

## Experiment 1

**Aim: Draft a project plan for any project.**

### HOSPITAL MANAGEMENT SYSTEM (HMS)

A **Hospital Management System (HMS)** is a comprehensive software solution designed to manage, control, and streamline all aspects of hospital operations. It acts as a **centralized digital platform** where administrators, doctors, nurses, lab technicians, pharmacists, and receptionists can work in a coordinated and efficient manner.

HMS replaces traditional **paper-based records** with a secure, organized, and easily accessible digital system. This not only enhances productivity but also reduces human errors, ensures data accuracy, and improves overall patient care.

The system integrates various hospital functions such as **patient registration, appointment scheduling, medical records management, doctor and staff scheduling, billing and accounting, inventory control, pharmacy management, and laboratory operations** into a **single unified platform**.

With HMS, patients can experience **faster service**, doctors can **access patient histories instantly**, and administrators can **monitor operations in real time** through dashboards and reports.

#### Key Features of HMS :-

##### 1. Patient Registration & Records

- Captures personal details like name, contact, age, gender, and address.
- Maintains a complete medical history including diagnoses, prescriptions, lab results, allergies, and ongoing treatments.
- Provides a **unique patient ID** for easy record retrieval across different hospital departments.

##### 2. Appointment Scheduling

- Allows patients to **book, reschedule, or cancel** appointments through reception or online portals.
- Displays real-time doctor availability to avoid scheduling conflicts.
- Sends **SMS/Email reminders** to reduce missed appointments.

##### 3. Doctor & Staff Management

- Maintains detailed doctor profiles, including qualifications, specialties, and consultation timings.
- Helps in **shift management** for nurses, technicians, and support staff.
- Tracks performance metrics and workload distribution.

#### 4. Billing & Payments

- Automatically generates bills based on consultations, diagnostic tests, medicines, and room charges.
- Supports multiple payment methods (cash, card, online payments).
- Maintains a **financial transaction log** for accounting and auditing purposes.

#### 5. Pharmacy Management

- Tracks stock levels of medicines and medical supplies.
- Issues medicines based on doctor prescriptions and updates stock in real time.
- Alerts for **low stock** and **near-expiry** medicines to prevent shortages or wastage.

#### 6. Laboratory Management

- Receives lab test requests directly from doctors via the system.
- Assigns tests to lab technicians and updates the progress.
- Uploads test reports to the patient's profile, which can be accessed by doctors and patients securely.

### Advantages of HMS

- **Efficiency:** Reduces manual paperwork and administrative delays.
- **Accuracy:** Minimizes human errors by automating data entry and calculations.
- **Accessibility:** Allows authorized staff to access records instantly from any department.
- **Security:** Protects sensitive patient data with encryption and role-based access control.
- **Scalability:** Can handle operations for both small clinics and large multi-specialty hospitals.

## Problem Analysis and Project Planning

Hospitals handle **thousands of records, patients, and tasks** every single day. These include patient admissions, doctor consultations, diagnostic tests, medicine issuance, billing, and discharge procedures.

When these operations are performed manually using paper files and registers, they often lead to:

- **Delays** in service delivery due to manual searching and updating of records.
- **Human errors** in data entry, billing, or prescription handling.
- **Difficulty in coordination** between departments like OPD, pharmacy, and labs.
- **Lack of data security**, as physical files are prone to loss or unauthorized access. ●  
**No real-time tracking**, making it hard for management to take quick decisions.

To solve these challenges, a **Hospital Management System (HMS)** automates and integrates all hospital operations into a **centralized digital system**. This ensures that every department works in synchronization and information flows smoothly between them.

## **Key Benefits of HMS**

1. **Reduces Paperwork and Errors:** By digitizing records, the system eliminates the need for repetitive manual entries and reduces mistakes.
2. **Improves Coordination:** Interconnected modules allow instant communication between reception, doctors, labs, and pharmacy.
3. **Quick Access to Records:** Authorized staff can retrieve patient history, prescriptions, and reports within seconds.
4. **Enhanced Security:** Sensitive patient information is protected with encryption and access control.
5. **Cost & Time Savings:** Faster operations mean reduced waiting times for patients and more efficient resource utilization for the hospital.
6. **Better Decision Making:** Real-time data and analytical reports help hospital administration in strategic planning.

## Experiment - 2

**Aim:** Development of SRS document.

### Modules of Hospital Management System (HMS)

#### 1.1 Administrator Module

The **Administrator** is the highest-level user in the system who has complete control over all modules. This module is responsible for:

- **User Management:** Create, update, and delete accounts for doctors, nurses, receptionists, lab technicians, pharmacists, and patients.
- **Department Management:** Create and maintain hospital departments such as Cardiology, Neurology, Orthopedics, Emergency, Pediatrics, etc.
- **Role & Permission Settings:** Assign user roles with specific access rights to maintain security.
- **System Monitoring:** Track hospital operations in real-time through dashboards.
- **Reports & Analytics:** Generate statistical reports for patient count, revenue, medicine usage, and doctor performance.
- **Backup & Security:** Manage system backups, restore data, and oversee cybersecurity measures.

#### 1.2 Doctor Module

This module is designed for doctors to manage patient consultations efficiently. Key features include:

- **View Appointments:** Access daily/weekly schedules and upcoming patient visits.
- **Patient History Access:** View medical records, previous diagnoses, test results, and prescriptions instantly.
- **Diagnosis & Prescription:** Record patient symptoms, give diagnoses, and prescribe medicines/tests directly into the system.
- **Lab Test Requests:** Send requests to the laboratory and receive results digitally.
- **Treatment Notes:** Maintain detailed treatment history for follow-ups.
- **Emergency Alerts:** Get notified about emergency patient arrivals.

#### 1.3 Receptionist Module

The receptionist handles patient-facing operations such as:

- **Patient Registration:** Add new patient details and generate unique patient IDs.
- **Appointment Booking:** Schedule, reschedule, and cancel appointments based on doctor availability.

**Admission & Discharge:** Record room allocation, admission details, and discharge summaries.

- **Visitor Management:** Maintain a log of visitors for security.
- **Payment Initiation:** Collect consultation fees and forward details to billing.

## 1.4 Pharmacy Module

This module helps in managing medicines and medical supplies:

- **Inventory Management:** Maintain stock levels of medicines, surgical items, and consumables.
- **Prescription Processing:** Issue medicines as per doctor's prescription.
- **Expiry Tracking:** Alert for medicines that are near expiry to avoid wastage.
- **Purchase Management:** Automatically generate purchase orders when stock is low.
- **Sales Reports:** Generate daily, weekly, and monthly sales summaries.

## 1.5 Laboratory Module

Used by lab technicians to manage diagnostic tests:

- **Test Request Management:** Receive test orders from doctors.
- **Sample Tracking:** Record collection, testing, and report preparation stages.
- **Result Upload:** Attach test results to patient records in digital format.
- **Report Notifications:** Notify doctors and patients when results are ready.
- **Test History:** Maintain a database of all tests for auditing and research.

## 1.6 Patient Module

For patient access to services and records:

- **Online Registration & Login:** Patients can create accounts and log in securely.
- **Appointment Booking:** View doctor availability and book appointments.
- **Access to Records:** View prescriptions, test results, and billing history.
- **Notifications:** Receive reminders for appointments, test reports, and medicines.
- **Feedback Submission:** Give feedback or lodge complaints to improve services.

## 1.7 Billing & Accounts Module (*Additional*)

Manages all financial transactions within the hospital:

- **Bill Generation:** Automatically prepare bills for consultations, tests, room charges, and pharmacy purchases.

- **Multi-Payment Options:** Support for cash, credit/debit cards, and online payments.
- **Insurance Integration:** Handle insurance claims and pre-approvals.
- Financial Reports:** Monthly and yearly revenue summaries.

## 1.8 Emergency & Ambulance Management Module (*Additional*)

Specifically for emergency response:

- **Emergency Registration:** Quickly register patients in critical condition without lengthy formalities.
- **Ambulance Tracking:** Assign and track ambulance locations in real-time.
- **Critical Alerts:** Notify doctors and staff immediately about emergency cases.

## (2) Functional and Non-Functional Requirements

### 2.1 Functional Requirements

Functional requirements define the **core operations** that the Hospital Management System must perform to meet its purpose.

#### A. Patient Registration & Records Management

- **Data Storage:** System must store personal details like name, age, gender, address, contact information, and emergency contact numbers.
- **Medical History Tracking:** Maintain complete medical history including previous visits, diagnoses, allergies, surgeries, and ongoing treatments.
- **Automatic Updates:** Patient history should be updated automatically after every consultation, lab test, or treatment.
- **Unique Patient ID:** Generate a unique patient ID for easy tracking across all departments.

#### B. Appointment Management

- **Scheduling:** Patients or receptionists can schedule new appointments with available doctors.
- **Rescheduling/Cancellation:** Ability to reschedule or cancel appointments with automatic updates to doctor schedules.
- **Availability Check:** Show real-time doctor availability to avoid overlapping appointments.
- **Reminders:** Send automated reminders via email/SMS to reduce no-shows.

### C. Billing & Payments

**Bill Generation:** Generate itemized bills covering consultation fees, diagnostic charges, pharmacy purchases, and room charges.

- **Multiple Payment Methods:** Accept cash, credit/debit cards, UPI, and online banking payments.
- **Payment Gateway Integration:** Secure integration with payment platforms for online transactions.
- **Transaction History:** Maintain a log of all payments for accounting and auditing purposes.

### D. Medicine & Stock Management

- **Real-Time Inventory Tracking:** Monitor stock of medicines, surgical items, and consumables.
- **Low Stock Alerts:** Automatic notifications for items reaching minimum stock levels.
- **Expiry Tracking:** Alerts for medicines nearing expiry to avoid wastage.
- **Stock In/Out Records:** Maintain records of all purchases and issues from inventory.

### E. Lab Test Management

- **Test Creation:** Create lab test orders from doctor prescriptions.
- **Sample Tracking:** Record sample collection time, technician details, and progress status.
- **Digital Report Upload:** Upload test results directly to the patient's digital profile.
- **Report Accessibility:** Allow doctors and patients to view results securely through the system.

## 2.2 Non-Functional Requirements

Non-functional requirements define the **quality attributes and constraints** of the system to ensure smooth performance.

### 1. Response Time

- All user actions (such as viewing patient details, booking appointments, or generating bills) should be completed within **less than 2 seconds** to ensure a smooth user experience.

### 2. Scalability

- The system should be able to handle **multiple hospital branches** and **thousands of concurrent users** without performance degradation.

### 3. Security

- Implement **role-based access control** so that users can only access the functions relevant to their role.  
Store all sensitive patient information in an **encrypted format** to ensure data confidentiality.
- Protect the system using **firewalls, intrusion detection systems, and secure login mechanisms**.

### 4. Availability

- Maintain at least **99.9% uptime** so that the hospital operations are never interrupted.

### 5. Backup & Recovery

- Automatic system backups should be scheduled every **24 hours** to prevent data loss.
- Recovery time in case of a system crash should be **less than 1 minute** to resume operations quickly.

### 6. Reliability

- The system should not experience more than **two downtime incidents per year** to ensure dependable service.

S.No	Type	Requirement	Detailed Description
1	Functional	<b>Patient Registration &amp; Records</b>	Store personal details, contact info, and medical history. Automatically update records after every visit. Generate unique patient ID.
2	Functional	<b>Appointment Management</b>	Schedule, reschedule, or cancel appointments. Show doctor availability in real-time. Send automated reminders via SMS/Email.
3	Functional	<b>Billing &amp; Payments</b>	Generate itemized bills for consultations, tests, medicines, and services. Support multiple payment modes. Integrate with payment gateways.
4	Functional	<b>Medicine &amp; Stock Management</b>	Track medicine inventory in real-time. Provide low stock and expiry alerts. Maintain stock in/out records.

5	Functional	<b>Lab Test Management</b>	Create and manage lab test orders. Track samples. Upload reports to patient profiles. Notify doctors and patients when reports are ready.
6	Non-Functional	<b>Response Time</b>	All operations should respond in less than 2 seconds.
7	Non-Functional	<b>Scalability</b>	Handle multiple branches and thousands of concurrent users without performance drop.
8	Non-Functional	<b>Security</b>	Implement role-based access control, encrypt sensitive data, and secure the system with firewalls and intrusion detection.
9	Non-Functional	<b>Availability</b>	Maintain at least 99.9% uptime to ensure continuous hospital operations.
10	Non-Functional	<b>Backup &amp; Recovery</b>	Automatic backup every 24 hours. Recovery within 1 minute in case of system crash.
11	Non-Functional	<b>Reliability</b>	System downtime should not exceed 2 incidents per year.

## Data Modeling

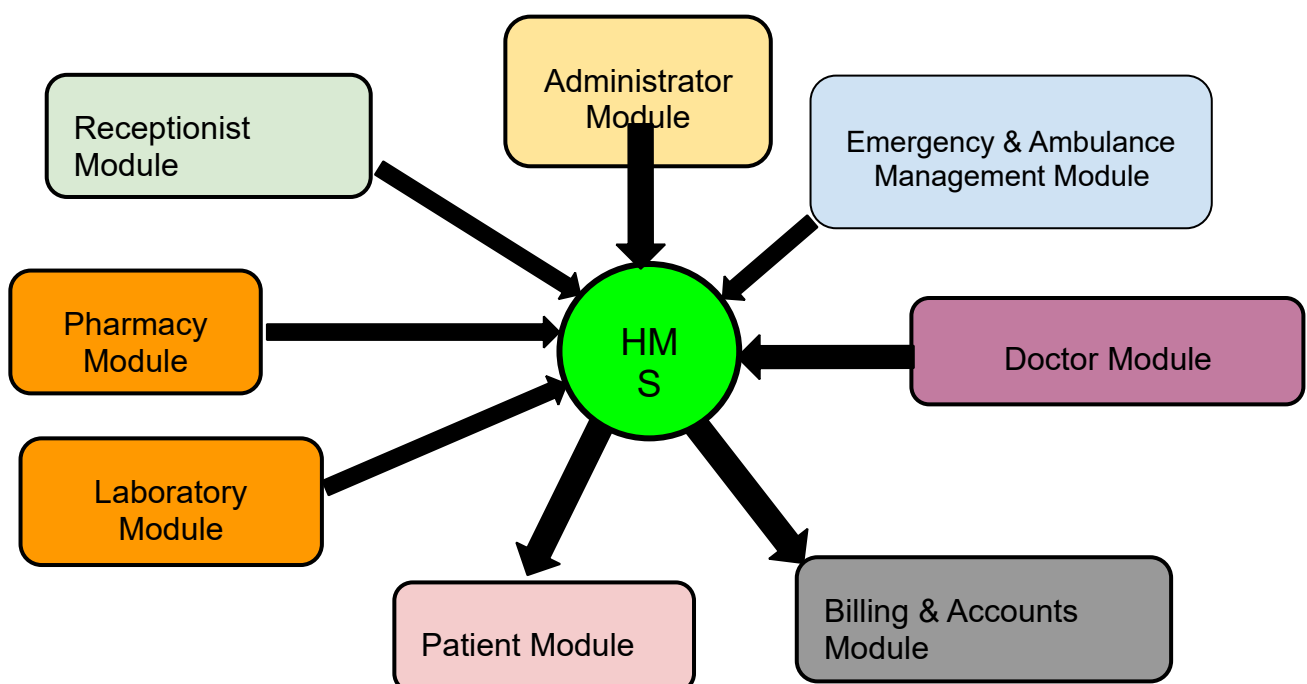
### Product Perspective

The **Hospital Management System** will be deployed within the hospital's **internal secure network** for in-house operations, but will also have a **secure web-based interface** for remote access by doctors and patients.

It will be integrated with:

- **Laboratory Systems:** For automatic transfer of test requests and results.
- **Pharmacy Systems:** For real-time medicine inventory management.
- **Billing Systems:** For seamless invoice generation and payment tracking.

The system will provide **role-specific dashboards** for administrators, doctors, receptionists, lab technicians, pharmacists, and patients, ensuring an intuitive and user-friendly experience for each role.



## Experiment – 3

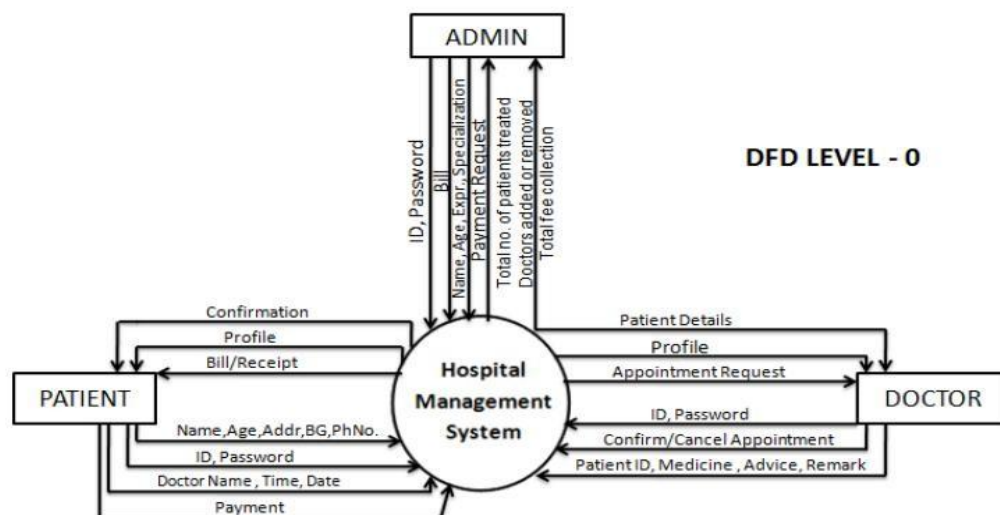
**Aim: To draw different levels for DFD.**

A DFD diagram is a Data Flow Diagram, a graphical tool that visually maps out the flow of data through a system or business process. Using standardized symbols, it shows where data enters the system, how it's transformed, where it's stored, and where it goes, helping to understand system functionality, identify inefficiencies, and improve processes.

**Level 0 DFD:**

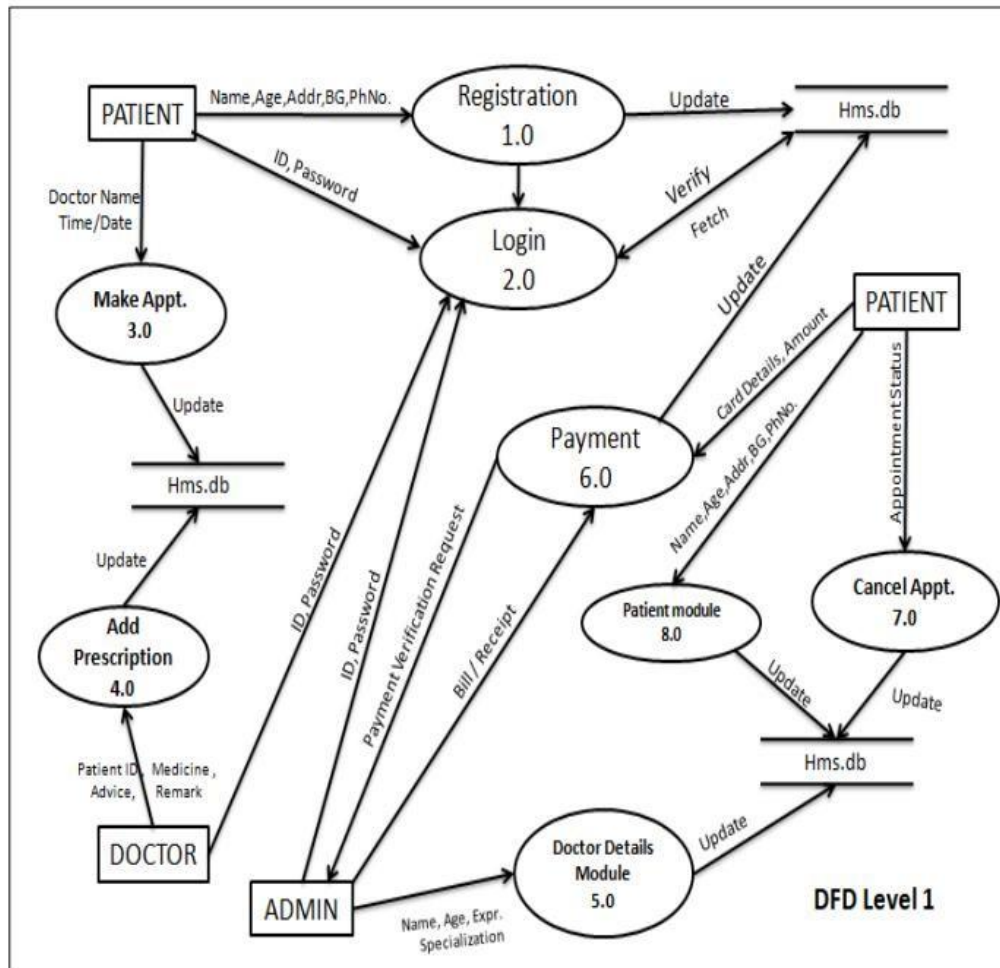
The context diagram provides the highest-level view of the system, showing the system as a single process and its interactions with external entities.

### CONTEXT LEVEL DIAGRAM



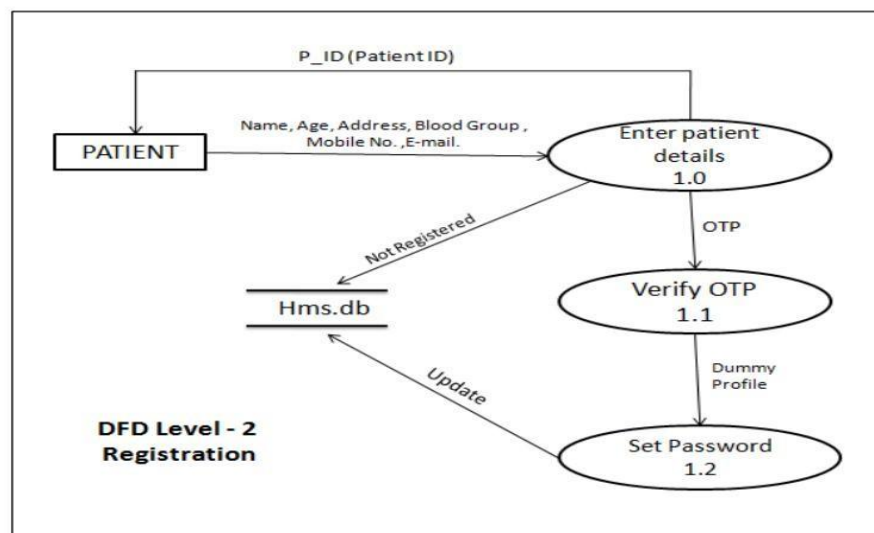
**Level 1 DFD: First Level Diagram**

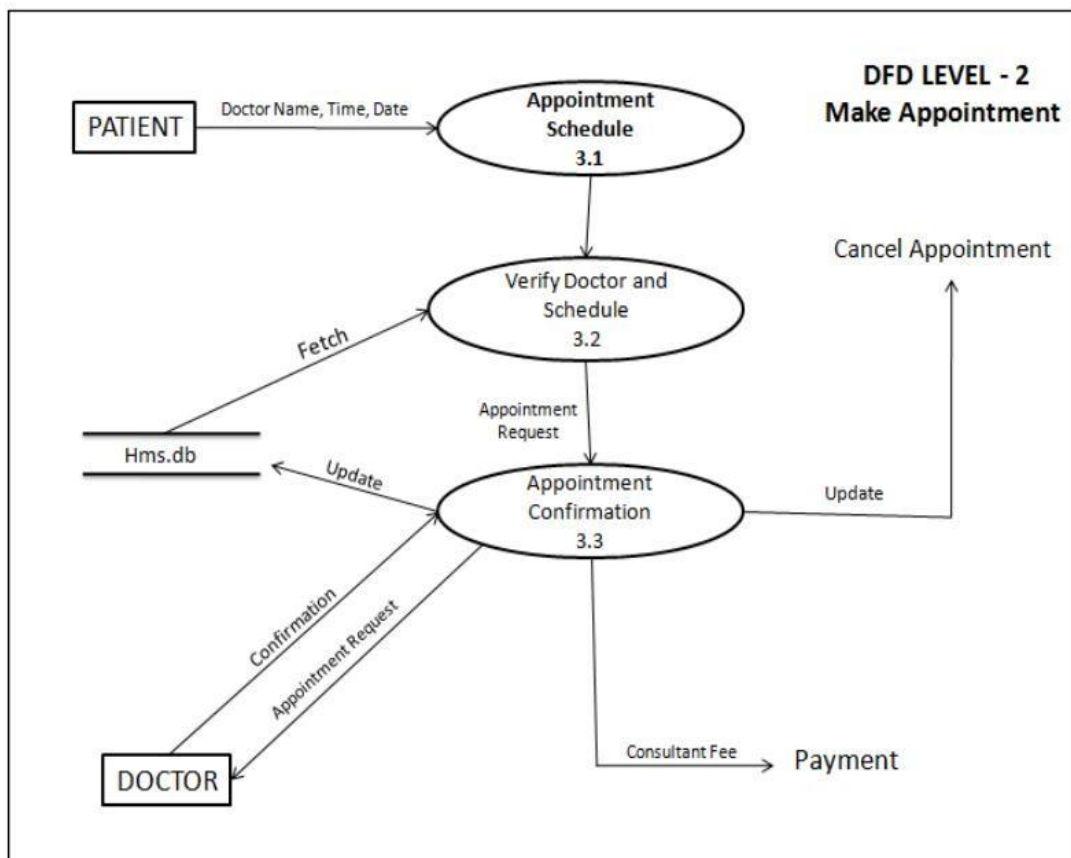
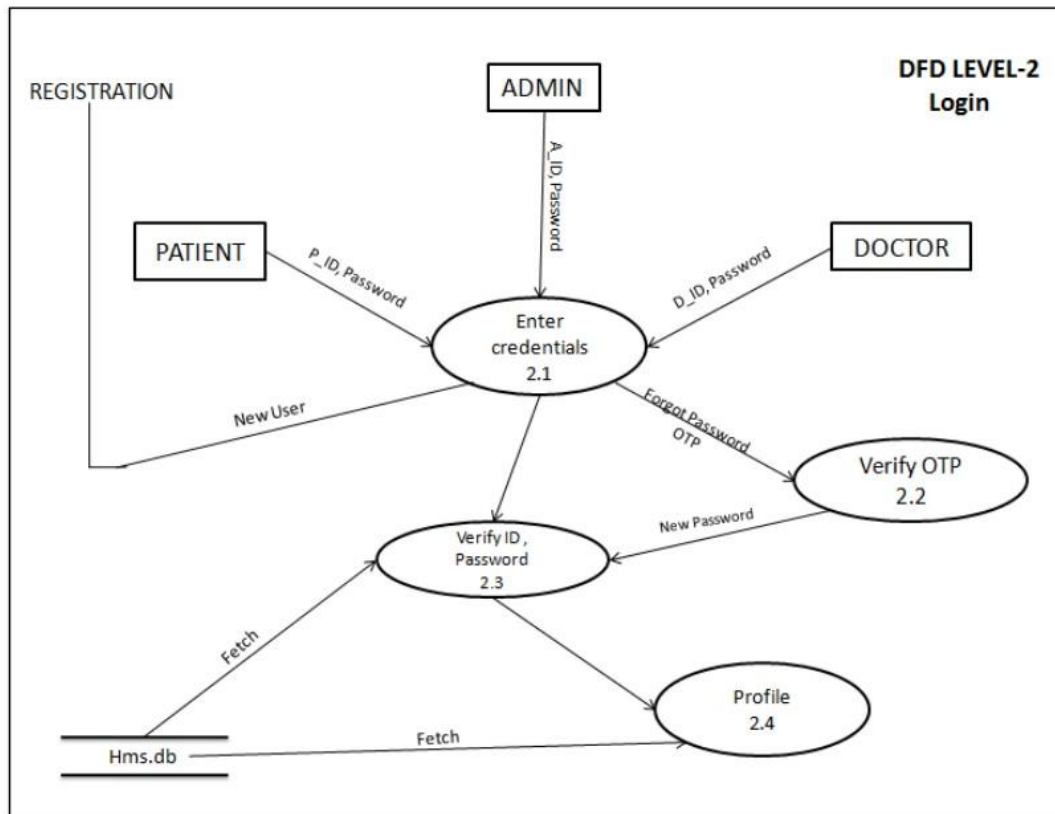
The Level 1 DFD breaks down the single process of the context diagram into its major subprocesses. This level shows the main functions performed by the system and the data flows between them and external entities.

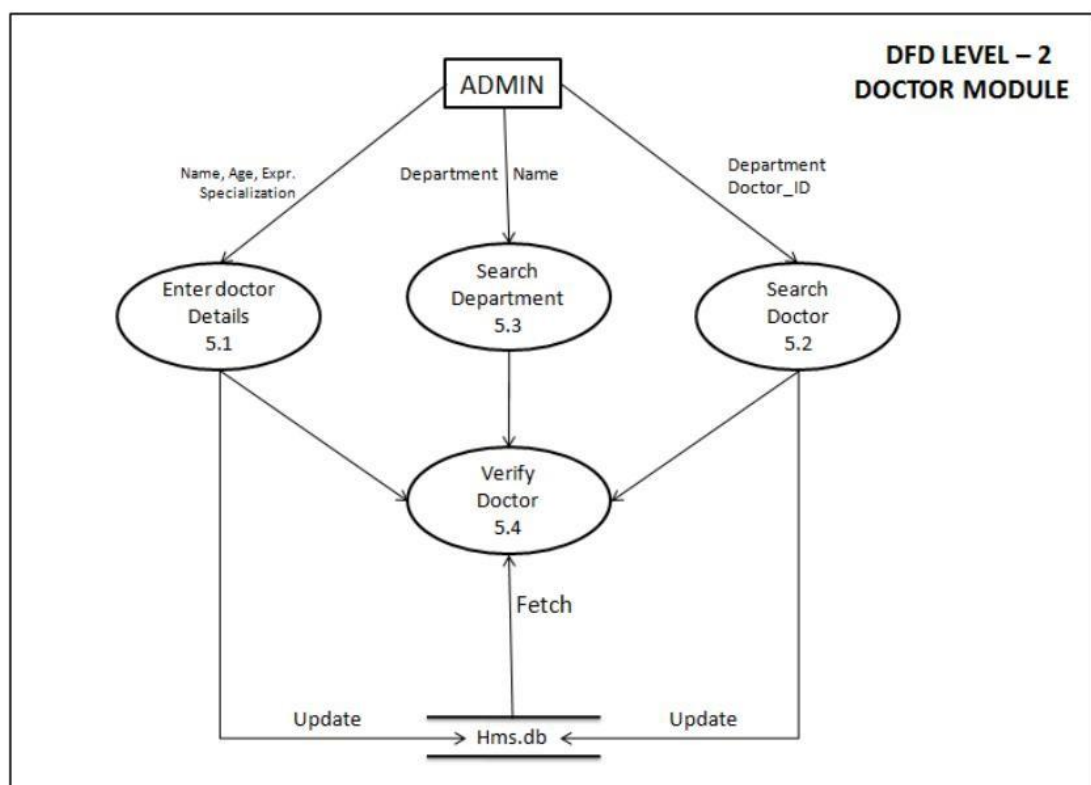
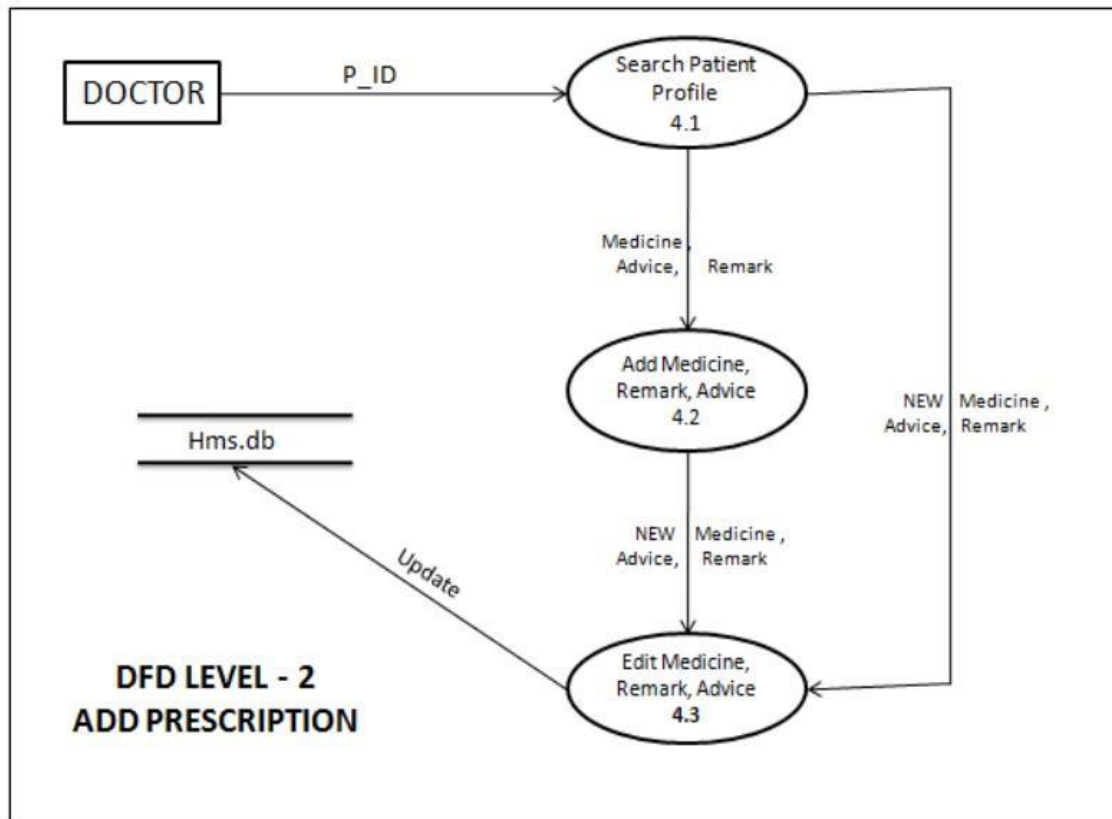


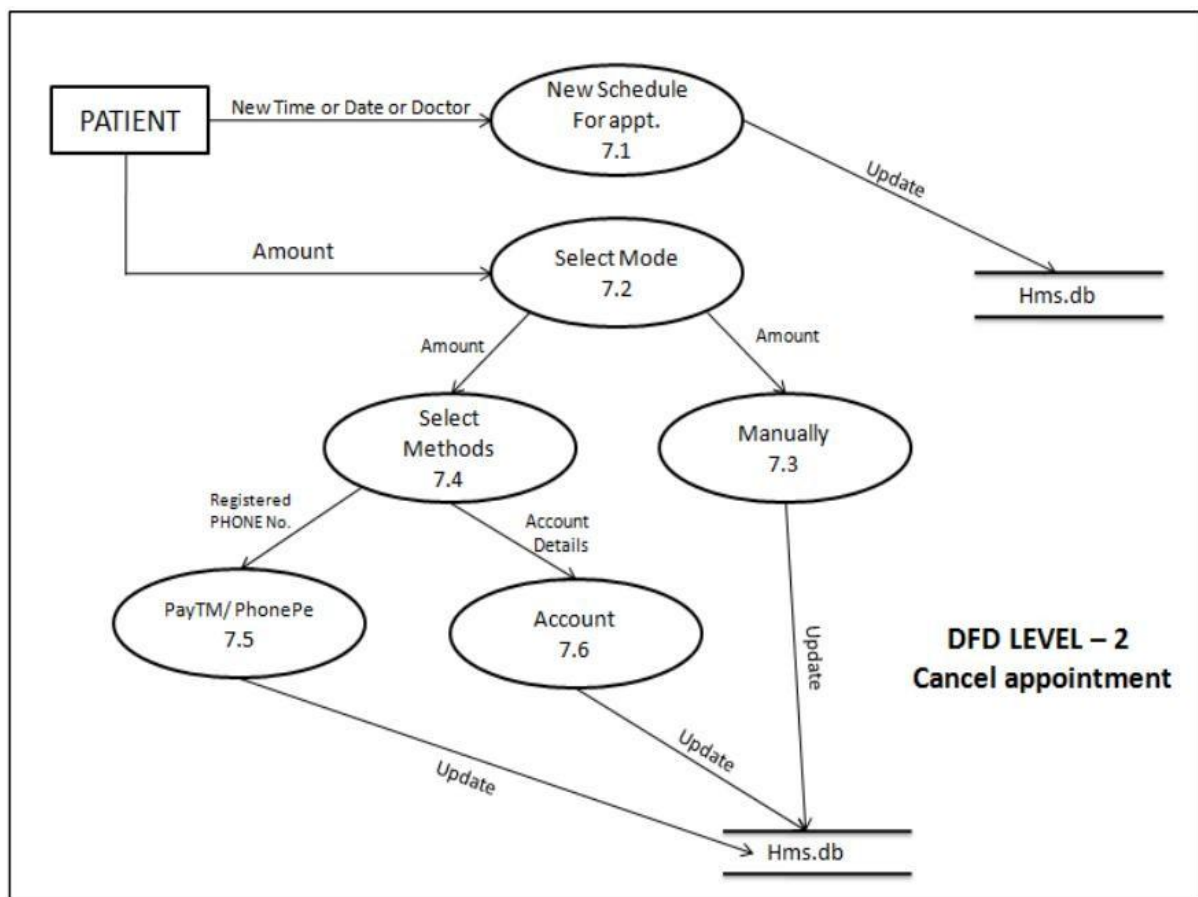
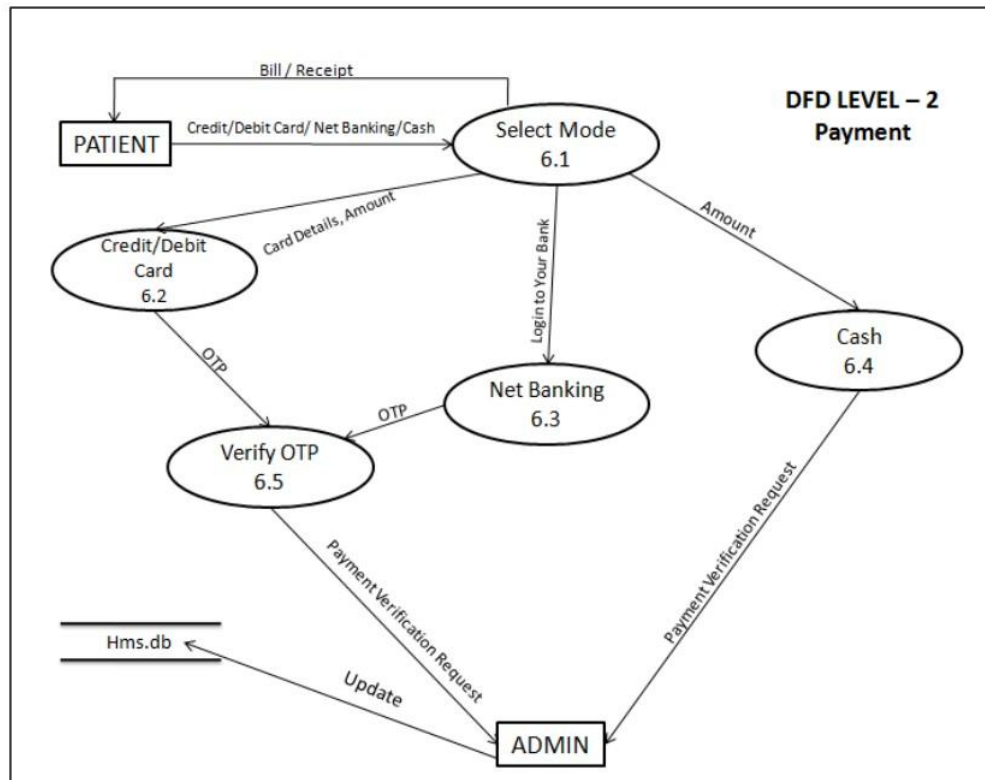
### Level 2 DFD: Second Level Diagram

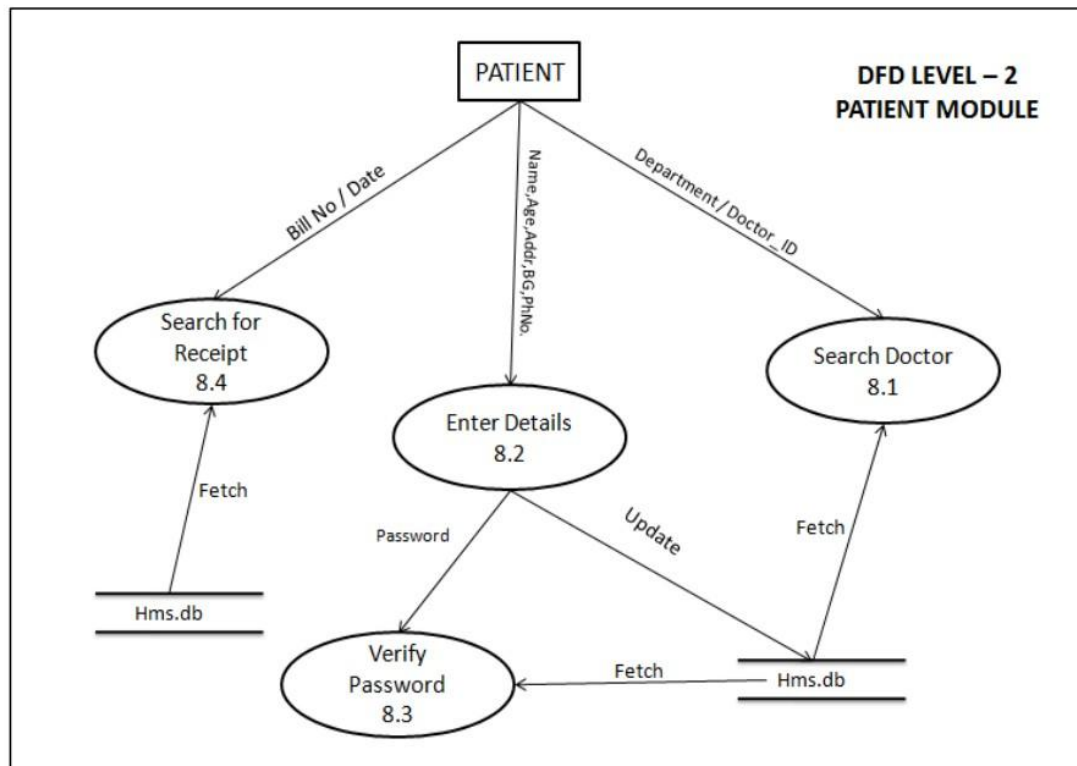
The Level 2 DFD further decomposes the Level 1 processes into more detailed subprocesses, providing a deeper understanding of the system's internal workings.











## Experiment – 4

**Aim: To develop an ER diagram.**

**ER Diagram** (Entity-Relationship Diagram) is a powerful tool to analyze and design the structure of a database. It shows the relationships between different entities and their attributes. An ER Model provides a means of communication and clarity in system design.

The **Hospital Management System** database keeps track of patients, doctors, appointments, staff, billing, and departments with the following considerations:

- The system keeps track of **hospital staff** with a secure authentication system comprising login ID and password.
- Staff maintain records of **patients**, including their personal information and medical history.
- Each **doctor** is assigned a unique doctor ID, specialization, years of experience, and availability schedule.
- **Patients** are registered with a patient ID, name, gender, contact numbers (multiple allowed), email, and address.
- **Appointments** are scheduled between patients and doctors, stamped with appointment date and time, and status (scheduled/completed/cancelled).
- Staff also manage **billing**, which includes bill number, patient ID, date of billing, list of services, and total amount.
- The hospital maintains **departments** like Cardiology, Neurology, Orthopedics, etc., each with a department ID, department name, and assigned doctors.
- Staff generate various reports about patients, doctors, appointments, and billing for management purposes.

This Hospital ER Diagram illustrates key information about the hospital, including entities such as Staff, Patients, Doctors, Departments, Billing, Appointments, and the Authentication System. It allows for clear understanding of the relationships between entities.

**Attributes:**

## 1. Patient

- P-ID (Primary Key)
  - Name
  - Age
  - Gender
  - DOB (Date of Birth)
  - Mob-No (Mobile Number)
- 

## 2. Doctor (specialization of Employee)

- Dept (Department)
  - Qualification
- 

## 3. Employee (Superclass)

- E-ID (Employee ID) (Primary Key)
  - Name
  - Address
  - City
  - State
  - Pin-no
  - Mob-No (Mobile Number)
  - Salary
  - Sex
- 

## 4. Nurse (specialization of Employee)

- [Inherited attributes from Employee]
  - (connected to Dept and Qualification via Doctor relationship as shown)
- 

## 5. Receptionist (specialization of Employee)

- [Inherited attributes from Employee]
- 

## 6. Bills

- B-ID (Bill ID) (Primary Key)
  - Amount
  - P-ID (Patient ID - Foreign Key)
- 

## 7. Test Report

- R-ID (Report ID) (Primary Key)
  - Test-Type
  - Result
  - P-ID (Patient ID - Foreign Key)
- 

## 8. Rooms

- R-ID (Room ID) (Primary Key)
  - Type (e.g., ICU, General, Private)
  - Capacity
  - Availability
- 

## 9. Records

- Record-no (Primary Key)
  - App-no (Appointment Number)
- 

**Relationships:**

- **Patient:**

- Consults with Doctor (M:N)
- Pays Bills (1:N)
- Has Test Report (1:N)
- Assigned to Rooms (N:1)

- **Doctor:**

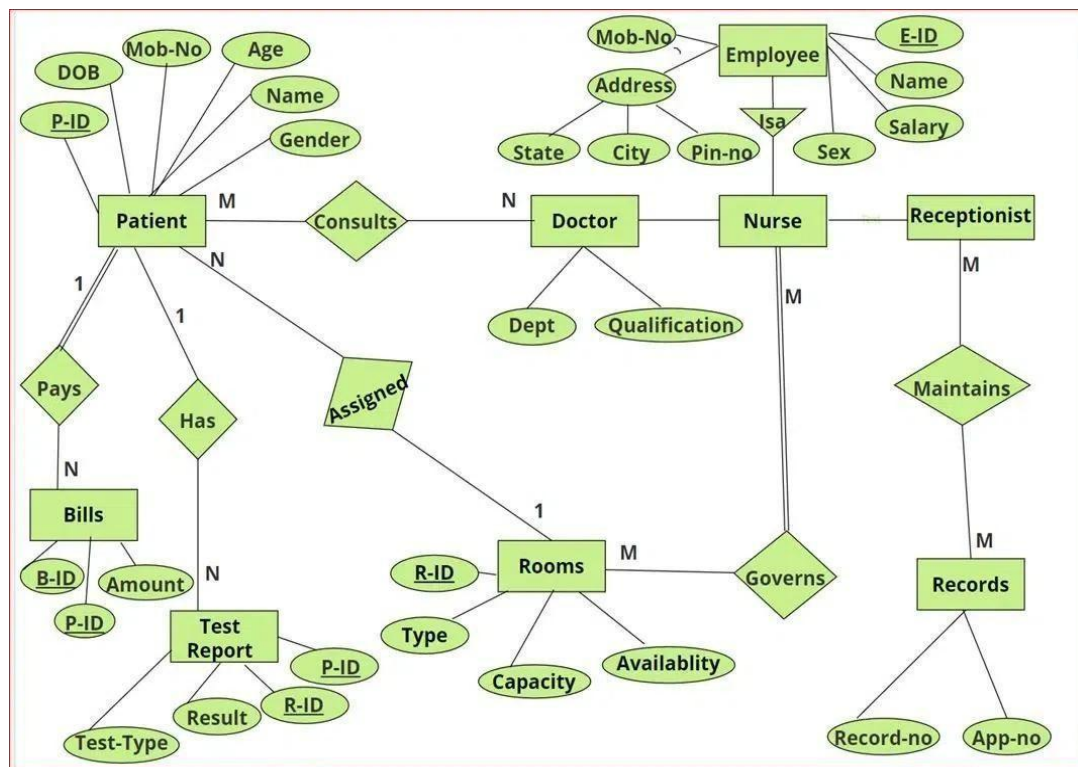
- Works in a Dept (Department)
- Has Qualification

- **Nurse:**

- Governs Rooms (M:M)

- **Receptionist:**

- Maintains Records (M:M)



### Relationship between Entities in Hospital Management System:

- A **patient** can consult **many doctors**, but a **doctor** can also consult **many patients**. The relationship is **M:N** (Many-to-Many).
- A **doctor** is assigned to one **department**, but a **department** can have many **doctors**. The relationship is **1:N** (One-to-Many).
- **Staff** keeps track of **patients**. The relationship is **M:N** (Many-to-Many).
- A **patient** can have multiple **bills**, but each **bill** belongs to only one **patient**. The relationship is **1:N** (One-to-Many).
- A **patient** can have multiple **test reports**, but each **test report** is associated with only one **patient**. The relationship is **1:N** (One-to-Many).
- A **nurse** can govern multiple **rooms**, and a **room** can be governed by multiple **nurses**. The relationship is **M:N** (Many-to-Many).
- A **receptionist** maintains multiple **records**, and each **record** is maintained by one **receptionist**. The relationship is **1:N** (One-to-Many). The **authentication system** provides login access to multiple **staff** members. The relationship is **1:N** (One-to-Many).

## Experiment – 5

### **Aim: To draw a user case diagram.**

A Use Case Diagram is a visual representation that illustrates the interactions between users (actors) and a system. It captures the functional requirements of a system, showing how different users engage with various use cases, or specific functionalities, within the system. Use case diagrams provide a high-level overview of a system's behavior, making them useful for stakeholders, developers, and analysts to understand how a system is intended to operate from the user's perspective, and how different processes relate to one another. They are crucial for defining system scope and requirements.

### **Use Case Diagram Notations**

UML notations provide a visual language that enables software developers, designers, and other stakeholders to communicate and document system designs, architectures, and behaviors in a consistent and understandable manner.

#### **1. Actors**

Actors are external entities that interact with the system. These can include users, other systems, or hardware devices. In the context of a Use Case Diagram, actors initiate use cases and receive the outcomes. Proper identification and understanding of actors are crucial for accurately modeling system behavior.

#### **2. Use Cases**

Use cases are like scenes in the play. They represent specific things your system can do. In the online shopping system, examples of use cases could be "Place Order," "Track Delivery," or "Update Product Information". Use cases are represented by ovals.

#### **3. System Boundary**

The system boundary is a visual representation of the scope or limits of the system you are modeling. It defines what is inside the system and what is outside. The boundary helps to establish a clear distinction between the elements that are part of the system and those that are external to it. The system boundary is typically represented by a rectangular box that surrounds all the use cases of the system.

### **Identify Actors**

Actors are the users or external systems that interact with the hospital management system.

- **Primary Actors:**

- **Patient:** Registers, books appointments, views medical history.
- **Doctor:** Views patient records, prescribes medication, updates medical history.
- **Receptionist:** Manages appointments, patient registration, and billing.

### **2. Identify Use Cases**

Use cases represent the specific functionalities or services the system provides to the actors.

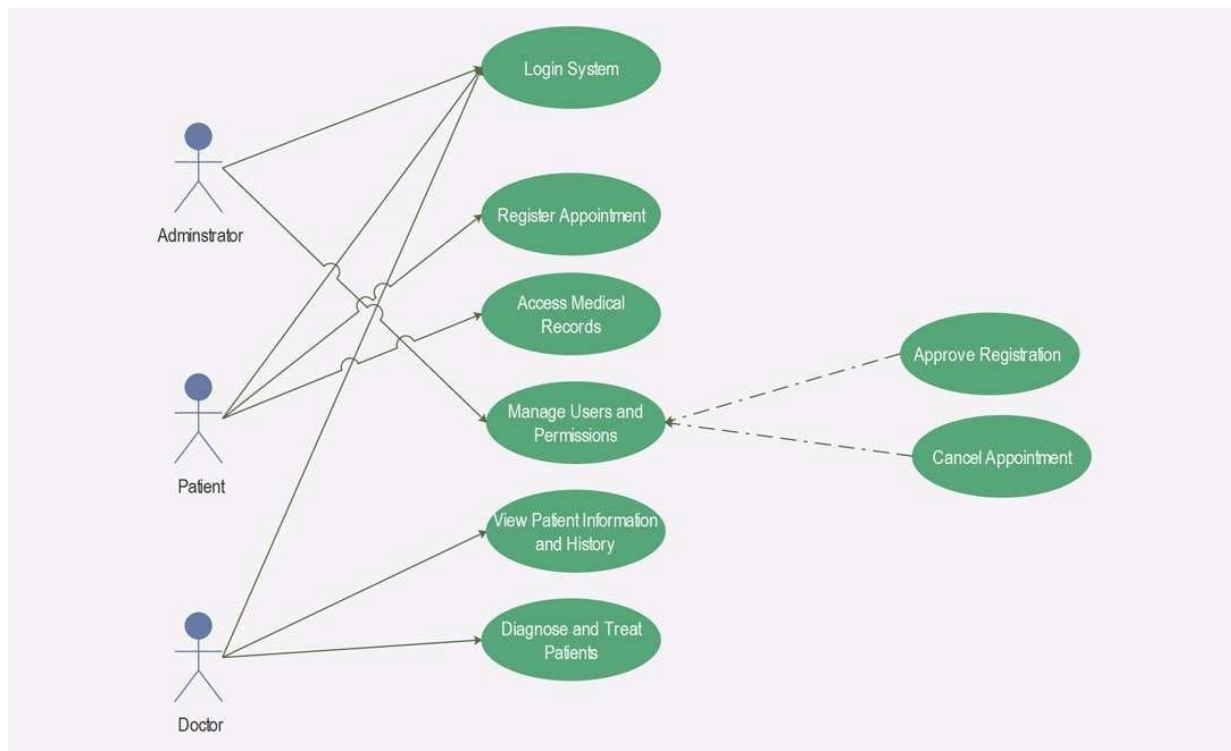
- **Patient Use Cases:**

- Register Patient
- Book Appointment
- View Medical History
- Pay Bill
- **Doctor Use Cases:**
  - View Patient Records
  - Prescribe Medication
  - Update Medical History
  - Request Lab Tests
- **Receptionist Use Cases:**
  - Schedule Appointment
  - Register New Patient
  - Generate Bill
- **Admin Use Cases:**
  - Manage Users (Doctors, Staff)
  - Manage Departments
  - Generate Reports

### 3. Define Relationships

Connect actors to the use cases they interact with using solid lines.

- **Include Relationship:** Use dashed arrows with <<include>> to show mandatory sub-tasks.
- **Extend Relationship:** Use dashed arrows with <<extend>> to show optional functionalities.
- **Generalization:** Use solid arrows with hollow arrowheads to show inheritance between actors.



## Experiment – 6

**Aim: To draw a sequence diagram and collaboration diagrams.**

### Sequence diagram

Sequence diagrams are a type of UML (Unified Modeling Language) diagram that visually represent the interactions between objects or components in a system over time. They focus on the order and timing of messages or events exchanged between different system elements. The diagram captures how objects communicate with each other through a series of messages, providing a clear view of the sequence of operations or processes.

### Sequence Diagram Notations

#### 1. Actors

An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram.

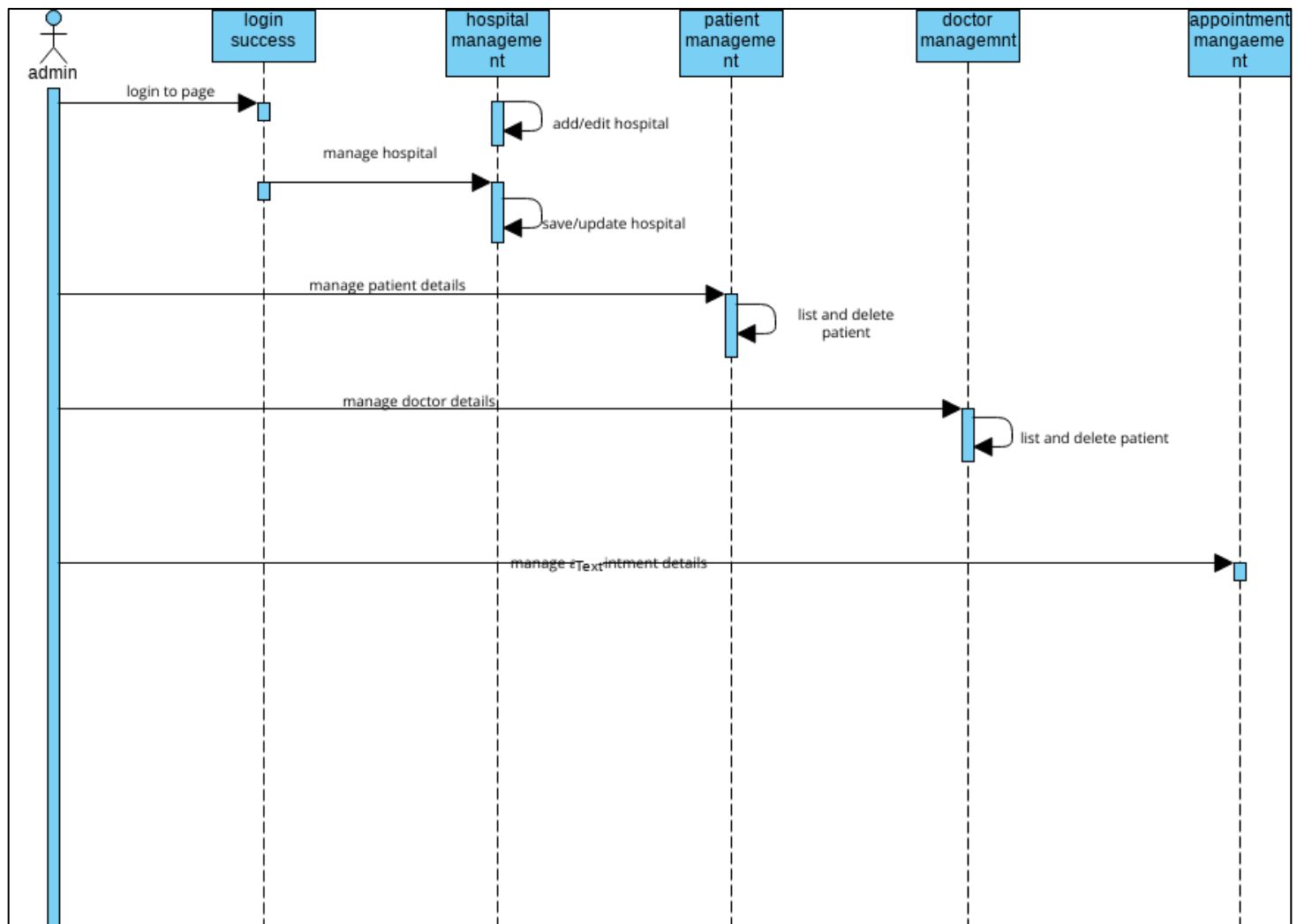
#### 2. Lifelines

A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram.

#### 3. Messages

Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline.

- We represent messages using arrows.
- Lifelines and messages form the core of a sequence diagram.



## Collaboration Diagrams

A Collaboration Diagram is a type of Interaction Diagram that visualizes the interactions and relationships between objects in a system. It shows how objects collaborate to achieve a specific task or behavior. Collaboration diagrams are used to model the dynamic behavior of a system and illustrate the flow of messages between objects during a particular scenario or use case.

## Components and their Notations in Collaboration Diagrams

### 1. Objects/Participants

Objects are represented by rectangles with the object's name at the top. Each object participating in the interaction is shown as a separate rectangle in the diagram. Objects are connected by lines to indicate messages being passed between them.

### 2. Multiple Objects

Multiple objects are represented by rectangles, each with the object's name inside, and interactions between them are shown using arrows to indicate message flows.

### **3. Actors**

They are usually shown at the top or side of the diagram. Actors indicate their involvement in the interactions with the system's objects or components. They are connected to objects through messages, showing the communication with the system.

### **4. Messages**

Messages represent communication between objects. Messages are shown as arrows between objects, indicating the flow of communication. Each message may include a label indicating the type of message.

### **5. Self-Message**

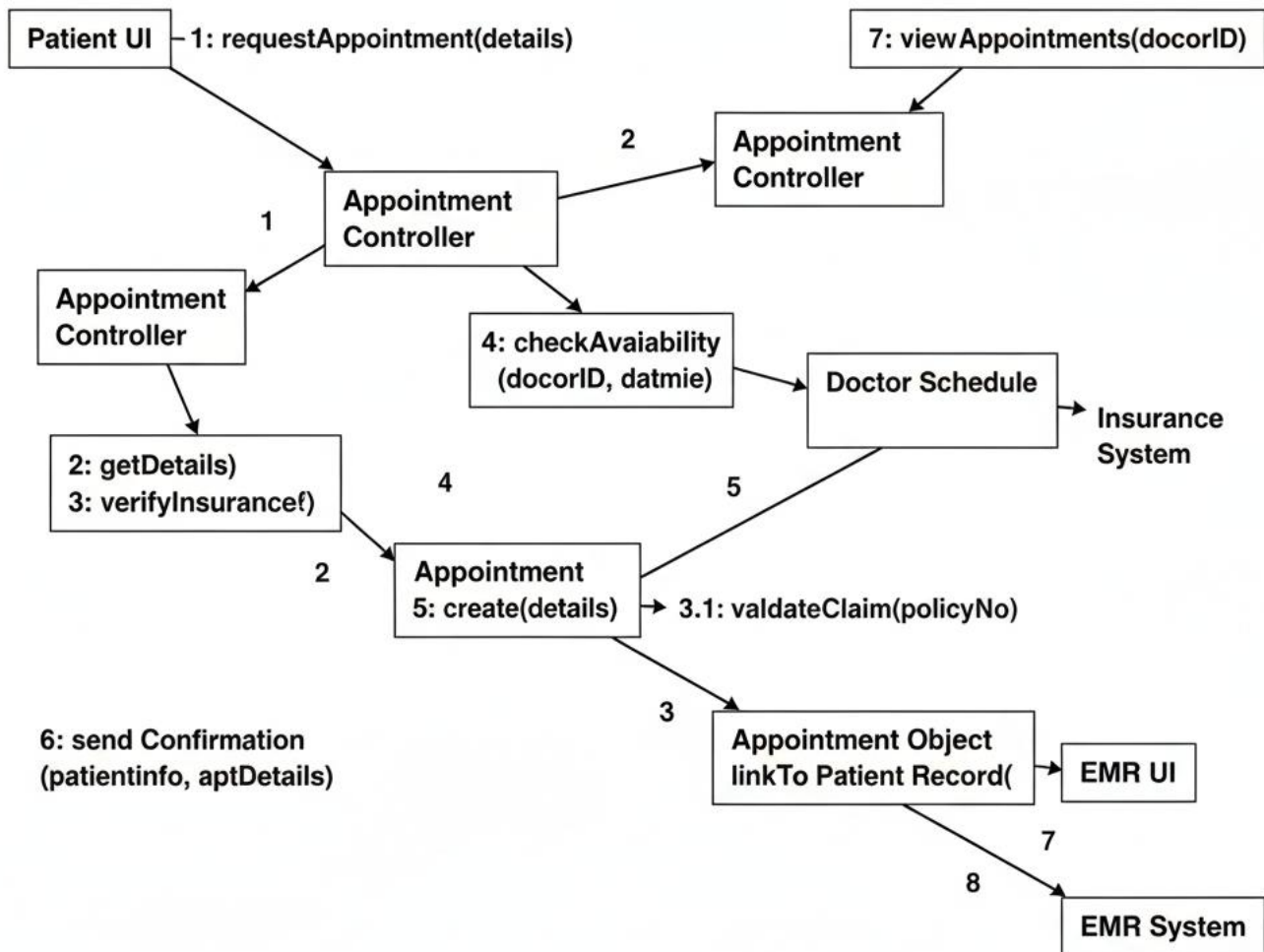
This is a message that an object sends to itself. It represents an action or behavior that the object performs internally without involving any other objects. Self-messages are useful for modeling scenarios where an object triggers its own methods or processes.

### **6. Links**

Links represent associations or relationships between objects. Links are shown as lines connecting objects, with optional labels to indicate the nature of the relationship. Links can be uni-directional or bi-directional, depending on the nature of the association.

### **7. Return Messages**

Return messages represent the return value of a message. They are shown as dashed arrows with a label indicating the return value. Return messages are used to indicate that a message has been processed and a response is being sent back to the calling object.



## Experiment – 7

### Aim: To draw a class diagram

A class diagram visually represents the structure of a system by showing its classes, attributes, methods, and the relationships between them.

1. Helps everyone involved in a project—like developers and designers—understand how the system is organized and how its components interact.
2. Helps to communicate and document the structure of the software.

### UML Class Notation

#### 1. Class Name:

- The name of the class is typically written in the top compartment of the class box and is centered and bold.

#### 2. Attributes:

- Attributes, also known as properties or fields, represent the data members of the class. They are listed in the second compartment of the class box and often include the visibility (e.g., public, private) and the data type of each attribute.

#### 3. Methods:

- Methods, also known as functions or operations, represent the behavior or functionality of the class. They are listed in the third compartment of the class box and include the visibility (e.g., public, private), return type, and parameters of each method.

#### 4. Visibility Notation:

- Visibility notations indicate the access level of attributes and methods. Common visibility notations include:
  - + for public (visible to all classes)
  - - for private (visible only within the class)
  - # for protected (visible to subclasses)
  - ~ for package or default visibility (visible to classes in the same package)

### Parameter Directionality

#### • In (Input):

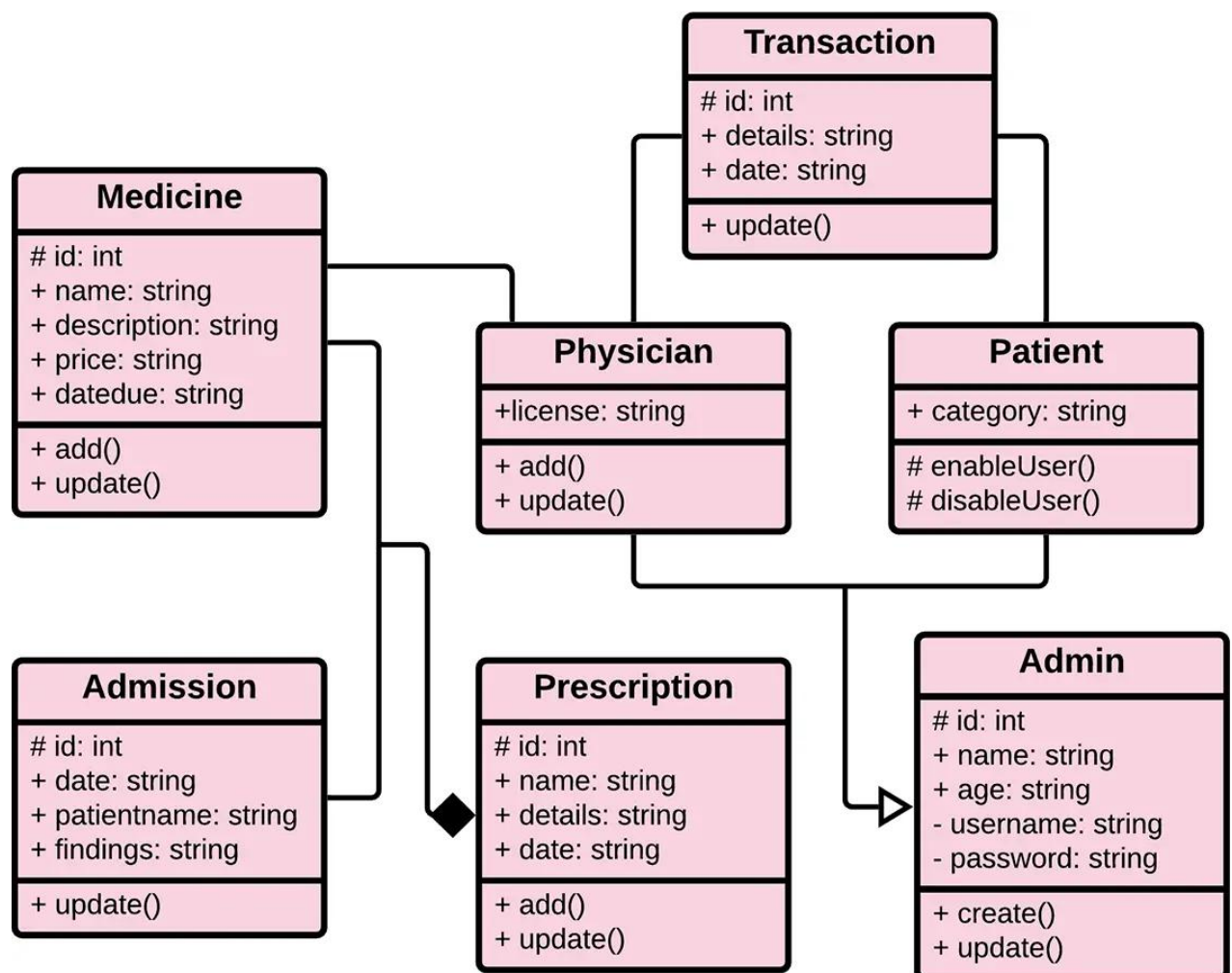
- An input parameter is a parameter passed from the calling object (client) to the called object (server) during a method invocation.
- It is represented by an arrow pointing towards the receiving class (the class that owns the method).

- **Out (Output):**

- An output parameter is a parameter passed from the called object (server) back to the calling object (client) after the method execution.
- It is represented by an arrow pointing away from the receiving class.

- **InOut (Input and Output):**

- An InOut parameter serves as both input and output. It carries information from the calling object to the called object and vice versa.
- It is represented by an arrow pointing towards and away from the receiving class.



## Experiment – 8

**Aim: To draw a Gantt chart and Network diagram.**

Generalized Activity Normalization Time Table (GANTT) chart is type of chart in which series of horizontal lines are present that show the amount of work done or production completed in given period of time in relation to amount planned for those projects.

**Gantt chart represents following things :**

- All the tasks are listed at leftmost column.
- The horizontal bars indicate or represent required time by corresponding particular task.
- When occurring of multiple horizontal bars takes place at same time on calendar, then that means concurrency can be applied for performing particular tasks.
- The diamonds indicate milestones.

### Timeline plan for integrating hospital management system

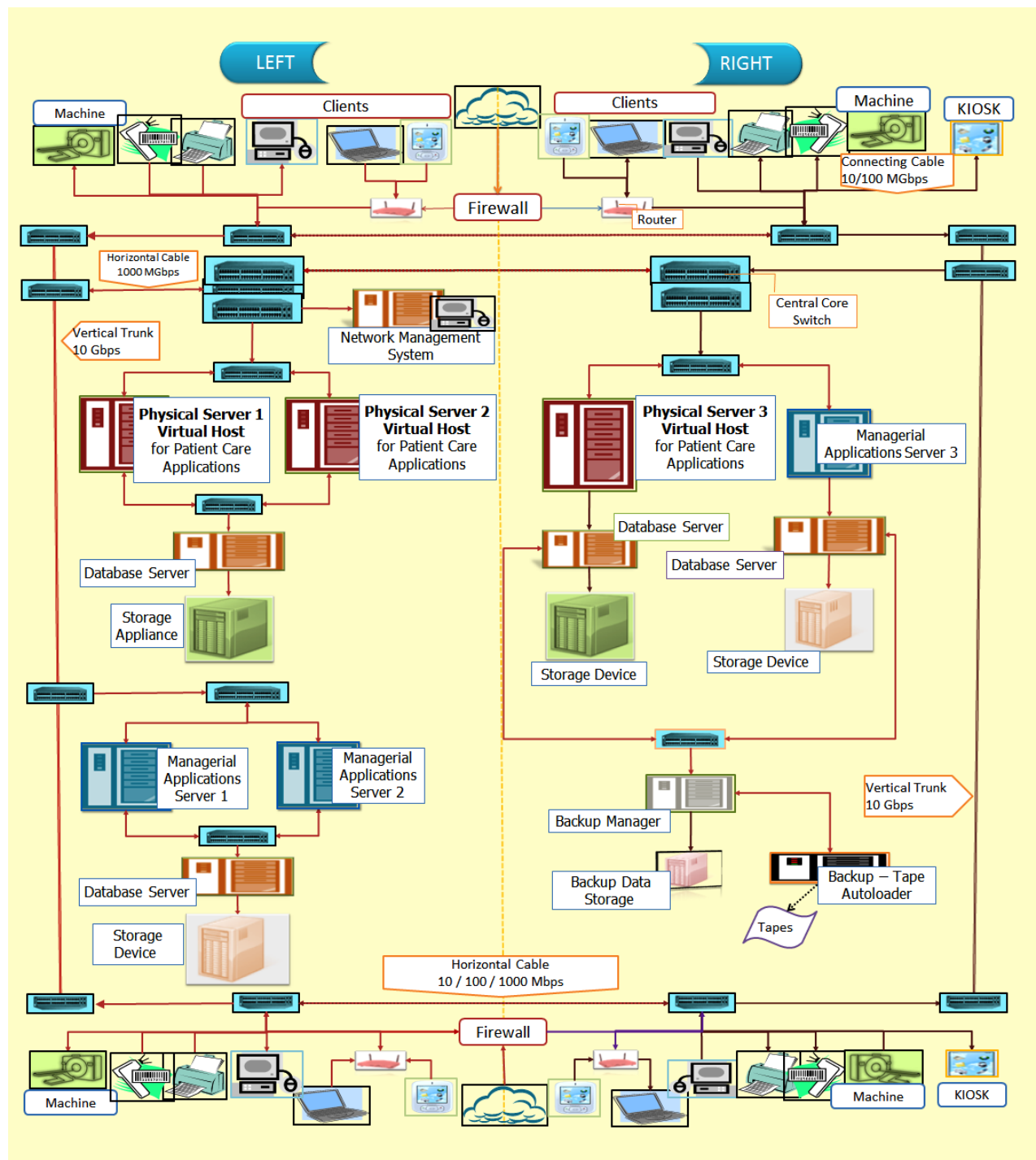
The following slide showcases phases of hospital management system to maintain electronic patient record. It presents information related to receiving implementation idea, establishing structure, etc.



This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

## Network diagram

A network diagram is a visual representation of project tasks and their dependencies, crucial in project management for planning and scheduling. It illustrates the flow of work, sequence of tasks, and relationships between them, aiding project managers in understanding project complexity and optimizing resource allocation. By providing a clear overview of the project structure, network diagrams facilitate effective communication, decision-making, and risk management throughout the project lifecycle.



## Experiment – 9

**Aim: To draw a Structured chart.**

Structure Chart represents the hierarchical structure of modules. It breaks down the entire system into the lowest functional modules and describes the functions and sub-functions of each module of a system in greater detail.

### Symbols in Structured Chart

#### 1. Module

It represents the process or task of the system. It is of three types:

- **Control Module:** A control module branches to more than one submodule.
- **Sub Module:** Sub Module is a module which is the part (Child) of another module.
- **Library Module:** Library Module are reusable and invokable from any module.

#### 2. Conditional Call

It represents that control module can select any of the sub module on the basis of some condition.

#### 3. Loop (Repetitive call of module)

It represents the repetitive execution of module by the sub module. A curved arrow represents a loop in the module.

#### 4. Data Flow

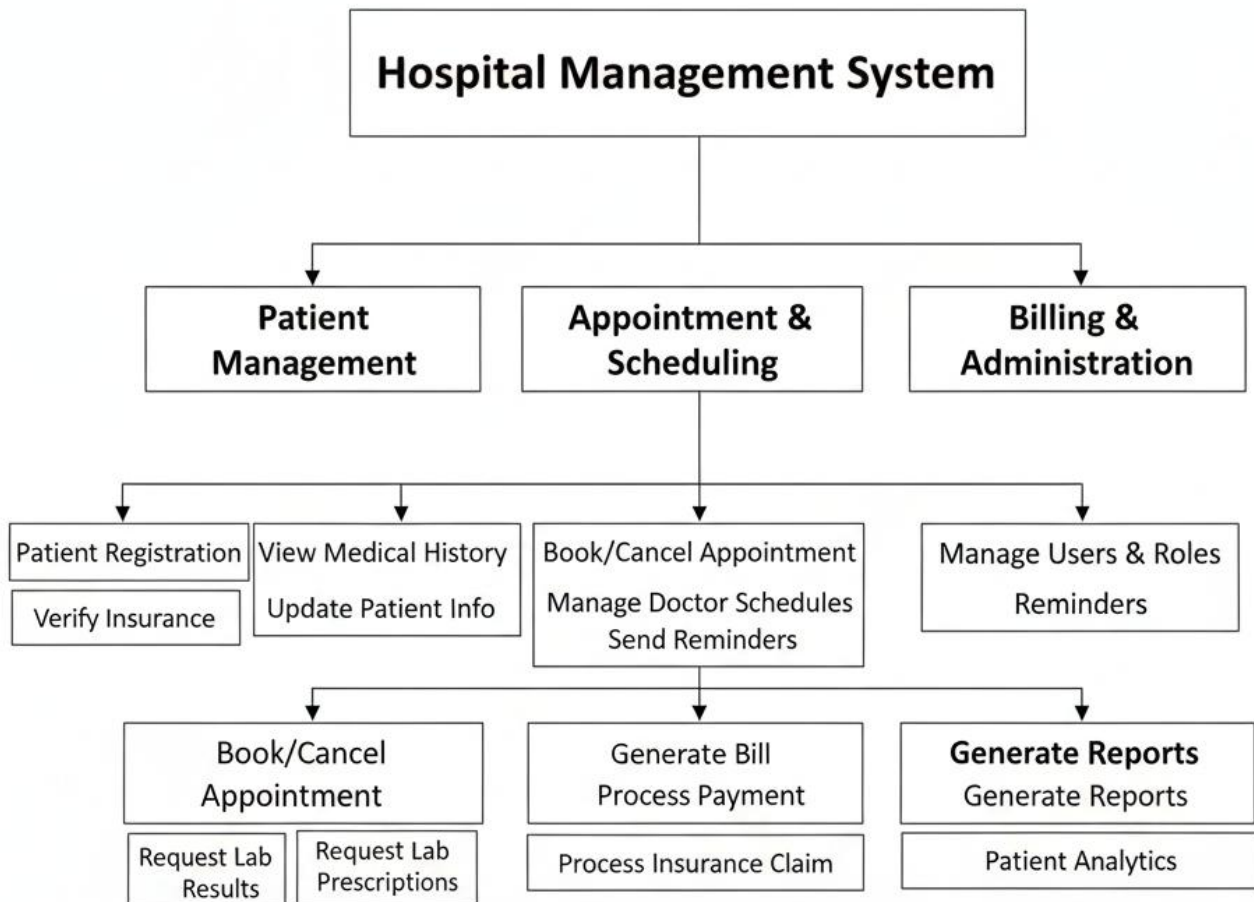
It represents the flow of data between the modules. It is represented by a directed arrow with an empty circle at the end.

#### 5. Control Flow

It represents the flow of control between the modules. It is represented by a directed arrow with a filled circle at the end.

#### 6. Physical Storage

It is that where all the information are to be stored.



## Experiment – 10

### **Aim: Development of design document.**

Software Design Document is a written document that provides a description of a software product in terms of architecture of software with various components with specified functionality.

The design specification addresses different aspects of the design model and is completed as the designer refines his representation of the software. These design documents are written by software engineers/designers or project managers and further passed to the software development team to give them an overview of what needs to be built and how.

### **Typical components of a design document**

- Introduction: A summary of the project and its goals.
- Overview: A high-level description of the design.
- Technical specifications: Details on the architecture, components, and functionality.
- User interface (UI) and user experience (UX) design: How the user will interact with the product.
- Testing strategy: A plan for how the product will be tested to ensure it meets requirements.
- Roles and responsibilities: Who is responsible for each part of the project.
- Milestones and timeline: Key development milestones and a project schedule.
- Glossary: Definitions of technical terms used in the document.

## **Software Design Document: Hospital Management System**

### **1. Introduction**

This document outlines the design and architecture for a comprehensive Hospital Management System (HMS). The HMS aims to streamline various hospital operations, improve patient care, and enhance administrative efficiency.

#### **1.1 Purpose**

The purpose of this document is to provide a detailed overview of the HMS's design, including its architecture, modules, database schema, and technical specifications. It serves as a guide for development, implementation, and maintenance teams.

#### **1.2 Scope**

The HMS will cover the following key areas:

- **Patient Management:** Registration, admissions, discharges, and patient records.
- **Appointment Scheduling:** Doctor appointments, resource allocation.
- **Billing and Invoicing:** Services, medication, and payment processing.
- **Inventory Management:** Pharmacy and medical supplies.
- **Reporting and Analytics:** Operational and financial reports.

## 2. System Architecture

The HMS will adopt a modular, layered architecture to ensure scalability, maintainability, and extensibility.

### 2.1 High-Level Architecture

The system will consist of the following main layers:

- **Presentation Layer:** User interfaces for different user roles (patients, doctors, administrators).
- **Application Layer:** Business logic, service orchestration, and API endpoints.
- **Data Access Layer:** Abstraction for database interactions.
- **Database Layer:** Relational database management system.

### 2.2 Module Breakdown

The HMS will be broken down into the following modules:

- **Patient Registration Module:** Manages patient demographics and medical history.
- **Appointment Management Module:** Handles scheduling and tracking of appointments.
- **Billing Module:** Processes all financial transactions.
- **Pharmacy Module:** Manages medication inventory and dispensing.
- **Laboratory Module:** Integrates with laboratory equipment and manages test results.
- **Reporting Module:** Generates various reports for analysis.

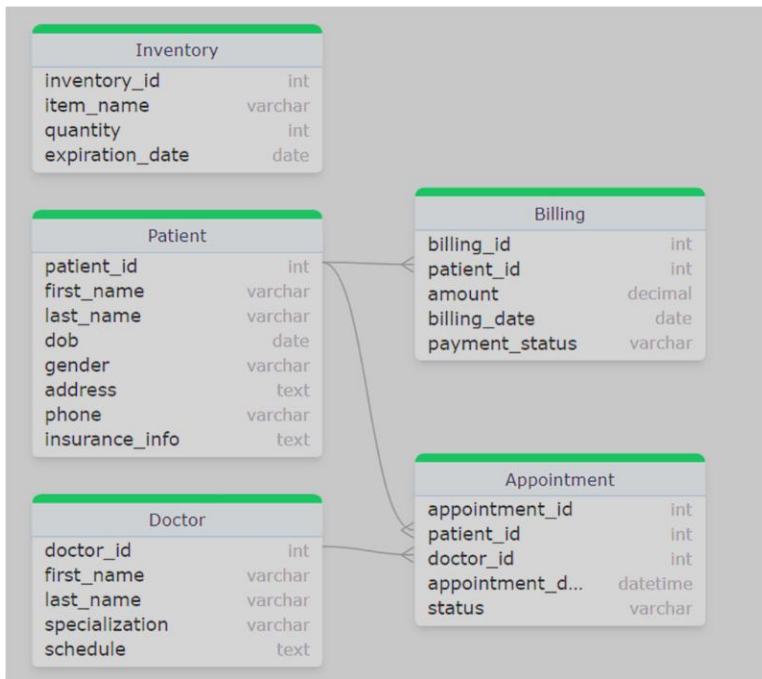
## 3. Data Design

### 3.1 Entity-Relationship Diagram (ERD)

[An Entity-Relationship Diagram illustrating the relationships between patient, doctor, appointment, billing, and inventory entities in the hospital management system.]

### 3.2 Database Schema

The database will be designed with the key tables and their relationships.



## 4. User Interface Design

The user interface will be intuitive and user-friendly, catering to the specific needs of each user role.

### 4.1 Wireframes and Mockups

[A collection of wireframes showing the user interface for patient registration, appointment booking, and the doctor's dashboard.]

Step 1 of 5

Welcome  
New Patient  
Registration  
Start

Step 2 3

Personal Details  
First Name  
Last Name  
Date of Birth  
Email  
Email  
Next

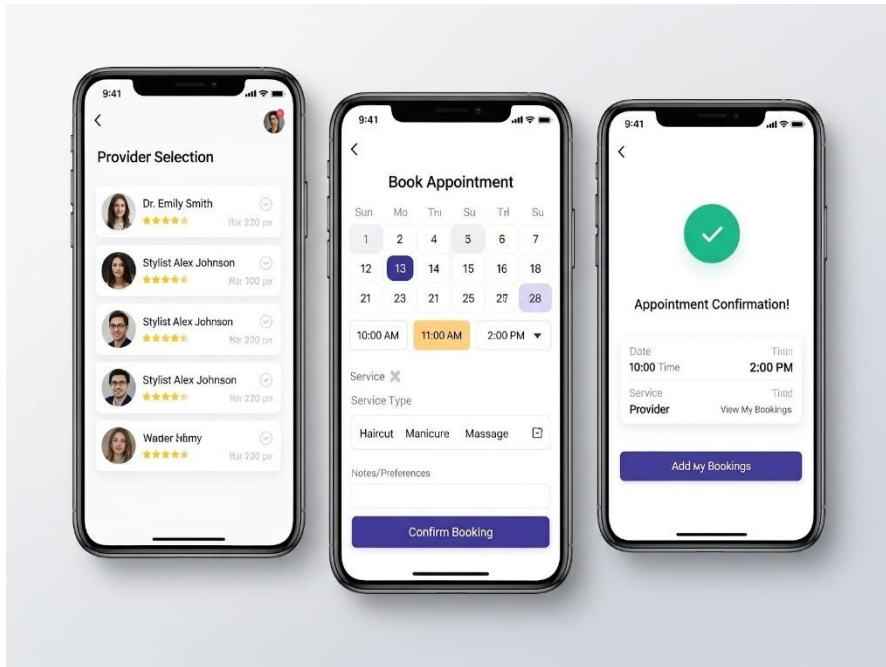
Step 15

Medical History  
☐ Allergies ☐ Allergies  
☒ Feet ☐ High Blood Pressure  
Medications  
Medications  
Next

Your Patient Information  
Your all entred information  
Next

Contact Information  
Phone Number  
Address  
Address  
Next

Profr Summary  
Your doctor (doctor id) has approved  
your all entred information  
Doctor  
Start Session  
Confirm Registration



## 4.2 User Roles

- **Administrator:** Full access to system configurations and reports.
- **Doctor:** Access to patient records, appointments, and treatment plans.
- **Nurse:** Patient vital signs, medication administration.
- **Receptionist:** Patient registration, appointment scheduling, billing.
- **Pharmacist:** Inventory management, medication dispensing.

## 5. Technical Specifications

### 5.1 Technology Stack

- **Frontend:** React.js / Angular

- **Backend:** Node.js / Java Spring Boot
- **Database:** PostgreSQL / MySQL
- **Cloud Platform:** AWS / Azure

## 5.2 Integrations

The HMS will support integration with external systems via APIs, including:

- **Laboratory Information Systems (LIS):** For lab test orders and results.
- **Radiology Information Systems (RIS):** For imaging orders and reports.
- **Payment Gateways:** For secure online payment processing.

## 6. Security Considerations

- **Authentication and Authorization:** Role-based access control (RBAC).
- **Data Encryption:** Encryption of sensitive patient data at rest and in transit.
- **Audit Trails:** Logging of all user activities for compliance and security monitoring

## 7. Future Enhancements

- **Telemedicine Integration:** Virtual consultations and remote patient monitoring.
- **Mobile Application:** Native mobile apps for patients and doctors.
- **AI-powered Diagnostics:** Integration with AI for predictive analytics and diagnostic support.

## 8. Conclusion

This document provides a foundational design for the Hospital Management System, outlining its architecture, modules, and key technical considerations. This design will serve as a robust framework for the development and successful deployment of a comprehensive and efficient HMS.

## 9. Appendix

### 9.1 References

- HL7 Standards: File

### 9.2 Contact Information

For further inquiries, please contact Person at Place.