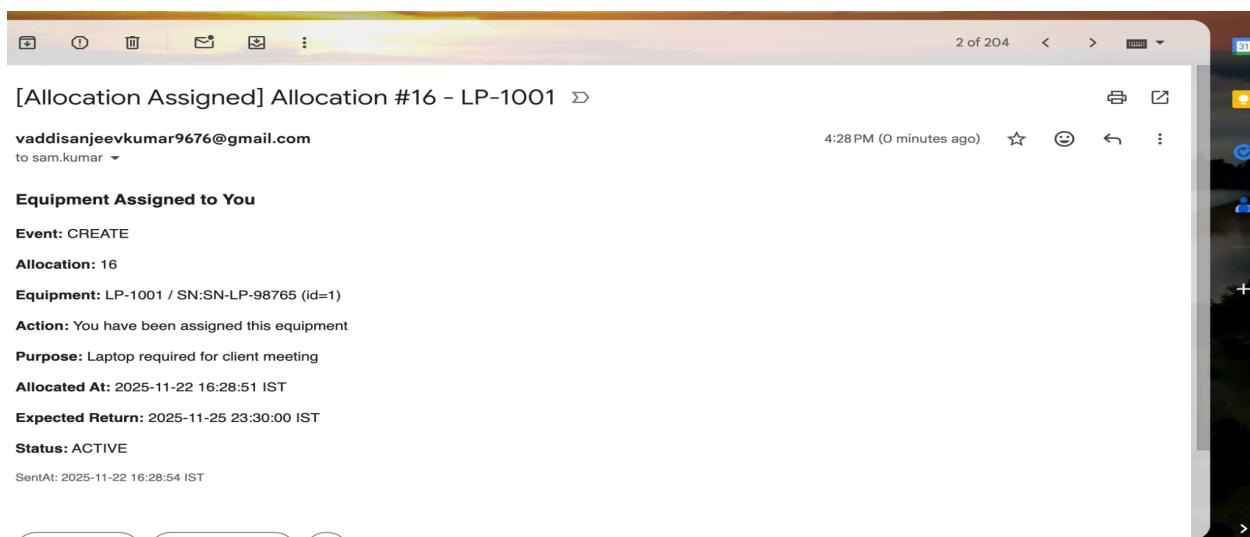
```
1 {
2   "equipmentId": 1,
3   "employeeId": 4,
4   "purpose": "Laptop required for client meeting",
5   "expectedReturnAt": "2025-11-25T18:00:00Z"
6 }
7
```

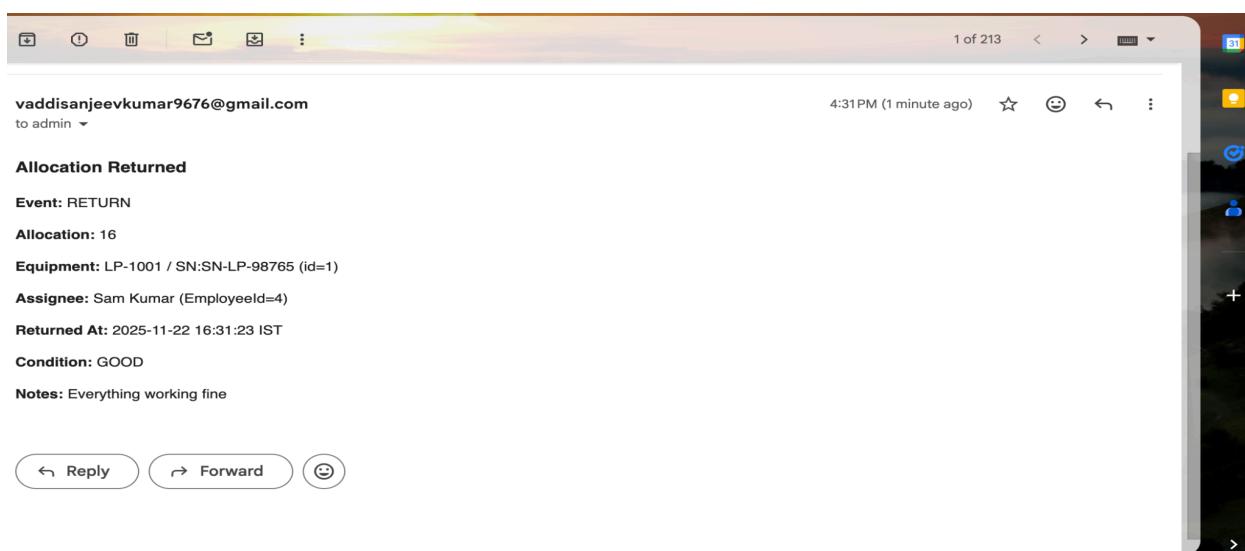
Body Cookies Headers (3) Test Results | ↻

201 Created 2.28 s 132 B 🔗 🔍 🔍 🔍 🔍 🔍 🔍 🔍 🔍

{ } JSON ▾ ▶ Preview Visualize ▾

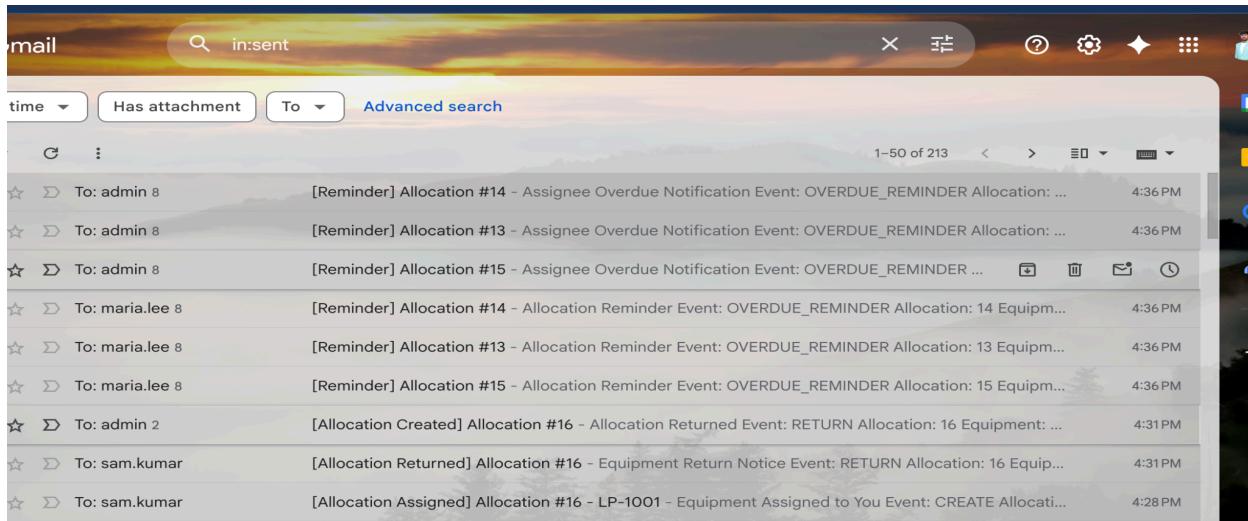
```
1 {
2   "allocationId": 16
3 }
```





The screenshot displays a Postman API test interface. The request method is POST, and the URL is `http://0.0.0.0:9000/api/equipment/allocations/overdue`. The "Body" tab is selected, showing the raw JSON payload sent to the server. The response status is 200 OK, with a response time of 7.96 seconds and a response size of 1016 B. The response body is a JSON array of allocation records:

	allocationId	equipmentId	employeeId	allocatedByEmployeeId	purpose	allocatedAt	expectedReturnAt	status	createdAt	updatedAt
0	13	3	3	1	Client meeting presentation	2025-11-21T12:18:18Z	2025-11-21T17:00:00Z	ACTIVE	2025-11-21T12:18:18Z	2025-11-21T12:18:18Z
1	14	4	3	1	Client meeting presentation	2025-11-21T12:48:34Z	2025-11-21T17:00:00Z	ACTIVE	2025-11-21T12:48:34Z	2025-11-21T12:48:34Z
2	15	5	3	1	Client meeting presentation	2025-11-21T13:07:00Z	2025-11-21T17:00:00Z	ACTIVE	2025-11-21T13:07:00Z	2025-11-21T13:07:00Z



The screenshot shows a Postman API request for "http://0.0.0.0:9000/api/equipment/types". The method is set to "GET". The "Body" tab is selected, showing the response body. The response is a JSON array containing one item, which is a single object representing a type of equipment. The object has properties: typeId (1), typeName (Laptop), description (office laptops for employees), isActive (true), createdat (2025-11-19T11:51:42Z), updatedat (2025-11-19T11:51:42Z), and isDeleted (false). The response status is 200 OK, with a duration of 623 ms and a size of 292 B.

```
[{"typeId": 1, "typeName": "Laptop", "description": "office laptops for employees", "isActive": true, "createdat": "2025-11-19T11:51:42Z", "updatedat": "2025-11-19T11:51:42Z", "isDeleted": false}]
```

GET | http://0.0.0.0:9000/api/equipment/types/1 | **Send**

Docs Params Authorization • Headers (7) **Body** Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (3) Test Results | ⏱

200 OK • 607 ms • 290 B • ⌂ | ⚙

{ } JSON ▾ ▶ Preview ⚡ Visualize | ▾

```
1 {  
2   " typeId": 1,  
3   " typeName": "Laptop",  
4   " description": "Office laptops for employees",  
5   " isActive": true,  
6   " createdAT": "2025-11-19T11:51:42Z",  
7   " updatedAT": "2025-11-19T11:51:42Z",  
8   " isDeleted": false  
9 }
```

POST | http://0.0.0.0:9000/api/equipment/types | **Send**

Docs Params Authorization • Headers (9) **Body** • Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** ▾

Body Cookies Headers (3) Test Results | ⏱

201 Created • 836 ms • 125 B • ⌂ | ⚙

{ } JSON ▾ ▶ Preview ⚡ Visualize | ▾

Beautify

```
1 {  
2   " typeName": "Laptop - 1",  
3   " description": "Portable computer device",  
4   " isActive": true  
5 }  
6
```

Body Cookies Headers (3) Test Results | ⏱

201 Created • 836 ms • 125 B • ⌂ | ⚙

{ } JSON ▾ ▶ Preview ⚡ Visualize | ▾

```
1 {  
2   " typeId": 2  
3 }
```

PUT http://0.0.0.0:9000/api/equipment/types/2

Send

Docs Params Authorization Headers (9) Body Scripts Settings Cookies Beautify

Body raw JSON

```
1 {  
2   "typeName": "High-end Laptop",  
3   "description": "Upgraded laptop with extended specs",  
4   "isActive": true  
5 }  
6
```

Body Cookies Headers (3) Test Results 200 OK 1.25 s 313 B

{ } JSON Preview Visualize

```
1 {  
2   " typeId": 2,  
3   "typeName": "High-end Laptop",  
4   "description": "Upgraded laptop with extended specs",  
5   "isActive": true,  
6   "createdAt": "2025-11-22T11:11:14Z",  
7   "updatedAt": "2025-11-22T11:12:16.595142Z",  
8   "isDeleted": false  
9 }
```

GET http://0.0.0.0:9000/api/equipment/items

Send

Docs Params Authorization Headers (7) Body Scripts Settings Cookies

Body raw JSON

This request does not have a body

Body Cookies Headers (3) Test Results 200 OK 567 ms 2.34 KB

{ } JSON Preview Visualize

```
1 [  
2   {  
3     "equipmentId": 1,  
4     "assetTag": "LP-1001",  
5     "serialNumber": "SN-LP-98765",  
6     "typeId": 1,  
7     "location": "Rack A2",  
8     "status": "AVAILABLE",  
9     "condition": "GOOD",  
10    "createdAt": "2025-11-19T11:52:22Z",  
11    "updatedAt": "2025-11-22T11:01:24Z",  
12    "isDeleted": false  
13  },  
14  {  
15    "equipmentId": 2,  
16    "assetTag": "LP-1002",  
17    "serialNumber": "SN-LP-12345",  
18    "typeId": 1,  
19    "location": "Rack A3",  
20  }]
```

GET Send

Docs Params Authorization Headers (7) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (3) Test Results |

200 OK | 1.54 s 336 B

{ } JSON Preview Visualize |

```
1 {  
2   "equipmentId": 1,  
3   "assetTag": "LP-1001",  
4   "serialNumber": "SN-LP-98765",  
5   "typeId": 1,  
6   "location": "Rack A2",  
7   "status": "AVAILABLE",  
8   "condition": "GOOD",  
9   "createdAt": "2025-11-19T11:52:22Z",  
10  "updatedAt": "2025-11-22T11:01:24Z",  
11  "isDeleted": false  
12 }
```

POST Send

Docs Params Authorization Headers (9) Body Scripts Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {  
2   "assetTag": "LAP-2025-1101",  
3   "serialNumber": "SN-LAP-55201",  
4   "typeId": 1,  
5   "location": "Main Office"  
6 }  
7
```

Body Cookies Headers (3) Test Results |

201 Created | 1.26 s 131 B

{ } JSON Preview Visualize |

```
1 {  
2   "equipmentId": 11  
3 }
```

PUT http://0.0.0.0:9000/api/equipment/items/10

Body **raw**

```
1 {  
2   "assetTag": "LAP-2025-1101-UPDATED",  
3   "serialNumber": "SN-LAP-55201-NEW",  
4   "typeId": 1,  
5   "location": "Conference Room"  
6 }  
7
```

200 OK | 2.43 s | 371 B | ⚡

Body Cookies Headers (3) Test Results | ↻

{ } JSON ▾ ▶ Preview □ Visualize | ↻

```
1 {  
2   "equipmentId": 10,  
3   "assetTag": "LAP-2025-1101-UPDATED",  
4   "serialNumber": "SN-LAP-55201-NEW",  
5   "typeId": 1,  
6   "location": "Conference Room",  
7   "status": "AVAILABLE",  
8   "condition": "GOOD",  
9   "createdAt": "2025-11-19T17:11:28Z",  
10  "updatedAt": "2025-11-22T11:19:38.031456Z",  
11  "isDeleted": false  
12 }
```

GET http://0.0.0.0:9000/api/maintenance/tickets

Headers (7)

Auth Type

Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Token

.....

200 OK | 579 ms | 477 B | ⚡

Body Cookies Headers (3) Test Results | ↻

{ } JSON ▾ ▶ Preview □ Visualize | ↻

```
1 [  
2   {  
3     "ticketId": 1,  
4     "equipmentId": 1,  
5     "allocationId": 1,  
6     "issueReportedAt": "2025-11-19T12:41:12Z",  
7     "issueNotes": "Cracked screen, needs replacement",  
8     "status": "RESOLVED",  
9     "severity": "HIGH",  
10    "createdByEmployeeId": 1,  
11    "assignedToEmployeeId": 3,  
12    "isResolved": true,  
13    "resolvedAt": "2025-01-18T10:15:00Z",  
14    "createdAt": "2025-11-19T12:41:12Z",  
15    "updatedAt": "2025-11-19T12:44:14Z",  
16    "isDeleted": false  
17  }  
18 ]
```

Terminal | Runner | Start Proxy | Cookies | Vault | Trash

The screenshot shows the Postman interface with a successful API call. The URL is `http://0.0.0.0:9000/api/maintenance/tickets/1`. The response status is **200 OK**, with a duration of 711 ms and a size of 475 B. The response body is a JSON object representing a ticket:

```
1 {  
2   "ticketId": 1,  
3   "equipmentId": 1,  
4   "allocationId": 1,  
5   "issueReportedAt": "2025-11-19T12:41:12Z",  
6   "issueNotes": "Cracked screen, needs replacement",  
7   "status": "RESOLVED",  
8   "severity": "HIGH",  
9   "createdByEmployeeId": 1,  
10  "assignedToEmployeeId": 3,  
11  "isResolved": true,  
12  "resolvedAt": "2025-01-18T10:15:00Z",  
13  "createdAt": "2025-11-19T12:41:12Z",  
14  "updatedAt": "2025-11-19T12:44:14Z",  
15  "isDeleted": false  
16 }
```

The screenshot shows the Postman interface with the following details:

- Method:** POST
- URL:** http://0.0.0.0:9000/api/maintenance/tickets
- Body:** JSON (selected)
- Request Body Content:**

```
1 {  
2   "equipmentId": 10,  
3   "allocationId": 1,  
4   "issueNotes": "Laptop screen flickering intermittently",  
5   "severity": "HIGH"  
6 }  
7
```
- Response Status:** 201 Created
- Response Headers:** 910 ms, 127 B
- Response Body:**

```
1 {  
2   "ticketId": 2  
3 }
```

POST http://0.0.0.0:9000/api/maintenance/tickets

Body raw binary GraphQL JSON

```
1 {  
2   "equipmentId": 11,  
3   "allocationId": null,  
4   "issueNotes": "Mouse not working",  
5   "severity": "LOW"  
6 }  
7
```

Body Cookies Headers (3) Test Results | ⏱

201 Created 931 ms 127 B | ⌂

{ } JSON ▾ ▶ Preview ⌂ Visualize | ⏱

```
1 {  
2   "ticketId": 3  
3 }
```

PUT http://0.0.0.0:9000/api/maintenance/tickets/1/status

Body raw binary GraphQL JSON

```
1 {  
2   "status": "IN_PROGRESS",  
3   "isResolved": false,  
4   "resolvedAt": null  
5 }  
6
```

Body Cookies Headers (3) Test Results | ⏱

200 OK 2.04 s 136 B | ⌂

{ } JSON ▾ ▶ Preview ⌂ Visualize | ⏱

```
1 {  
2   "message": "Status updated"  
3 }
```

PUT http://0.0.0.0:9000/api/maintenance/tickets/2/status

Send

Docs Params Authorization Headers (9) Body Scripts Settings Cookies Beautify

Body raw binary GraphQL JSON

```
1 {
2   "status": "RESOLVED",
3   "isResolved": true,
4   "resolvedAt": "2025-11-22T14:00:00Z"
5 }
6
```

Body Cookies Headers (3) Test Results 200 OK 1.31 s 136 B

{ } JSON Preview Visualize

```
1 {
2   "message": "Status updated"
3 }
```

PUT http://0.0.0.0:9000/api/maintenance/tickets/2/assign

Send

Docs Params Authorization Headers (9) Body Scripts Settings Cookies Beautify

Body raw binary GraphQL JSON

```
1 {
2   "assignedToEmployeeId": 4
3 }
4
```

Body Cookies Headers (3) Test Results 200 OK 1.35 s 137 B

{ } JSON Preview Visualize

```
1 {
2   "message": "Ticket assigned"
3 }
```

Docker

The screenshot shows the Postman application interface. On the left, the sidebar displays 'equipment-gcp' workspace, 'Collections' (selected), 'Environments', 'History', 'Flows', and 'Files (BETA)'. The main area shows a collection named 'My first collection' containing two folders: 'First folder inside collection' and 'Second folder inside collection'. The 'First folder' has three requests: 'GET', 'POST', and 'GET'. The 'Second folder' has two requests: 'GET' and 'GET'. Below the collection tree, there is a section titled 'Create a collection for your requests' with a note about grouping related requests and setting common authorization. A 'Create Collection' button is present.

Request:

```
POST http://34.27.107.42:9000/api/login
```

Body:

```
1 {
2   "username": "admin",
3   "password": "password@123"
4 }
```

Response:

200 OK 2.70 s 349 B

```
1 {
2   "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJTUlZT1NiJ9,
3   eyJzdWIiOiIxIiwiaXNzIjoiZXFlaXBlZw50LWFsbG9jYXRpb24iLCJleHaiOjE3NjQyMjczMDoImlhdCI6MTc2NDE0MDk
4   wX0.uf71Boz9ejwsWIKRs76BxUxF6GfNFzmTcrp2uEqwss",
5   "userId": 1,
6   "username": "admin",
7   "expiresIn": 86400
8 }
```

The screenshot shows the Postman application interface. The sidebar is identical to the first screenshot, displaying 'equipment-gcp' workspace, 'Collections' (selected), 'Environments', 'History', 'Flows', and 'Files (BETA)'. The main area shows the same 'My first collection' structure with its respective requests.

Request:

```
GET http://34.27.107.42:9000/api/equipment/allocations
```

Body:

```
1 ^vP to Ask AI
```

Response:

200 OK 2.33 s 6.31 KB

```
1 {
2   "allocationId": 3,
3   "equipmentId": 2,
4   "employeeId": 3,
5   "allocatedByEmployeeId": 1,
6   "purpose": "Client meeting presentation",
7   "allocatedAt": "2025-11-19T23:07:30Z",
8   "expectedReturnAt": "2025-11-21T22:30:00Z",
9   "returnedAt": "2025-11-22T10:42:41Z",
10  "conditionOnReturn": "GOOD",
11  "returnNotes": "All good. Returned after client demo.",
12  "status": "RETURNED",
13  "createdAt": "2025-11-19T23:07:30Z",
14  "updatedAt": "2025-11-22T10:42:41Z",
15  "isDeleted": false
16 }
```

The screenshot shows a list of items, likely notifications or tasks, in a grid format. The columns from left to right are: a checkbox, a star icon, a reply-all icon, a recipient field (e.g., To: admin), a detailed message summary, and a timestamp. A blue vertical bar highlights the second item in the list.

<input type="checkbox"/>			To: admin	[Reminder] Allocation #15 - Assignee Overdue Notification Event: OVERDUE_REMINDER Allocation: 15 Equipment: LP-1005 / SN:SN-LP...	12:35PM
<input type="checkbox"/>			To: maria.lee	[Reminder] Allocation #15 - Allocation Reminder Event: OVERDUE_REMINDER Allocation: 15 Equipment: LP-1005 / SN:SN-LP-12348 (id...	12:35PM
<input type="checkbox"/>			To: admin	[Reminder] Allocation #14 - Assignee Overdue Notification Event: OVERDUE_REMINDER Allocation: 14 Equipment: LP-1004 / SN:SN-LP...	12:35PM
<input type="checkbox"/>			To: admin	[Reminder] Allocation #13 - Assignee Overdue Notification Event: OVERDUE_REMINDER Allocation: 13 Equipment: LP-1003 / SN:SN-LP...	12:35PM
<input checked="" type="checkbox"/>			To: maria.lee	[Reminder] Allocation #14 - Allocation Reminder Event: OVERDUE_REMINDER Allocation: 14 Equipment: LP-1004 / SN:SN-L...	
<input type="checkbox"/>			To: maria.lee	[Reminder] Allocation #13 - Allocation Reminder Event: OVERDUE_REMINDER Allocation: 13 Equipment: LP-1003 / SN:SN-LP-12346 (id...	12:35PM
<input type="checkbox"/>			To: admin 2	[Event Created] Inter-Company Cricket Tournament - [Event Created] Inter-Company Cricket Tournament Title: Inter-Company Cricke...	9:20 AM
<input type="checkbox"/>			To: admin	[Event Created] Product Dinner - [Event Created] Product Dinner Title: Product Dinner Type: Launch Date: 2025-12-15 14:30:00 IST Lo...	9:20 AM
<input type="checkbox"/>			To: admin 9	[Reminder] Allocation #13 - Assignee Overdue Notification Event: OVERDUE_REMINDER Allocation: 13 Equipment: LP-1003 / SN:SN-LP...	Nov 22
<input type="checkbox"/>			To: admin 9	[Reminder] Allocation #14 - Assignee Overdue Notification Event: OVERDUE_REMINDER Allocation: 14 Equipment: LP-1004 / SN:SN-LP...	Nov 22
<input type="checkbox"/>			To: admin 9	[Reminder] Allocation #15 - SentAt: 2025-11-22 17:06:02 IST	Nov 22