Project Proposal

# On Project Title

**Guided By:- Created By:-**

**Trainer Name: Anuj Kumar Student Name: Anuj Kumar**

**AFid : AF04991734**

**Student Name: Himanshu**

**AFid : AF049717773**

**Student Name: Rahul Prajapati**

**AFid :AF04991823**

**Batch Code: ANP-D2405**

**Course Code: ITPR**

**Table Of Contents**

**Index**

1. Title of the Project…………………………….
2. Introduction……………………………………
3. Objective
4. Project Category
5. Analysis
   * Modules and Description
   * Database Design
   * ER Diagram
   * Data Flow Diagram
6. Complete Structure
   * Process Logical Diagram
7. Platform Used
   * Hardware Requirement
   * Software Requirement
8. Future Scope
9. Bibliography

**Tittle of the Project**

Hospital Management System

# Introduction

The Hospital Management System (HMS) is a software solution designed to digitalize and streamline the daily operations of a hospital. Instead of maintaining separate registers for patients, doctors, appointments, lab tests, and billing, the system stores all data in a centralized database.

The system mainly focuses on:

* Managing patient information and visit details.
* Managing doctors and departments with their specialization.
* Scheduling and tracking appointments between patients and doctors.
* Recording laboratory tests and results.
* Generating billing information for services availed by patients.

# OBJECTIVE

# To automate basic operations of a hospital using a computer-based system.

# To maintain accurate and up-to-date records of patients, doctors, and departments.

# To provide a proper system for booking and managing appointments.

# To store and retrieve lab test details and results efficiently.

# To generate bills for patients based on appointments and lab tests.

# To reduce manual paperwork and chances of errors.

# To allow quick searching and reporting from stored data.

# 

# PROJECT CATEGORY

* Database Management/Application Development
* Frontend: Core Java (Console/Command Line)
* Backend: MySQL Database
* Connectivity: JDBC (Java Database Connectivity).

# ANALYSIS

**a) Modules and Description**

**1. Patient Management Module**

* Add new patient records (name, age, gender, phone, address, etc.).
* Update patient details if any information changes.
* Search and view patient details by Patient ID, name, or phone number.
* Maintain history of visits and tests linked to the patient.

**2. Doctor & Department Management Module**

* Add doctor details such as name, specialization, phone, and department.
* Maintain a master list of hospital departments  
  (e.g., Cardiology, Orthopedic, Neurology).
* Link each doctor to a specific department.
* View the list of doctors under each department.

**3. Appointment Management Module**

* Book appointments between patient and doctor.
* Record appointment date, time, and purpose.
* Check doctor availability before booking.
* Update or cancel appointments when needed.
* View all appointments for a doctor or patient.

**4. Test Management Module**

* Register lab tests for patients (e.g., Blood test, X-ray, MRI).
* Store test name, type, date, and results.
* Link each lab test to the corresponding patient  
  (and optionally to an appointment/doctor).
* Provide quick access to the patient's test history during future visits.

**5. Billing Module**

* Generate bills for patients based on consultation fees, lab test charges, and services.
* Store bill amount, date, and payment status (Paid/Unpaid).
* View bills patient-wise or date-wise.
* Help hospital administration maintain financial records.

**6. Reports Module (Optional but Useful)**

* Patient-wise report (all appointments, tests, and bills).
* Daily appointments report.
* Daily/monthly billing report.

**b) Database Design**

**1. DEPARTMENT Table Design**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Datatype** | **Properties** |
| department\_id | int | primary key, auto increment |
| department\_name | varchar(100) | not null |

**2. PATIENT Table Design**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Datatype** | **Properties** |
| patient\_id | int | primary key, auto increment |
| name | varchar(100) | not null |
| gender | varchar(10) | not null |
| age | int | not null |
| phone | varchar(15) | not null |
| address | varchar(200) | not null |

**3. DOCTOR Table Design**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Datatype** | **Properties** |
| doctor\_id | int | primary key, auto increment |
| name | varchar(100) | not null |
| specialization | varchar(50) | not null |
| phone | varchar(15) | not null |
| department\_id | int | foreign key → department(department\_id) |

**Relation:** One patient can have multiple appointment with multiple

Doctors.

**4. APPOINTMENT Table Design**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Datatype** | **Properties** |
| appointment\_id | int | primary key, auto increment |
| patient\_id | int | foreign key → patient(patient\_id) |
| doctor\_id | int | foreign key → doctor(doctor\_id) |
| appointment\_date | date | not null |
| appointment\_time | time | not null |
| remarks | varchar(200) | nullable |

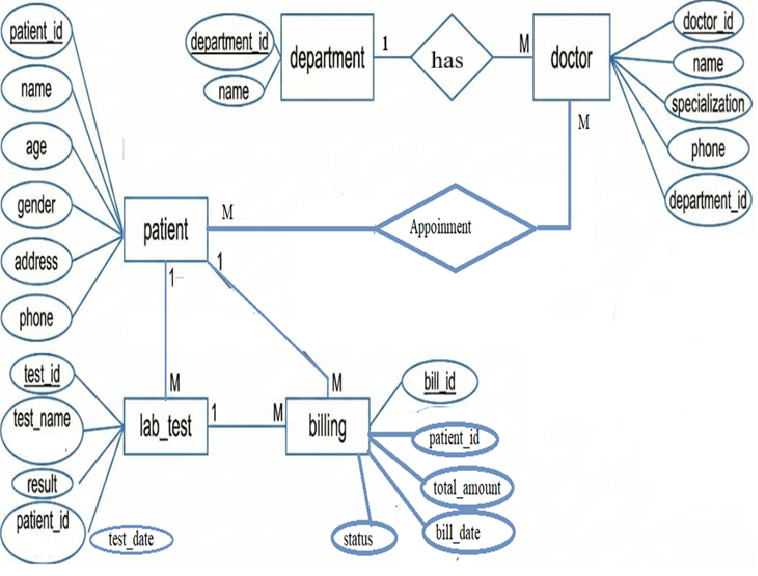
**5. LAB\_TEST Table Design**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Datatype** | **Properties** |
| test\_id | int | primary key, auto increment |
| patient\_id | int | foreign key → patient(patient\_id) |
| test\_name | varchar(100) | not null |
| test\_date | date | not null |
| result | varchar(200) | nullable |

**6. BILLING Table Design**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Datatype** | **Properties** |
| bill\_id | int | primary key, auto increment |
| patient\_id | int | foreign key → patient(patient\_id) |
| bill\_date | date | not null |
| total\_amount | decimal(10,2) | not null |
| status | varchar(20) | default 'unpaid' |

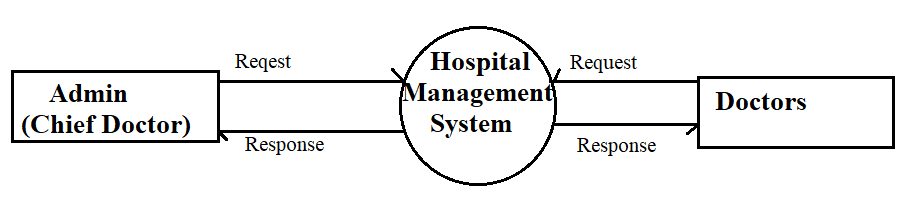
**Entity Relationship Diagram**



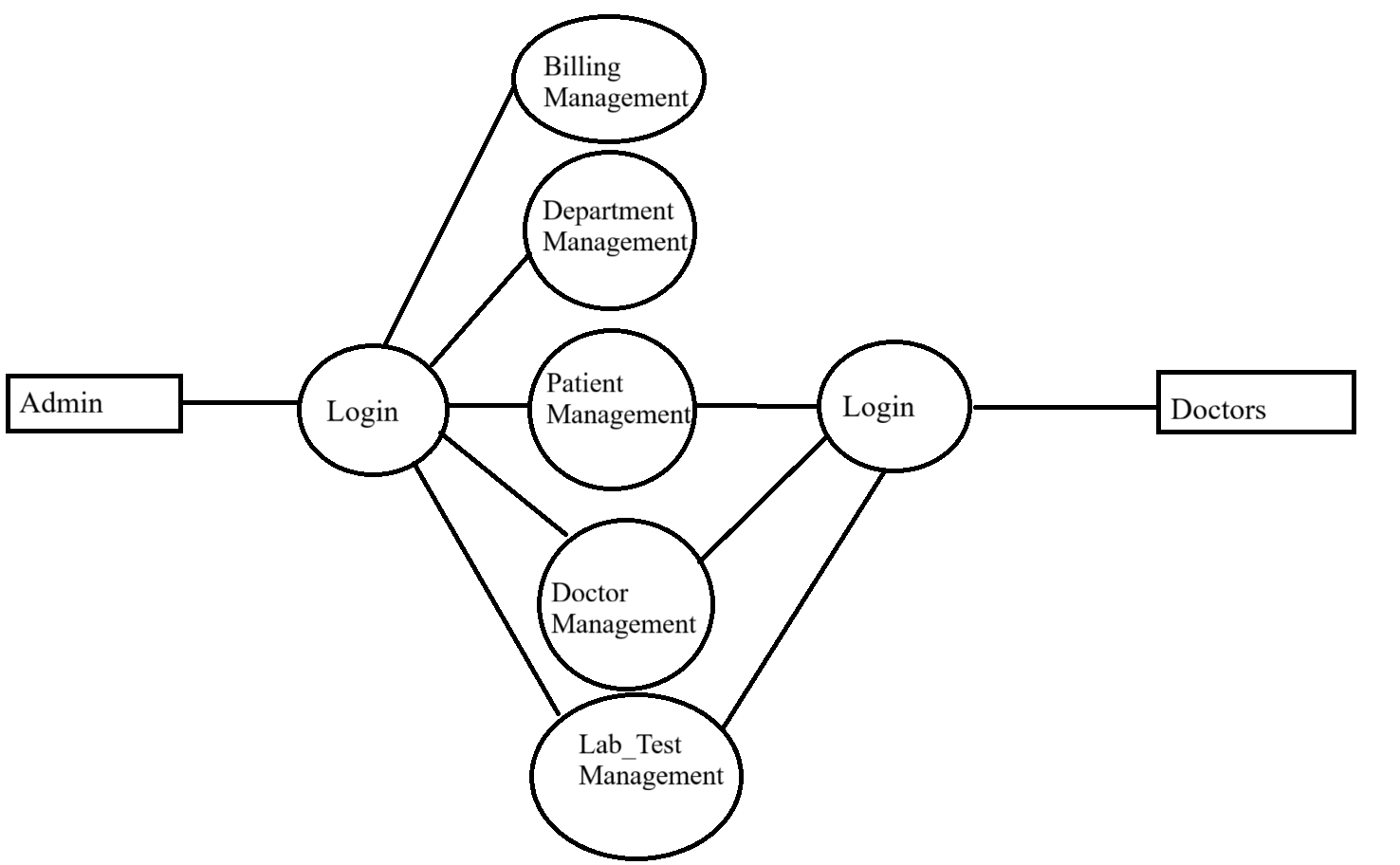


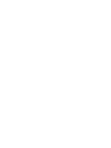
**Data Flow Diagram**

* 1. 0 Level DFD

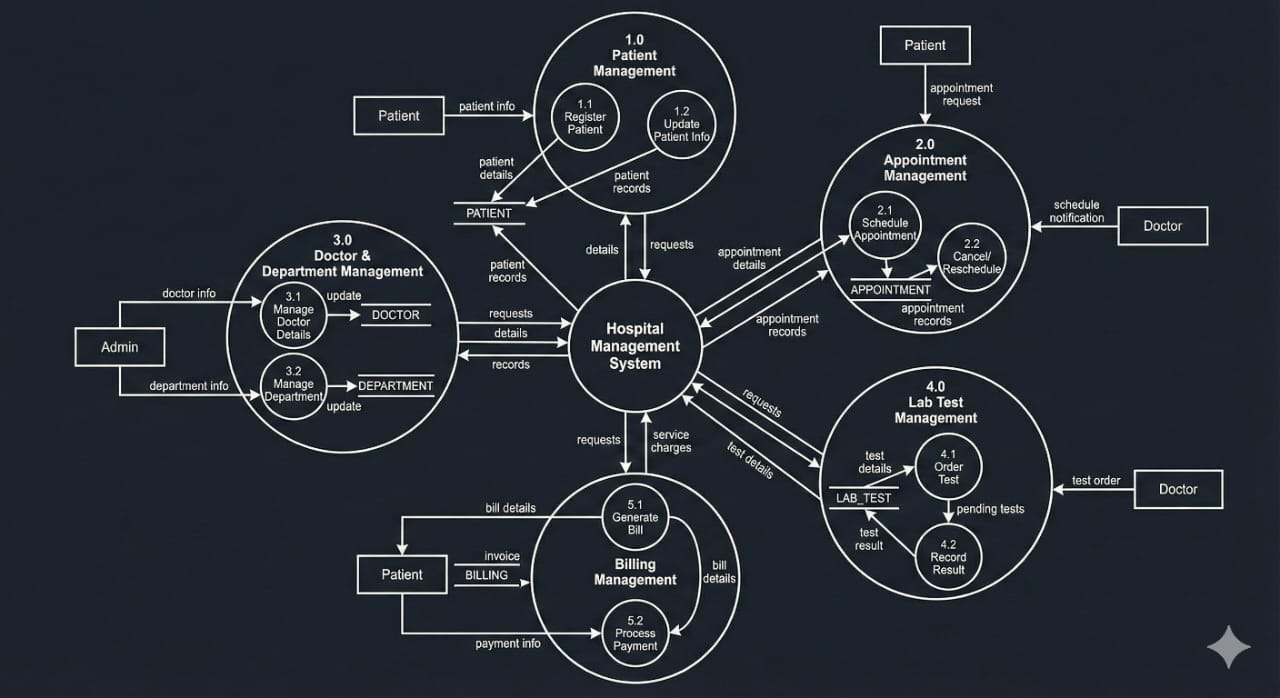
****

* 1. 1 level DFD



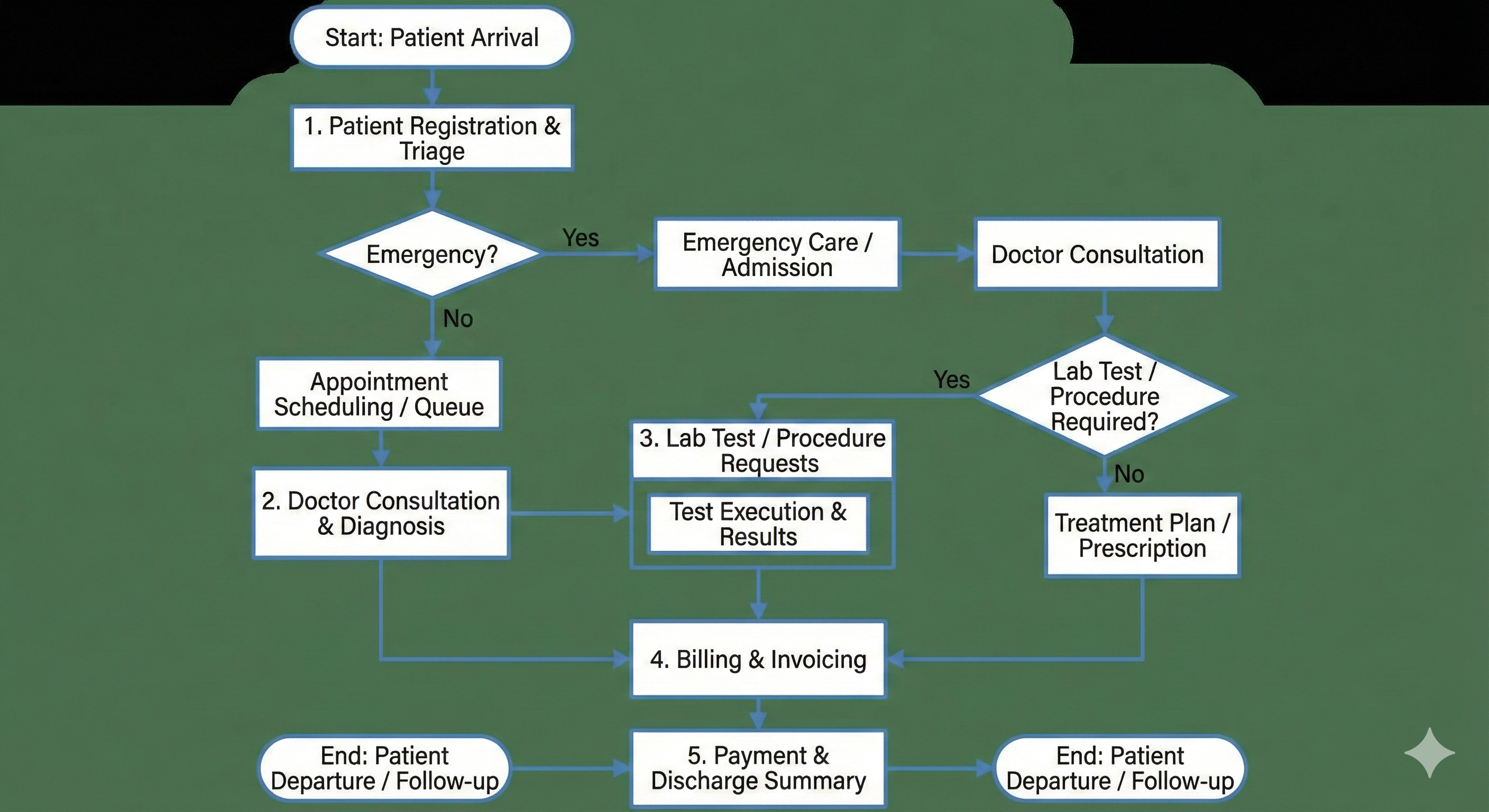


* 1. 2 level DFD



**Complete Structure**

**Process Logic Diagram**



## **Platform Used**

* **Hardware Requirements:** Any standard PC or laptop is sufficient. Since this is a simple command-line program, minimal specs (e.g. 1–2 GHz CPU, ~1–2 GB RAM) are adequate. It does not require specialized hardware; even a low-end machine can run the Java CLI app and MySQL server for demo purposes.
* **Software Requirements:**
  + **Operating System:** Windows, Linux, or macOS (any OS that can run Java and MySQL).
  + **Java Development Kit (JDK):** Java SE (version 8 or later recommended). The application is written in Java, so a JDK must be installed and configured in the system PATH[github.com](https://github.com/itsantu/hospital-management#:~:text=Prerequisites).
  + **MySQL Server:** MySQL Community Server (or any MySQL-compatible database) to host the hospital database. Ensure MySQL is running and accessible[github.com](https://github.com/itsantu/hospital-management#:~:text=Prerequisites).
  + **JDBC Connector:** MySQL Connector/J library (JAR) added to the project so Java can connect to MySQL[github.com](https://github.com/itsantu/hospital-management#:~:text=Prerequisites).
  + **IDE/Text Editor (optional):** Any Java editor (e.g. Eclipse, IntelliJ, VS Code) can be used for coding; but since the app is CLI, an IDE is not strictly necessary. The GitHub example used VS Code

## **Future Scope**

1. **Addition of Advanced Modules:**  
   The system can be expanded to include Billing, Pharmacy Management, Laboratory Reports, Ward/Bed Allocation, and Staff Payroll to transform it into a complete hospital information management solution.
2. **Upgrade to GUI or Web Application:**  
   The current command-line system can be redesigned with JavaFX/Swing for a graphical interface or converted into a web-based solution using Spring Boot, improving accessibility and usability.
3. **Integration of Online Portals:**  
   Future enhancements may include patient and doctor portals for booking appointments, checking reports, updating medical records, and managing schedules remotely.
4. **Automated Notifications:**  
   The system can be integrated with SMS/email gateways to send reminders for appointments, medicine refills, and follow-up visits, reducing patient no-shows and improving hospital workflow.
5. **Cloud-Based Deployment:**  
   Deploying the system on cloud platforms can enable real-time access, automatic backups, multi-branch connectivity, and high scalability for larger healthcare networks.
6. **Enhanced Security:**  
   Implementation of encryption, role-based access control, secure logins, and audit trails can strengthen data confidentiality and meet healthcare compliance standards.
7. **Mobile App Development:**  
   A mobile application can be developed for doctors, staff, and patients to access the system from anywhere, improving responsiveness and convenience.

**Bibliography**

* Java Documentation, Oracle: Official reference for Java programming concepts, object-oriented structure, and JDBC connectivity.
* MySQL Documentation: Comprehensive guide for SQL queries, relational database design, and MySQL server configuration.
* GeeksforGeeks: Conceptual references used for understanding Hospital Management System modules, database design, ER diagrams, and Java examples.
* Tutorials Point: Learning resource for Java fundamentals, JDBC operations, and structured programming techniques.
* GitHub Community Projects.  
  Open-source Java–MySQL command-line project examples used for understanding modular structure and