Use Case Diagram for Drone Integration in HMS

Actors:

1. Administrator – Oversees and configures drone operations.
2. Doctor – Requests urgent deliveries (medications or samples).
3. Nurse – Requests medical supplies delivery.
4. Pharmacist – Dispatches medications.
5. Lab Technician – Sends and receives lab samples.
6. Drone Operator – Monitors and manages drones in-flight.
7. Patient – Tracks delivery of personal medical supplies.
8. Drone System – Automated actor representing the drones and their control system.

Use Cases by Platform

1. Desktop Platform

Primarily used by the Administrator, Pharmacist, Lab Technician, and Drone Operator to manage drones and monitor operations.

Administrator

Manage Drone Inventory: Register new drones, assign IDs, track operational status.

Monitor Drone Status: View current drone activity, location, and condition.

Configure Drone Routes: Set delivery routes and specify restrictions.

Generate Drone Reports: Get performance metrics for analysis.

Pharmacist

Prepare Drone Delivery: Set up medication packages for drone transport.

Track Delivery Status: Monitor the delivery progress of dispatched medications.

Lab Technician

Request Sample Delivery: Request drone pickup for urgent lab samples.

Receive Delivered Samples: Track and receive samples delivered by drone.

Drone Operator

Monitor Drone Operations: View real-time drone positions and status.

Handle Emergency Interventions: Override drone route for urgent redirection.

1. Mobile Platform

Accessible to Doctors, Nurses, and Patients, focusing on real-time requests, tracking, and status updates.

Doctor

Request Urgent Delivery: Send emergency requests for medications or lab sample transport.

Nurse

Request Supply Delivery: Request critical medical supplies to the patient’s location or hospital.

Patient

Track Delivery Status: View status updates for personal medical supply deliveries.

Shared Use Cases Across Platforms

Automated Drone Dispatch (Drone System): Automatically assigns drones to delivery requests based on availability.

Route Optimization (Drone System): Automatically calculates and updates the fastest route for each delivery.

Battery and System Monitoring (Drone System): Monitors battery and system health, prompting maintenance if necessary.

Class Diagram for Drone Integration in HMS

The class diagram captures drone functionality and key components in the HMS to support this integration.

// Drone System Class

Public class DroneSystem {

Private String id;

Private String status; // “Available”, “In Transit”, “Under Maintenance”

Private int batteryLevel;

Private Location currentLocation;

Private Route currentRoute;

Public void dispatchDrone(DeliveryRequest request) { /\* Implementation \*/ }

Public void monitorBattery() { /\* Implementation \*/ }

Public void updateRoute(Route newRoute) { /\* Implementation \*/ }

Public void sendStatusUpdate() { /\* Implementation \*/ }

}

// Administrator Class

Public class Administrator {

Private String username;

Private String password;

Public void manageDroneInventory() { /\* Implementation \*/ }

Public void configureDroneRoutes(Route route) { /\* Implementation \*/ }

Public void generateDroneReports() { /\* Implementation \*/ }

}

// Drone Operator Class

Public class DroneOperator {

Private String id;

Private String name;

Public void monitorDroneOperations() { /\* Implementation \*/ }

Public void handleEmergencyIntervention(DroneSystem drone) { /\* Implementation \*/ }

}

// DeliveryRequest Class

Public class DeliveryRequest {

Private String requestId;

Private String type; // “Medication”, “Lab Sample”, “Medical Supplies”

Private Location pickupLocation;

Private Location deliveryLocation;

Private Date requestDate;

Public DeliveryRequest(String type, Location pickup, Location delivery) { /\* Implementation \*/ }

}

// Doctor Class

Public class Doctor {

Private String id;

Private String name;

Public void requestUrgentDelivery(DeliveryRequest request) { /\* Implementation \*/ }

}

// Nurse Class

Public class Nurse {

Private String id;

Private String name;

Public void requestSupplyDelivery(DeliveryRequest request) { /\* Implementation \*/ }

}

// Patient Class

Public class Patient {

Private String id;

Private String name;

Public void trackDeliveryStatus(String deliveryId) { /\* Implementation \*/ }

}

// Pharmacist Class

Public class Pharmacist {

Private String id;

Private String name;

Public void prepareDroneDelivery(DeliveryRequest request) { /\* Implementation \*/ }

}

// Lab Technician Class

Public class LabTechnician {

Private String id;

Private String name;

Public void requestSampleDelivery(DeliveryRequest request) { /\* Implementation \*/ }

Public void receiveDeliveredSamples() { /\* Implementation \*/ }

}

// Route Class

Public class Route {

Private Location startLocation;

Private Location endLocation;

Private List<Location> waypoints;

Public void calculateOptimalRoute() { /\* Implementation \*/ }

}

// Location Class

Public class Location {

Private double latitude;

Private double longitude;

}

Key Classes Explained

DroneSystem: Manages drone operations, including dispatch, battery monitoring, route updates, and status updates.

Administrator: Manages inventory and configuration of drones, as well as generating reports.

DroneOperator: Responsible for monitoring drones in-flight and handling emergency interventions.

DeliveryRequest: Represents requests for delivery, whether it’s medication, lab samples, or supplies.

Route: Represents delivery routes with methods for calculating optimal paths.

Location: Represents geographic coordinates for drone routing and delivery.

Access Modifiers

Private attributes protect sensitive data like id or status.

Public methods allow interaction with other classes and users.

Protected could be added to attributes that might be extended in subclasses, such as name for user roles.