Project 1: Hotel Reception Desk Application

A hotel with 10 floors, each featuring a consistent room layout, requires a Reception Desk application for seamless room allotment and guest service management. Each floor contains three categories of rooms—Deluxe, Luxury, and Luxury Suite—each with a unique price point. The standard check-in and check-out time is set at 11 AM daily.

The application should provide functionality for managing guest check-ins and check-outs, including an option for ID proof upload during check-in. All guest and booking data should be securely stored in a database.

Key Requirements:

1. **Room Booking and Management**:
    - Enable smooth check-in and check-out processes with real-time room availability tracking.
    - Store guest booking details and ID proofs securely in a database.
2. **Automated Notifications and Service Coordination**:
    - Upon room booking, an automated notification should trigger communication with the **Room Service**, **WiFi Service**, and **Restaurant Service** teams:
        - **Room Service**: Sends a welcome email to the guest with emergency contact numbers and room service contact details.
        - **WiFi Service**: Sends an email with Wi-Fi login credentials.
        - **Restaurant Service**: Emails the daily menu to the guest each morning, stopping automatically upon check-out.
    - These operations should be managed by a dedicated microservice that employs the **Akka Framework** with the Actor Model to handle asynchronous messaging for these teams.
3. **Backend and Integration**:
    - Develop the booking system as a Play Framework web application, supporting REST API operations.
    - Use Kafka for efficient, reliable message queuing to ensure communication with the dedicated service microservice.
    - The Akka Actor Model will handle task-specific messaging for the Room Service, WiFi, and Restaurant Service teams within this microservice.

4. **Database Design, Email Handling, and Deployment**:
    - Design tables to store room categories, guest data, ID proofs, check-in/check-out records, and notifications.
    - Utilize Akka and Kafka within the microservice for efficient email handling and notifications.
    - Ensure all components, including the Play Framework application and the service microservice, are deployed in **Docker** containers for modular, consistent deployment across environments.

Project 2: A corporate office requires a **Visitor Management System** to streamline visitor check-in, check-out, and service coordination. This system should allow reception staff to manage visitor entry and exit while notifying relevant internal teams of visitor arrivals.

System Requirements

1. **Visitor Check-In and Check-Out**:
    - Receptionists can check in visitors and capture essential details like name, contact information, and purpose of visit.
    - Visitors must upload an ID proof during check-in, which is securely stored in the system.
2. **Automated Notifications**:
    - Upon check-in, notifications are sent to:
        - **Host Employee**: Receives an email notifying them of the visitor's arrival.
        - **IT Support**: Prepares visitor Wi-Fi access and sends login details to the visitor via email.
        - **Security Team**: Receives a message about the visitor's entry and clearance status.
    - Notifications should automatically stop after check-out.
3. **Backend and Deployment**:
    - The system should be developed as a Play Framework web application, with a REST API to manage check-in/check-out operations.
    - A separate microservice will be created using **Akka Actors** to handle messaging for the IT and Security teams.

- **Kafka** will be used to queue messages, ensuring reliable communication with service teams.
- All components should be containerized with **Docker** for easy deployment.

Project 3: Corporate Equipment Allocation and Tracking System

A corporate office requires a **Corporate Equipment Allocation and Tracking System** to streamline the process of distributing and tracking shared office equipment like laptops, projectors, and tablets. The system allows reception staff to manage equipment allocation and ensure that each item is tracked for availability, condition, and timely return.

System Requirements

1. **Equipment Allocation and Return**:
   - Reception staff manages equipment allocation requests, recording employee details (name, department, and purpose) along with the expected return date.
   - A record is created for each item allocated, including employee details, allocation date, and expected return date.
   - Upon return, the system verifies the equipment's condition and logs return details, alerting the maintenance team if servicing or repairs are needed.
2. **Automated Notifications**:
   - **Overdue Reminder**: Sends an automated reminder email to employees if equipment is not returned by the expected date.
   - **Maintenance Team Notification**: Alerts the maintenance team if an item is flagged as damaged or due for servicing upon return.
   - **Inventory Team Notification**: Notifies the inventory team whenever equipment is allocated or returned, ensuring up-to-date availability tracking.
3. **Backend and Deployment**:
   - Develop the system using the **Play Framework** with REST APIs to manage allocation/return operations and track equipment status.
   - A dedicated microservice using **Akka Actors** will handle asynchronous notifications, including overdue reminders and maintenance alerts.
   - Use **Kafka** as a message broker to queue and manage notifications to maintenance and inventory teams, ensuring reliable communication.

- All components should be containerized with **Docker** for scalable and consistent deployment.

Project 4:

Office Meeting Room Management System with Room Preparation Notifications

A corporate office requires an Office Meeting Room Management System to streamline the reservation, usage, and readiness of its meeting rooms. With limited rooms available, the system should enable Admin Staff to manage room bookings efficiently, while a Room Service Team prepares rooms based on real-time notifications. Additionally, System Admins will oversee the system's operation, including user access, room availability, and notification settings.

System Requirements

1. **Meeting Room Reservation:**
    - Admin staff can reserve rooms on behalf of employees, entering details such as employee name, department, purpose of the meeting, and expected duration.
    - The system provides real-time room availability and prevents double bookings by locking rooms immediately upon reservation.
    - System Admins will have access to configure the system, manage user roles, monitor room availability, and troubleshoot any booking conflicts.
2. **Automated Notifications:**
    - Booking Confirmation: Sends a confirmation email to the employee with details about the meeting room location, reservation time, and any special requirements.
    - Room Preparation Notification: Notifies the Room Service Team with details of the upcoming booking, allowing them to prepare the room (e.g., cleaning, setting up equipment).
    - Reminder Alert: Sends a reminder email to the employee 15 minutes before the scheduled reservation.
    - Release Notification: If the room remains unused for 15 minutes after the start time, the system automatically releases it, notifies the admin staff, and updates the Room Service Team to reset or clean the room for the next booking.
3. **Backend and Deployment:**
    - Develop the system using Play Framework with REST APIs to handle reservations, room availability, and user roles.

- Implement a dedicated Akka-based microservice for handling automated notifications, ensuring timely updates to the employee, admin staff, and Room Service Team.
- Use Kafka to manage notification queues reliably, ensuring reminders, confirmations, and room preparation alerts are consistently delivered.
- Deploy all components using Docker for streamlined, scalable deployment across environments.

Project 5:

An event management company specializes in organizing and coordinating various services for events like weddings, corporate gatherings, and private parties. They need an Event Management Service Coordination System to help the event manager assign tasks to different service teams, track preparation timelines, and send notifications to ensure everything is ready for the event day. The system will involve a single user (the event manager) who coordinates multiple teams.

System Requirements

1. **Event Setup and Task Assignment:**
    - The event manager creates an event plan with details like event type (e.g., wedding, corporate event, birthday), date, and expected guest count.
    - Based on the event requirements, the event manager assigns tasks to various service teams:
        - Catering Team: Responsible for food and beverage service.
        - Decorations Team: Sets up and decorates the event space.
        - Entertainment Team: Manages audio-visual equipment, DJs, or live performers.
        - Logistics Team: Arranges transportation, guest assistance, and any required setup/tear-down.
    - Each team receives specific instructions for their tasks, deadlines, and any special requests related to the event.
2. **Automated Notifications and Reminders:**
    - Task Assignment Notification: Sends notifications to each service team when they are assigned tasks, with details on expectations, timelines, and special requirements.
    - Preparation Reminders: Sends reminders to each team one day before and a few hours before their scheduled setup time, ensuring all teams are on schedule.

- Progress Check-In: Periodically sends reminders to each team to update the task status (e.g., In Progress, Completed), allowing the event manager to monitor progress.
- Event Day Alert: Notifies the event manager and all teams of final preparations on the event day to ensure all tasks are complete.
- Issue Alert: Notifies the event manager immediately if any delays or issues are reported by a team, allowing for quick intervention.

3. **Backend and Deployment:**
    - Develop the system using Play Framework with REST APIs for managing events, task assignments, and notifications.
    - Implement a dedicated Akka-based microservice to handle real-time notifications, reminders, and alerts for each team based on the event schedule.
    - Use Kafka to queue notifications, ensuring reliable delivery of reminders, progress updates, and alerts for all teams involved.
    - Deploy all components using Docker to provide a consistent environment and scalability.