

Web FrameWork Development

Deliverable 2



April 6, 2025

TU Dublin

Sam O’Connor (B00148322)

**Use case Spec Diagrams**

Given my Username ends with ‘22’ I have been tasked with the theme ‘Health’ and the must have Use Case is ‘Purchase Orders (PO-Items)’.

A diagram of a product

AI-generated content may be incorrect.Purchase Orders Use case diagram:

The diagram represents a Purchase Order System. There are three main user roles:

* Admin
* Manager
* Employee

**Roles:**

* Admin:
  + **Review Purchase Orders:** The Admin can see and review orders that have been created.
  + **Approve Orders:** They decide if an order should be approved.
* Manager:
  + **Create Purchase Order:** The Manager is responsible for creating new purchase orders.
  + **Cancel Purchase Order:** There is a direct link from the Manager to cancel an order if necessary.
* Employee:
  + **View Purchase Orders:** Employees can only view the orders; they don’t have permissions to create, cancel, or review orders.

**Relationships:**

***<<extend>>* between Manage Purchase Order and Approve Purchase Orders:**  
This means that while the admin is managing purchase orders, they *might* also choose to approve one — but they don’t have to. It’s an optional step that only happens when needed.

***<<extend>>* between Manage Purchase Order and Order Rejected:**  
Sometimes, instead of approving an order, the admin might reject it. Since rejecting isn’t something they always do, it’s shown as an optional action using extend.

**<<include>> between Manage Purchase Order and View Purchase Order:**  
Whenever the admin is managing orders, they always need to view the order details first. Because this step is always part of the process, it's shown using include.

**<<extend>> between Create Purchase Order and View Purchase Order:**  
After the manager creates an order, they can choose to look at it, but it's not required. That’s why it’s shown as an optional extra step with extend.

**<<extend>> between Cancel Purchase and View Purchase Order:**  
To cancel an order, the manager usually needs to look at it first. But in some cases, they might already know what they’re cancelling, so viewing it isn’t always needed — that’s why it’s an extend.

Patient Admission Use Case

A diagram of a system

AI-generated content may be incorrect.

There are three main user roles:

* Admin
* Receptionist
* Patient

**Roles:**

**• Patient:**

* **Register:** The patient fills out their own registration information to start the admission process.
* **Update Details:** After registering, the patient can optionally update their personal information.

**• Receptionist:**

* **Approve Registration:** The receptionist is responsible for reviewing and approving patient registrations.
* **Verify Insurance:** As part of approval, the receptionist must check that the patient’s insurance is valid.
* **Assign Room:** If the patient is being admitted, the receptionist may assign them a room.
* **Cancel Registration:** If needed, the receptionist can cancel a registration (e.g., if details are incorrect or insurance fails).

**• Admin:**

* **Approve Registration:** The admin has access to approve registrations (like a supervisor).
* **Cancel Registration:** The admin also has the ability to cancel registrations.
* **View Registration List:** The admin can view the list of all registered patients but doesn’t get involved in the actual registration or approval process.

**Relationships:**

* **<<extend>>** between **Register** and **Update Details**:  
  This means that after registering, patients have the optional ability to update their information.
* **<<include>>** between **Approve Registration** and **Verify Insurance**:  
  Verifying insurance is a required step in the approval process, so it is always included.
* **<<extend>>** between **Approve Registration** and **Assign Room**:  
  Assigning a room only happens if the patient is being admitted, so it’s an optional step.
* **<<extend>>** between **Approve Registration** and **Cancel Registration**:  
  Cancellation happens only in certain situations (e.g., duplicate entry, insurance failed), so it’s optional.
* **Dashed dependency line** between **Register** and **Approve Registration**:  
  This shows that when a patient registers, their information is passed on to the receptionist for approval. However, the patient does **not** have access to the approval process.

Patient Appointment Use Case Diagram

A diagram of a patient appointment

AI-generated content may be incorrect.

There are three main user roles:  
• Patient  
• Receptionist  
• Doctor

**Roles:**  
• **Patient:**  
    • Request Appointment: The patient picks a preferred date/time and asks the receptionist to book it.  
    • Cancel Appointment: The patient can cancel a booked appointment by contacting the receptionist.

• **Receptionist:**  
    • Schedule Appointment: The receptionist takes the patient’s request and enters it into the system.  
    • Send Appointment Confirmation: After booking, the system (or receptionist) always sends a confirmation message to the patient.  
    • View Appointment List: The receptionist can view all upcoming appointments in the calendar.  
    • Cancel Appointment: The receptionist can cancel appointments in the system when the patient asks or in other valid cases.  
    • Reschedule Appointment: The receptionist can change an appointment’s date/time if the patient or doctor requests it.

• **Doctor:**  
    • View Appointments: The doctor can see their personal schedule of booked appointments.  
    • Reschedule Appointment: The doctor can request a new time for an existing appointment, which the receptionist then processes.

**Relationships:**  
• <<include>> between **Request Appointment** and **Schedule Appointment**:  
Every patient request leads to scheduling by the receptionist.

• <<include>> between **Schedule Appointment** and **Send Appointment Confirmation**:  
Every time an appointment is scheduled, a confirmation is always sent to the patient.

• <<include>> between **Schedule Appointment** and **View Appointment**:  
When scheduling an appointment, the receptionist always checks the appointment calendar to avoid conflicts. So viewing is always part of the scheduling process.

• <<extend>> between **Schedule Appointment** and **Cancel Appointment**:  
Cancelling is optional and only happens if the patient or doctor asks, or if other changes come up.

• <<extend>> between **Schedule Appointment** and **Reschedule Appointment**:  
Rescheduling is optional and only happens when the date/time needs to be changed.

• Dashed dependency line between **Request Appointment** and **Schedule Appointment**:  
This shows that when a patient makes a request, their info is passed along to the receptionist to complete the scheduling. The patient doesn’t schedule it themselves.

A diagram of a diagram

AI-generated content may be incorrect.Patient Payment Use Case Diagram

There are three main user roles:  
• Patient  
• Receptionist  
• Admin

**Roles:**

**• Patient:**

* **View Receipt:** The patient can check the details of their current charges.
* **Make Payment:** The patient pays their bill, either in person or online.
* **View Billing History:** The patient can see past receipts and payment history.

**• Receptionist:**

* **Create Receipt:** The receptionist creates a receipt after treatment or services have been completed.
* **Process Payment:** The receptionist confirms and processes the patient’s payment in the system.
* **Issue Receipt:** Once payment is processed, a receipt is issued to the patient.

**• Admin:**

* **View Billing History:** The admin has access to all patient billing records.
* **Generate Financial Report:** The admin can generate reports based on payment activity and issued receipts.

**Relationships:**

* **<<include>> between Make Payment and View Receipt:**  
  The patient must check the bill details before making a payment.
* **<<include>> between Make Payment and Process Payment:**  
  All payments made by the patient go through the receptionist for processing.
* **<<include>> between Process Payment and Issue Receipt:**  
  After a payment is processed, the system always issues a receipt to the patient.
* **Dashed dependency from Issue Receipt to Generate Financial Report:**  
  This shows that receipts (and processed payments) are used as a data source when the admin generates financial reports.

**Electronic Medical Records (EMR) Management Use Case**

A diagram of a medical record

AI-generated content may be incorrect.

There are four main user roles:  
• Patient  
• Doctor  
• Nurse  
• Admin

**Roles:**

**• Patient**

* **View Medical Record:** The patient can view their own record (subject to permissions).

**• Doctor**

* **Create Medical Record:** When a new patient is registered, the doctor creates their initial record.
* **Update Medical Record:** The doctor adds diagnoses, treatment notes or test results to an existing record.
* **View Medical Record:** The doctor can view any patient’s record when treating them.

**• Nurse**

* **Update Medical Record:** The nurse records vital signs, medications administered, and nursing notes.
* **View Medical Record:** The nurse can view the record to check orders and care plans.

**• Admin**

* **View Medical Record:** The admin has read‑access to any record for audit or reporting.
* **Access Permissions:** The admin can set or adjust who may view or edit each patient’s record.

**Relationships:**

* **Dashed dependency** between **Create Medical Record** and **Update Medical Record:**  
  When a patient’s record is created, the system establishes it so that updates can happen later (but updating isn’t automatic).
* **<<include>> between Update Medical Record and View Medical Record:**  
  You always view a record before updating it, so viewing is a mandatory sub‑step of updating.
* **<<extend>> between View Medical Record and Access Permissions:**  
  In certain situations—such as sensitive data or privacy rules—the system will invoke the permissions check. This happens only when needed, so it’s an extension.