DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRI SIVASUBRAMANIYA NADAR COLLEGE OF ENGINEERING

(An Autonomous Institution, Affiliated to Anna University)

UCS2404 - DATABASE MANAGEMENT SYSTEMS

Assignment 1 (CAT 1)

HOSPITAL MANAGEMENT SYSTEM

By

SAI RAHUL, CSE B

R.NO.3122 21 5001 090

<u>INTRODUCTION</u>

<u>ABSTRACT</u>

Hospital Management system includes registration of patients, storing their details into the system, and also booking their appointments with doctors.
The software has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically. User can search availability of a doctor and the details of a patient using the id. The Hospital Management System can be entered using a username and password. It is accessible either by an administrator or receptionist. Only they can add data into the database. The data can be retrieved easily. The interface is very user-friendly. The data is well protected for personal use and makes the data processing very fast.

The Software is having mainly two modules. One is at Administration Level and other one is of user. The Application maintains authentication in order to access the application. Administrator task includes managing doctors information and patient's information. To achieve this aim the database was designed in such a way that we have one for the patient and other for the doctors and this can be accessed by the Admin. The complaints which are given by user will be referred by authorities.
The Patient modules include checking appointments, prescription and also paying doctor's fee online.

PROBLEM STATEMENT

In this busy world humans don’t have the time to wait in infamously long hospital queues. The problem is, queuing at hospital is often managed manually by administrative staff through token and then wait for their turn to meet the doctor and in some cases, patients become aware of doctors absence in the last minute and feel disappointed with the wastage of their time and efforts. In some cases, doctors refuse to take unscheduled appointments due to their busy schedule.

Hospital Management System (HMS) will help us overcome all these problems because, through this software, patients can book their appointments at home, they can check whether the doctor they want to meet is available or not. Doctors can also confirm or decline appointments. This will help both patient and the doctor because if the doctor declines’ appointment then patient will know this in advance and patient will visit hospital only when the doctor confirms’ the appointment this will save time and money of the patient.

Patients can also pay the doctor’s consultant fee online to save their time.

HMS is essential for all healthcare establishments, be it hospitals, nursing homes, health clinics, rehabilitation centers, dispensaries, or clinics. The main goal is to computerize allthe details regarding the patient and the hospital. The installation of this healthcare software results in improvement in administrative functions and hence better patient care, which is the prime focus of any healthcare unit.

Benefits of implementing a hospital management system:

- *Appointment booking*
  - Helps patients cut the long queue and saves their time
  - Is equipped with features like automated email and text message reminders
- *Role-Based Access Control*
  - Allows employees to access only the necessary information to effectivelyperform their job duties
  - Increases data security and integrity
- *Overall cost reduction*
  - Cuts down paper costs as all the data are computerized
  - No separate costs for setting up physical servers
- *Data accuracy*
  - Removes human errors
  - Alerts when there’s a shortage of stock
-

- *Data security*
    - Helps to keep patients records private
    - Restricts access through role-based access control

- *Revenue management*
    - Makes daily auditing simple
    - Helps with statistics and other financial aspects

Data analysis

the tables needed are:

Patient table:

PatientID (Primary Key)
PatientName
Age
Gender
ContactNumber
Address

Doctor table:

DoctorID (Primary Key)
DoctorName
Specialization
ContactNumber
Email

Employee table:

EmployeeID (Primary Key)
EmployeeName
JobTitle
DepartmentID (Foreign Key)

Department table:

DepartmentID (Primary Key)
DepartmentName

Appointment table:

AppointmentID (Primary Key)
PatientID (Foreign Key)
DoctorID (Foreign Key)
Appointment Date

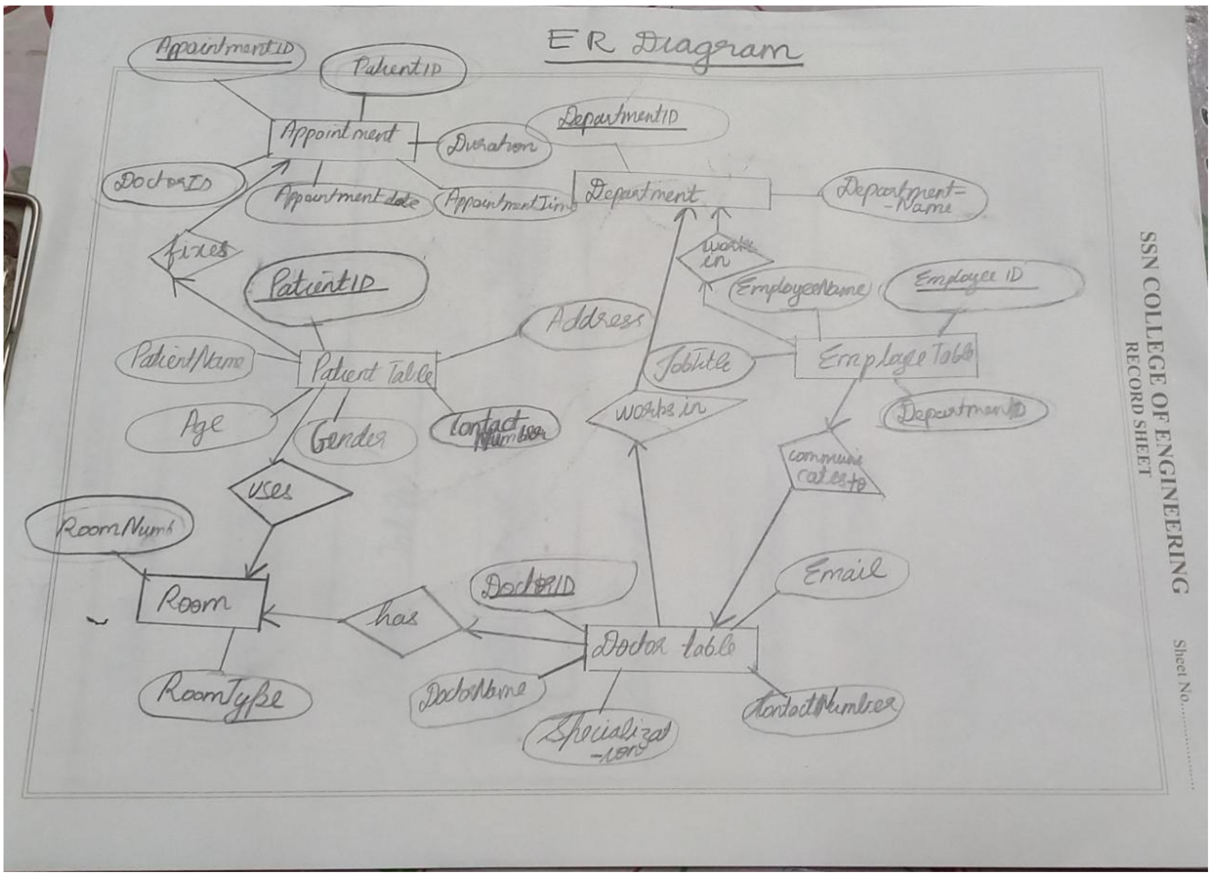
AppointmentTime
Duration

Room table:

RoomNumber (Primary Key)
RoomType

These tables represent the entities and their attributes in a hospital management system. They provide a structure to store and manage the data related to patients, doctors, employees, appointments, departments, and rooms.

ER Diagram:



In table format:
Patient table:

Column Name	Data Type	Constraints
PatientID	Primary Key	
PatientName	VARCHAR	
Age	INT	
Gender	VARCHAR	
ContactNumber	VARCHAR	
Address	VARCHAR	

Doctor table:

Column Name	Data Type	Constraints
DoctorID	Primary Key	
DoctorName	VARCHAR	
Specialization	VARCHAR	
ContactNumber	VARCHAR	
Email	VARCHAR	

Employee table:

Column Name	Data Type	Constraints
EmployeeID	Primary Key	
EmployeeName	VARCHAR	
JobTitle	VARCHAR	
DepartmentID	VARCHAR	Foreign Key RefDepartment(DepartmentID)

Department table:

Column Name	Data Type	Constraints
DepartmentID	Primary Key	
DepartmentName	VARCHAR	

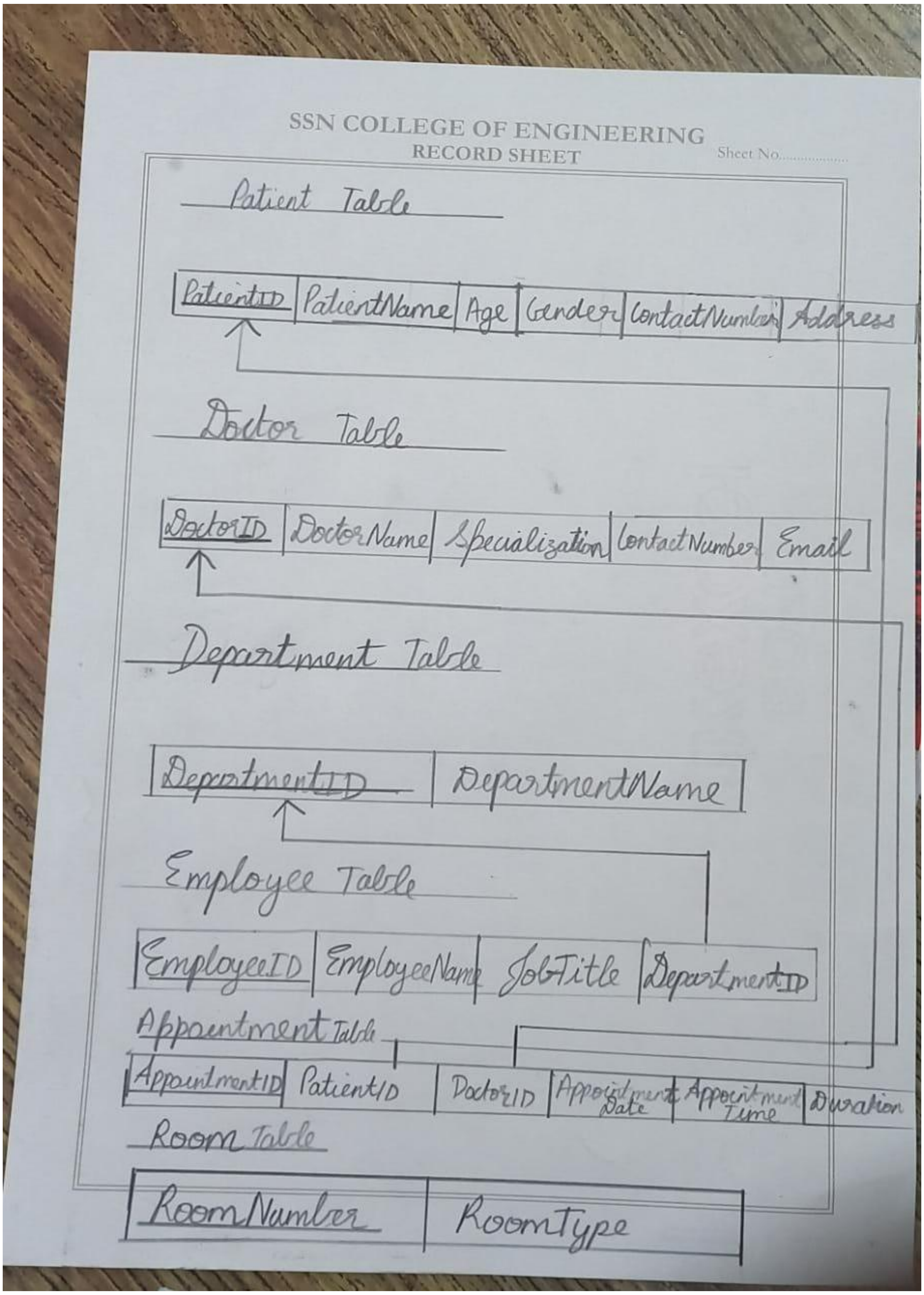
Appointment table:

Column Name	Data Type	Constraints
AppointmentID	Primary Key	
PatientID	Foreign Key	References Patient(PatientID)
DoctorID	Foreign Key	References Doctor(DoctorID)
AppointmentDate	DATE	
AppointmentTime	TIME	
Duration	INT	

Room table:

Column Name	Data Type	Constraints
RoomNumber	Primary Key	
RoomType	VARCHAR	

ER To relational model:



SSN COLLEGE OF ENGINEERING
RECORD SHEET
Sheet No..................

Patient Table

PatientID	PatientName	Age	Gender	ContactNumber	Address

Doctor Table

DoctorID	DoctorName	Specialization	ContactNumber	Email

Department Table

DepartmentID	DepartmentName

Employee Table

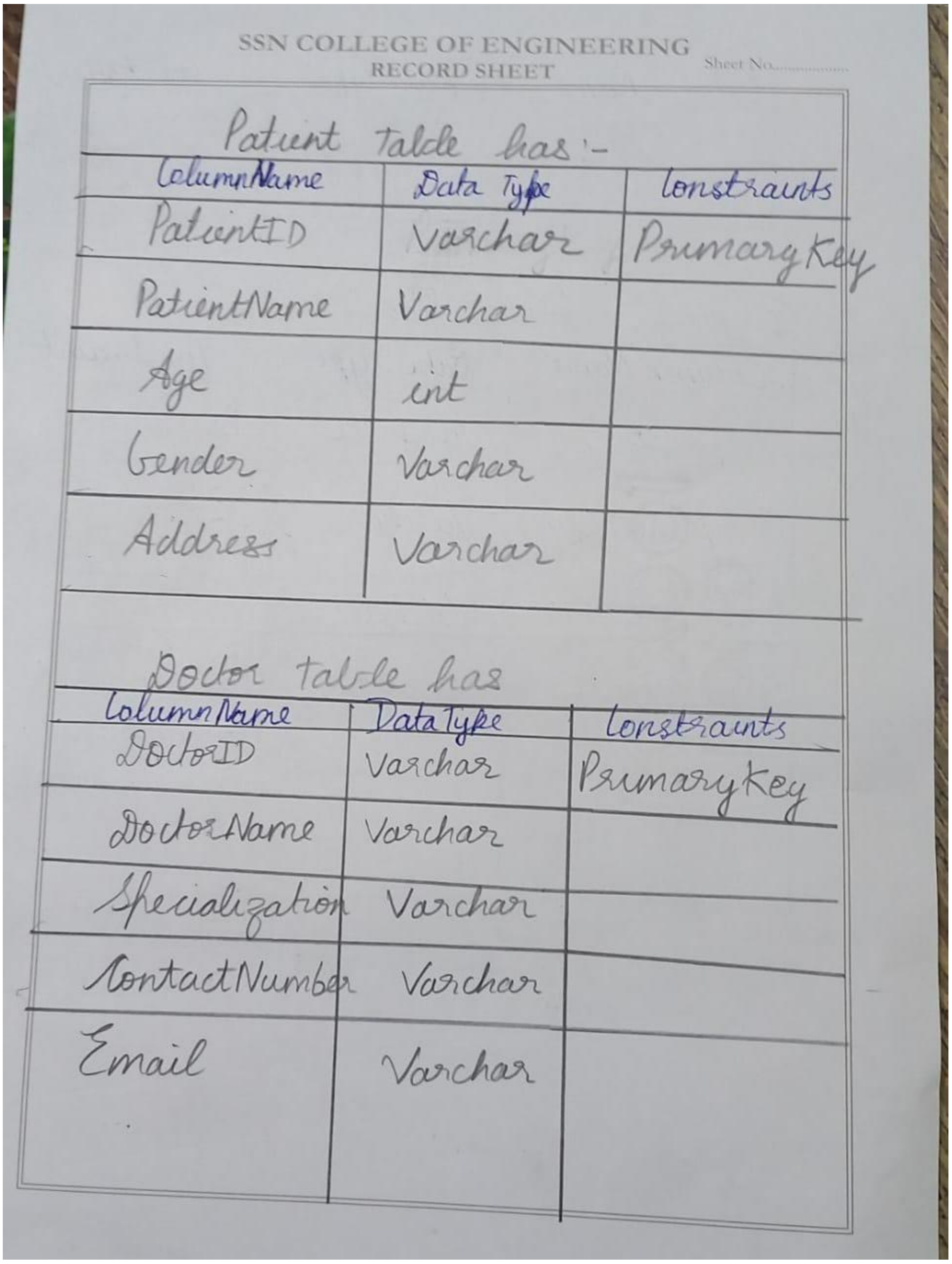
EmployeeID	EmployeeName	JobTitle	DepartmentID

Appointment Table

AppointmentID	PatientID	DoctorID	Appointment Date	Appointment Time	Duration

Room Table

RoomNumber	RoomType



SSN COLLEGE OF ENGINEERING
RECORD SHEET
Sheet No.

Patient table has:-

Column Name	Data Type	Constraints
PatientID	Varchar	Primary Key
PatientName	Varchar	
Age	int	
Gender	Varchar	
Address	Varchar	

Doctor table has

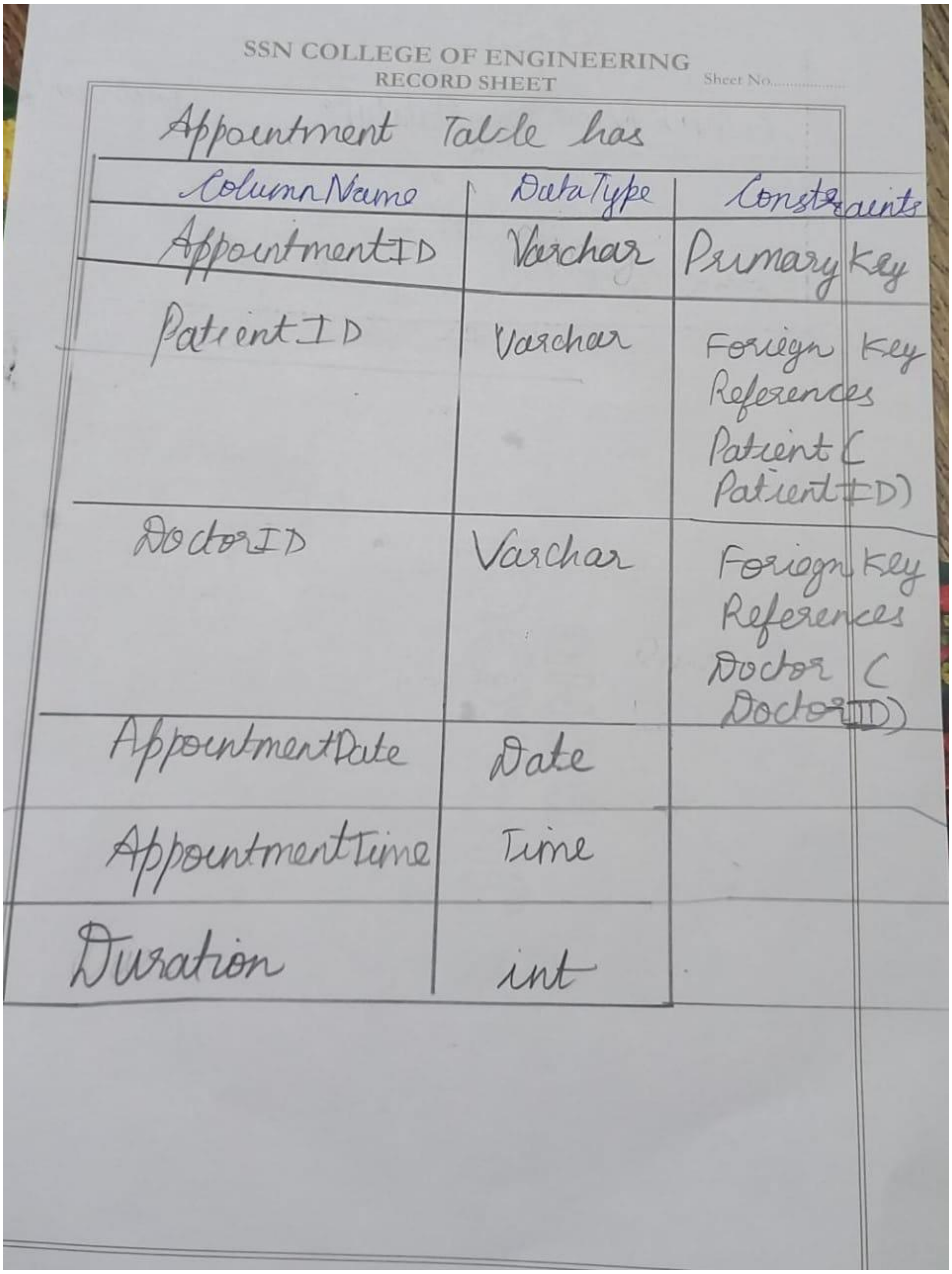
Column Name	Data Type	Constraints
DoctorID	Varchar	Primary Key
DoctorName	Varchar	
Specialization	Varchar	
ContactNumber	Varchar	
Email	Varchar	

Department Table has:-

Column Name	Data Type	Constraints
DepartmentID	Varchar	Primarykey
DepartmentName	Varchar	

Employee Table has:-

Column Name	Data Type	Constraints
EmployeeID	Varchar	Primarykey
EmployeeName	Varchar	
Job Title	Varchar	
DepartmentID	Varchar	Foreign Key References Department(DepartmentID)

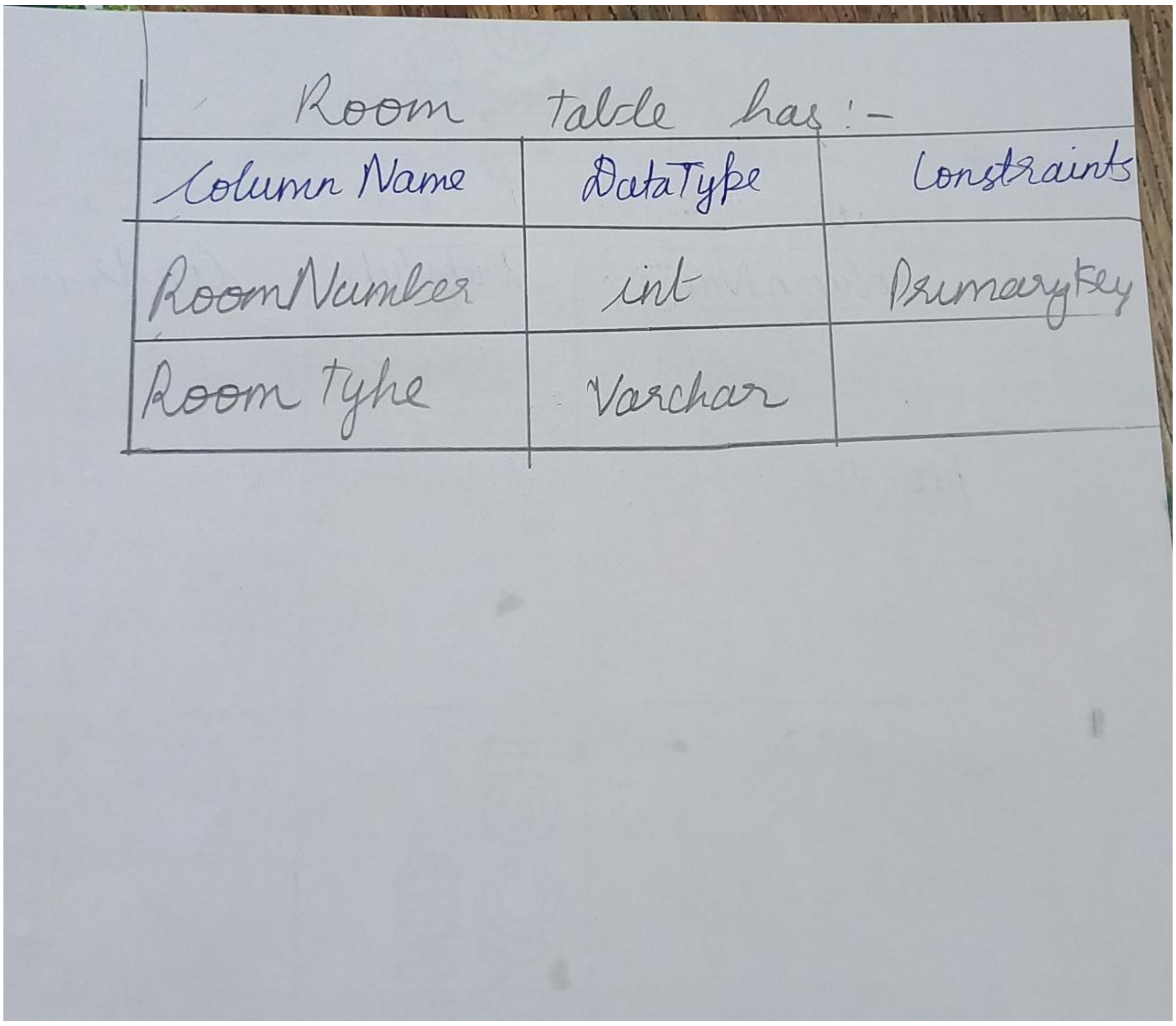


SSN COLLEGE OF ENGINEERING
RECORD SHEET

Sheet No.

Appointment Table has

Column Name	Data Type	Constraints
AppointmentID	Varchar	Primary Key
PatientID	Varchar	Foriegn Key References Patient (PatientID)
DoctorID	Varchar	Foriegn Key References Doctor (DoctorID)
AppointmentDate	Date	
AppointmentTime	Time	
Duration	int	



Room table has:-

Column Name	DataType	Constraints
RoomNumber	int	PrimaryKey
Room Type	Varchar	

(Phase-1: Identification of constraints and Functional dependencies (FD) among set of attributes)

For the Hospital Management System, in the first phase of analyzing the data, we need to identify the constraints and functional dependencies among the attributes. This process involves a deeper understanding of the problem specification, interactions/dependencies among various attributes, and user requirements for the specific real-time application.

Constraints:

Unique Constraints:

Patient ID: Each patient in the system should have a unique identifier to avoid duplication and ensure accurate identification.
Employee ID: Every employee working in the hospital should have a unique identification number.
Room Number: Each room within the hospital should have a unique number assigned to it.

Range Constraints:

Age: The age of patients and employees should fall within a valid and acceptable range.
Appointment Duration: The duration of an appointment should have a valid range, ensuring that appointments are neither too short nor too long.

Not Null Constraints:

Patient Name: The name of a patient should not be left empty.
Employee Name: The name of an employee should be provided and should not be null.
Appointment Time: The time of an appointment should be specified and should not be left blank.

Other constraints

•System is wirelessly networked with an encryption.
•System is only accessible within the hospital’s website only.
•Database is password protected.
•Should use less RAM and processing power.
•Each user should have individual ID and password.
•Only administrator can access the whole system.

Functional Dependencies:

Patient-Doctor Dependency:

Doctor ID: Each patient should be associated with a specific doctor responsible for their treatment.
Medical History: The medical history of a patient depends on the assigned doctor, as different doctors may have different approaches to treatment and may record different information.

Employee-Department Dependency:

Department ID: Employees working in different departments should be associated with the respective department they belong to.
Job Title: The job title or role of an employee may vary depending on the department they work in.

Appointment-Patient Dependency:

Patient ID: Each appointment should be linked to a specific patient.
Appointment Date: The date of an appointment depends on the patient and should be scheduled accordingly.
These examples provide a starting point for identifying constraints and functional dependencies within a hospital management system. It is crucial to thoroughly analyze the problem, interact with stakeholders, and consider specific user requirements to identify all relevant constraints and dependencies accurately.

Given below are the functional dependencies for the tables in the hospital management system:

Patient table:

PatientID → PatientName, Age, Gender, ContactNumber, Address
(The PatientID uniquely determines the attributes PatientName, Age, Gender, ContactNumber, and Address. Each patient has a unique ID associated with their information.)

Doctor table:

DoctorID → DoctorName, Specialization, ContactNumber, Email
(The DoctorID uniquely determines the attributes DoctorName, Specialization, ContactNumber, and Email. Each doctor has a unique ID associated with their information.)

Employee table:

EmployeeID → EmployeeName, JobTitle, DepartmentID

(The EmployeeID uniquely determines the attributes EmployeeName, JobTitle, and DepartmentID. Each employee has a unique ID associated with their information.)

Department table:

DepartmentID → DepartmentName
(The DepartmentID uniquely determines the attribute DepartmentName. Each department has a unique ID associated with its name.)

Appointment table:

AppointmentID → PatientID, DoctorID, AppointmentDate, AppointmentTime, Duration
(The AppointmentID uniquely determines the attributes PatientID, DoctorID, AppointmentDate, AppointmentTime, and Duration. Each appointment has a unique ID associated with the appointment details.)

Room table:

RoomNumber → RoomType
(The RoomNumber uniquely determines the attribute RoomType. Each room has a unique number associated with its type.)

These functional dependencies indicate how the attributes in each table are related and provide a basis for data integrity and normalization in the hospital management system.

Other Assumptions and dependencies

- Each user must have a valid user id and password
- Server must be running for the system to function
- Users must log in to the system to access any record.
- Only the Administrator can delete records.

Phase 2:

All the tables meet 1nf 2nf 3nf and bcnf even with constraints.
Reasons are as follows:

Definition of 1NF

1NF (First Normal Form) is the initial level of normalization in the relational database model. It sets the basic requirements for organizing data in a tabular format and eliminates redundant data. 1NF disallows realtions within relations or relations as attribute values within tuples.To satisfy 1NF, a relation must adhere to the following conditions:

Atomic Values: Each attribute (column) in a relation should contain atomic values. This means that each attribute cannot have multiple values or be further divided into smaller parts. For example, a "Name" attribute should not contain multiple names separated by commas; it should store only a single name.

Unique Rows: Each row in a relation should be unique. There should be no duplicate rows in the relation. Every row should have a unique combination of values across all attributes.

Orderless: The order of rows and columns should be irrelevant to the interpretation of the data. The relational model treats relations as unordered sets of tuples, meaning the database system should not rely on any specific order for processing or retrieving data.

By adhering to these conditions, a relation is considered to be in 1NF. It provides the foundation for higher levels of normalization, such as 2NF, 3NF, and beyond, which further eliminate data redundancy and ensure data integrity in a relational database.

Definition of 2NF

2NF (Second Normal Form) is the next level of normalization in the relational database model. It builds upon the requirements of 1NF and adds further rules to eliminate data redundancy and partial dependencies. To satisfy 2NF, a relation must fulfill the following conditions:

It must already be in 1NF (First Normal Form).

All non-key attributes must be fully functionally dependent on the entire primary key.

In summary, 2NF requires a relation to be in 1NF and have all non-key attributes fully functionally dependent on the entire primary key. If any partial dependencies exist, the relation should be split into separate relations to satisfy ***2NF.***

Definition of 3NF:

3NF (Third Normal Form) is a level of database normalization in which a relation must satisfy the following conditions:

It must already be in 2NF (Second Normal Form).

There should be no transitive dependencies.

A transitive dependency occurs when a non-key attribute depends on another non-key attribute rather than directly depending on the primary key. In 3NF, all non-key attributes should depend only on the primary key and not on other non-key attributes.

By eliminating transitive dependencies, 3NF ensures that data redundancy is minimized and data integrity is maintained in the relational database.

Definition of BCNF:

BCNF (Boyce-Codd Normal Form) is a high level of normalization in the relational database model. It is an extension of the third normal form (3NF) and aims to eliminate all non-trivial dependencies by considering all functional dependencies in a relation. To satisfy BCNF, a relation must meet the following conditions:

It must already be in 3NF (Third Normal Form).

Every determinant of a non-trivial functional dependency in the relation must be a candidate key.

In BCNF, a non-trivial functional dependency means that the dependent attribute(s) are not a subset of the determinant. A determinant is an attribute or set of attributes on which other attributes are functionally dependent.

By ensuring that every determinant of a non-trivial functional dependency is a candidate key, BCNF eliminates all potential data redundancy and ensures that each attribute in the relation is functionally determined by the candidate key(s) alone.

BCNF provides a higher level of normalization compared to 3NF, but it can result in more relations being split to achieve the desired level of normalization and maintain data integrity in the database.

Ans for phase 2
:Let us review the functional dependencies/Constraints of all the tables to check 1NF,2NF,3NF and BCNF

Constraints:

Unique Constraints:

Patient ID: Each patient in the system should have a unique identifier to avoid duplication and ensure accurate identification.
Employee ID: Every employee working in the hospital should have a unique identification number.
Room Number: Each room within the hospital should have a unique number assigned to it.

Range Constraints:

Age: The age of patients and employees should fall within a valid and acceptable range.
Appointment Duration: The duration of an appointment should have a valid range, ensuring that appointments are neither too short nor too long.

Not Null Constraints:

Patient Name: The name of a patient should not be left empty.
Employee Name: The name of an employee should be provided and should not be null.
Appointment Time: The time of an appointment should be specified and should not be left blank.

Functional Dependencies:

Patient-Doctor Dependency:

Doctor ID: Each patient should be associated with a specific doctor responsible for their treatment.
Medical History: The medical history of a patient depends on the assigned doctor, as different doctors may have different approaches to treatment and may record different information.

Employee-Department Dependency:

Department ID: Employees working in different departments should be associated with the respective department they belong to.

Job Title: The job title or role of an employee may vary depending on the department they work in.

<u>Appointment-Patient Dependency</u>:

Patient ID: Each appointment should be linked to a specific patient.
Appointment Date: The date of an appointment depends on the patient and should be scheduled accordingly.
These examples provide a starting point for identifying constraints and functional dependencies within a hospital management system. It is crucial to thoroughly analyze the problem, interact with stakeholders, and consider specific user requirements to identify all relevant constraints and dependencies accurately.

Given below are the functional dependencies for the tables in the hospital management system:

Patient table:

PatientID → PatientName, Age, Gender, ContactNumber, Address
(The PatientID uniquely determines the attributes PatientName, Age, Gender, ContactNumber, and Address. Each patient has a unique ID associated with their information.)

Doctor table:

DoctorID → DoctorName, Specialization, ContactNumber, Email
(The DoctorID uniquely determines the attributes DoctorName, Specialization, ContactNumber, and Email. Each doctor has a unique ID associated with their information.)

Employee table:

EmployeeID → EmployeeName, JobTitle, DepartmentID
(The EmployeeID uniquely determines the attributes EmployeeName, JobTitle, and DepartmentID. Each employee has a unique ID associated with their information.)

Department table:

DepartmentID → DepartmentName
(The DepartmentID uniquely determines the attribute DepartmentName. Each department has a unique ID associated with its name.)

Appointment table:

AppointmentID → PatientID, DoctorID, AppointmentDate, AppointmentTime, Duration
(The AppointmentID uniquely determines the attributes PatientID, DoctorID, AppointmentDate, AppointmentTime, and Duration. Each appointment has a unique ID associated with the appointment details.)

Room table:

RoomNumber → RoomType
(The RoomNumber uniquely determines the attribute RoomType. Each room has a unique number associated with its type.)

These functional dependencies indicate how the attributes in each table are related and provide a basis for data integrity and normalization in the hospital management system.

<u>Other Assumptions and dependencies</u>

- Each user must have a valid user id and password
- Server must be running for the system to function
- Users must log in to the system to access any record.
- Only the Administrator can delete records.

Analyzing whether the tables meet 1NF, 2NF, and 3NF, taking into account the constraints:

Patient table:

Primary Key: PatientID

Constraints: NOT NULL (PatientName, Age, Gender), UNIQUE (ContactNumber)

Functional Dependencies:

PatientID → PatientName, Age, Gender, ContactNumber, Address
1NF: The table has a primary key, and there are no repeating groups or nested data. Therefore, it satisfies 1NF.

2NF: There are no partial dependencies since all non-key attributes depend on the entire primary key. Therefore, it satisfies 2NF.

3NF: There are no transitive dependencies. All non-key attributes depend only on the primary key. Therefore, it satisfies 3NF.

Doctor table:

Primary Key: DoctorID

Constraints: NOT NULL (DoctorName, Specialization), UNIQUE (ContactNumber, Email)

Functional Dependencies:

DoctorID → DoctorName, Specialization, ContactNumber, Email
1NF: The table has a primary key, and there are no repeating groups or nested data. Therefore, it satisfies 1NF.

2NF: There are no partial dependencies since all non-key attributes depend on the entire primary key. Therefore, it satisfies 2NF.

3NF: There are no transitive dependencies. All non-key attributes depend only on the primary key. Therefore, it satisfies 3NF.

Employee table:

Primary Key: EmployeeID

Constraints: NOT NULL (EmployeeName, JobTitle)

Functional Dependencies:

EmployeeID → EmployeeName, JobTitle
1NF: The table has a primary key, and there are no repeating groups or nested data. Therefore, it satisfies 1NF.

2NF: There are no partial dependencies since all non-key attributes depend on the entire primary key. Therefore, it satisfies 2NF.

3NF: There are no transitive dependencies. All non-key attributes depend only on the primary key. Therefore, it satisfies 3NF.

Department table:

Primary Key: DepartmentID

Constraints: NOT NULL (DepartmentName)

Functional Dependencies:

DepartmentID → DepartmentName
1NF: The table has a primary key, and there are no repeating groups or nested data. Therefore, it satisfies 1NF.

2NF: There is only one attribute in the table, so partial dependencies are not applicable. Therefore, it satisfies 2NF.

3NF: There are no transitive dependencies. All non-key attributes depend only on the primary key. Therefore, it satisfies 3NF.

Appointment table:

Primary Key: AppointmentID

Constraints: NOT NULL (AppointmentDate, AppointmentTime, Duration)

Functional Dependencies:

AppointmentID → AppointmentDate, AppointmentTime, Duration
1NF: The table has a primary key, and there are no repeating groups or nested data. Therefore, it satisfies 1NF.

2NF: There are no partial dependencies since all non-key attributes depend on the entire primary key. Therefore, it satisfies 2NF.

3NF: There are no transitive dependencies. All non-key attributes depend only on the primary key. Therefore, it satisfies 3NF.

Room table:

Primary Key: RoomNumber

Constraints: NOT NULL (RoomType)

1NF:

The Room table has a primary key (RoomNumber) that uniquely identifies each room.
The RoomType attribute does not contain repeating groups or nested data.
Therefore, the Room table satisfies 1NF.

2NF:

Since the Room table has only two attributes (RoomNumber and RoomType), partial dependencies are not applicable.
Therefore, the Room table automatically satisfies 2NF.

3NF:

The Room table has no transitive dependencies. All attributes directly depend on the primary key (RoomNumber).
Therefore, the Room table satisfies 3NF.

In conclusion, based on the given constraints, the Room table satisfies 1NF, 2NF, and 3NF.

To check BCNF

To determine if the tables satisfy BCNF while considering the constraints, we need to ensure that for each functional dependency X → Y in a table, X is a superkey. Let's reassess the tables with constraints in mind:

Patient table:

Primary Key: PatientID
Constraints: NOT NULL (PatientName, Age, Gender), UNIQUE (ContactNumber)
Functional Dependencies:
PatientID → PatientName, Age, Gender, ContactNumber, Address
The primary key PatientID is a superkey, and all functional dependencies have the superkey on the left-hand side. Therefore, the Patient table satisfies BCNF.
Doctor table:

Primary Key: DoctorID
Constraints: NOT NULL (DoctorName, Specialization), UNIQUE (ContactNumber, Email)
Functional Dependencies:
DoctorID → DoctorName, Specialization, ContactNumber, Email
The primary key DoctorID is a superkey, and all functional dependencies have the superkey on the left-hand side. Therefore, the Doctor table satisfies BCNF.
Employee table:

Primary Key: EmployeeID
Constraints: NOT NULL (EmployeeName, JobTitle)
Functional Dependencies:
EmployeeID → EmployeeName, JobTitle
The primary key EmployeeID is a superkey, and all functional

dependencies have the superkey on the left-hand side. Therefore, the Employee table satisfies BCNF.

Department table:

Primary Key: DepartmentID

Constraints: NOT NULL (DepartmentName)

Functional Dependencies:

DepartmentID → DepartmentName

The primary key DepartmentID is a superkey, and all functional dependencies have the superkey on the left-hand side. Therefore, the Department table satisfies BCNF.

Appointment table:

Primary Key: AppointmentID

Constraints: NOT NULL (AppointmentDate, AppointmentTime, Duration)

Functional Dependencies:

AppointmentID → AppointmentDate, AppointmentTime, Duration

The primary key AppointmentID is a superkey, and all functional dependencies have the superkey on the left-hand side. Therefore, the Appointment table satisfies BCNF.

Room table:

Primary Key: RoomNumber

Constraints: NOT NULL (RoomType)

Functional Dependencies:

RoomNumber → RoomType

The primary key RoomNumber is a superkey, and all functional dependencies have the superkey on the left-hand side. Therefore, the Room table satisfies BCNF.

Considering the constraints, all tables still satisfy BCNF because each functional dependency has a superkey on the left-hand side. Hence, the tables meet the requirements of BCNF even with the constraints taken into account.

<u>Normalizing the tables and decomposing them into relations that satisfy up to 3NF or BCNF.</u>

Patient table:

Primary Key: PatientID

Doctor table:

Primary Key: DoctorID

Employee table:

Primary Key: EmployeeID
Foreign Key: DepartmentID

Department table:

Primary Key: DepartmentID

Appointment table:

Primary Key: AppointmentID
Foreign Keys: PatientID, DoctorID

Room table:

Primary Key: RoomNumber
To check if these tables satisfy normalization up to 3NF, we need to examine the functional dependencies and eliminate any transitive dependencies.

Patient table:
No transitive dependencies exist. It satisfies 3NF.
Doctor table:

No transitive dependencies exist. It satisfies 3NF.
Employee table:

No transitive dependencies exist. It satisfies 3NF.
Department table:

No transitive dependencies exist. It satisfies 3NF.
Appointment table:

No transitive dependencies exist. It satisfies 3NF.
Room table:

No transitive dependencies exist. It satisfies 3NF.
Based on the analysis, all the tables satisfy normalization up to 3NF.

DATABASE DESIGN DOCUMENTATION FOR HOSPITAL MANAGEMENT SYSTEM

ABSTRACT

Hospital Management system includes registration of patients, storing their details into the system, and also booking their appointments with doctors.
The software has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically. User can search availability of a doctor and the details of a patient using the id. The Hospital Management System can be entered using a username and password. It is accessible either by an administrator or receptionist. Only they can add data into the database. The data can be retrieved easily. The interface is very user-friendly. The data is well protected for personal use and makes the data processing very fast.

The Software is having mainly two modules. One is at Administration Level and other one is of user. The Application maintains authentication in order to access the application. Administrator task includes managing doctors information and patient’s information. To achieve this aim the database was designed in such a way that we have one for the patient and other for the doctors and this can be accessed by the Admin. The complaints which are given by user will be referred by authorities.
The Patient modules include checking appointments, prescription and also paying doctor’s fee online.

PROBLEM STATEMENT

In this busy world humans don’t have the time to wait in infamously long hospital queues. The problem is, queuing at hospital is often managed manually by administrative staff through token and then wait for their turn to meet the doctor and in some cases, patients become aware of doctors absence in the last minute and feel disappointed with the wastage of their timeand efforts. In some cases, doctors refuse to take unscheduled appointments due to their busy schedule.

Hospital Management System (HMS) will help us overcome all these problemsbecause, through this software, patients can book their appointments at home,they can check whether the doctor they want to meet is available or not.
Doctors can also confirm or decline appointments. This will help both patientand the doctor because if the doctor declines’ appointment then patient will know this in advance and patient will visit hospital only when the doctor
confirms’ the appointment this will save time and money of the patient. Patients can also pay the

doctor’s consultant fee online to save their time.

HMS is essential for all healthcare establishments, be it hospitals, nursing homes, health clinics, rehabilitation centers, dispensaries, or clinics. The main goal is to computerize allthe details regarding the patient and the hospital. The installation of this healthcare software results in improvement in administrative functions and hence better patient care, which is the prime focus of any healthcare unit.

2.1.1 System Interfaces

❖ User Interfaces

- This section provides a detailed description of all inputs into and outputs from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.
- The **protocol used** shall be **HTTP**.

❖

❖ Hardware Interfaces

- **Laptop/Desktop PC**-Purpose of this is to give information when Patients ask information about doctors, medicine available lab tests etc. To perform such Action it need very efficient computer otherwise due to that reason patients have to wait for a long time to get what they ask for.
- **Laser Printer (B/W)** - This device is for printing patients' info etc.
- **Wi-Fi router** - Wi-Fi router is used to for internetwork operations inside of a hospital and simply data transmission from pc's to sever.

❖ Software Interfaces

- **Mysql server** - Database connectivity and management
- **OS Windows 7/8/8.1**- Very user friendly and common OS

2.1.2 <u>System Specifications</u>

2.1.2.1 H/W Requirement

- ☞ Core i5 processor
- ☞ 2GB Ram.
- ☞ 20GB of hard disk space in terminal machines
- ☞ 1TB hard disk space in Server Machine

2.1.2.2 S/W Requirement

- ☞ Windows 7 or above operating system
- ☞ Mysql server

<u>Benefits of implementing a hospital management system:</u>

- *Appointment booking*
  - Helps patients cut the long queue and saves their time
  - Is equipped with features like automated email and text message reminders
- *Role-Based Access Control*
  - Allows employees to access only the necessary information to effectivelyperform their job duties

  - Increases data security and integrity
- *Overall cost reduction*
  - Cuts down paper costs as all the data are computerized
  - No separate costs for setting up physical servers

- *Data accuracy*
  - Removes human errors
  - Alerts when there’s a shortage of stock

-
- *Data security*
  - Helps to keep patients records private
  - Restricts access through role-based access control

- *Revenue management*
  - Makes daily auditing simple
  - Helps with statistics and other financial aspects

Requirements:
Following are the requirements for the system:

- *Store patient information, including name, age, gender, contact number, and address.*
- *Manage doctor records, including name, specialization, contact number, and email.*
- *Maintain employee details, such as name, job title, and department.*
- *Organize departments within the hospital, identified by a unique department ID and name.*
- *Schedule appointments between patients and doctors, tracking appointment details like date, time, and duration.*
- *Manage room information, including room number and room type.*

Constraints:
The following constraints are applied to ensure data integrity:
Patient table constraints:

Primary Key constraint on PatientID to ensure each patient has a unique identifier.
NOT NULL constraint on PatientName, Age, and Gender to ensure these attributes are always provided.
CHECK constraint on Age to ensure it is a positive value.
Unique constraint on ContactNumber to ensure each contact number is unique for each patient.
Doctor table constraints:

Primary Key constraint on DoctorID to ensure each doctor has a unique identifier.

NOT NULL constraint on DoctorName and Specialization to ensure these attributes are always provided.

Unique constraint on ContactNumber and Email to ensure each doctor has a unique contact number and email.

Employee table constraints:

Primary Key constraint on EmployeeID to ensure each employee has a unique identifier.

NOT NULL constraint on EmployeeName and JobTitle to ensure these attributes are always provided.

Foreign Key constraint on DepartmentID to link the employee to a department.

Department table constraints:

Primary Key constraint on DepartmentID to ensure each department has a unique identifier.

NOT NULL constraint on DepartmentName to ensure it is always provided.

Appointment table constraints:

Primary Key constraint on AppointmentID to ensure each appointment has a unique identifier.

Foreign Key constraints on PatientID and DoctorID to link the appointment to a specific patient and doctor.

NOT NULL constraints on AppointmentDate, AppointmentTime, and Duration to ensure these attributes are always provided.

CHECK constraint on Duration to ensure it is a positive value.

Room table constraints:

Primary Key constraint on RoomNumber to ensure each room has a unique identifier.

NOT NULL constraint on RoomType to ensure it is always provided.

Design Choices:

Based on the problem specification and requirements, the following design choices were made:

Each table has a primary key attribute to ensure uniqueness and efficient record retrieval.

Foreign keys are used to establish relationships between tables, such as linking employees to departments and appointments to patients and doctors.

Appropriate data types are chosen for each attribute to accurately represent the information.

Constraints are implemented to enforce data integrity and business rules.

The normalized tables and their relationships satisfy up to 3NF, ensuring minimal redundancy and data anomalies. The design choices align with the problem specification and requirements, providing an effective and robust database schema for the hospital management system.

Another version of the database schema design, indicating the foreign key references are as follows:

Patient table:

Patient (PatientID [PRIMARY KEY], PatientName, Age, Gender, ContactNumber, Address)

Doctor table:

Doctor (DoctorID [PRIMARY KEY], DoctorName, Specialization, ContactNumber, Email)

Employee table:

Employee (EmployeeID [PK], EmployeeName, JobTitle, DepartmentID [FOREIGN KEY])

Department table:

Department (DepartmentID [PK], DepartmentName)

Appointment table:

Appointment (AppointmentID [PK], PatientID [FK referencing Patient], DoctorID [FOREIGN KEY referencing Doctor], AppointmentDate, AppointmentTime, Duration)

Room table:

Room (RoomNumber [PK], RoomType)

In the revised schema, the foreign keys have been explicitly marked with the tables they reference.

The relationships between the tables are as follows:

The Appointment table has a foreign key PatientID, which references the Patient table.

The Appointment table also has a foreign key DoctorID, which references the Doctor table.

The Employee table has a foreign key DepartmentID, which references the Department table.

This database schema design ensures that the appropriate relationships are established between the tables through the use of foreign keys. It satisfies the requirements of normalization up to the 3NF level, eliminating redundancy and maintaining data integrity.