

# Database Management System Course Project

## Hospital Management System

### Abstract :

The purpose of the project is to computerize the Front Office Management of Hospital to develop software which is user friendly, simple, fast and cost effective. It deals with the collection of patient's information like adding, updating, deleting, searching patients, viewing patient diagnosis, updating bills etc. The main function of the system is to register and store patient details and doctor details and retrieve details as and when required, and also to manipulate these details meaningfully.

### Introduction :

The Hospital Management System can be entered using a valid username and password. It is accessible by an admin, doctors and receptionist. The data can be retrieved easily and is well protected for personal use with fast data processing. The system is easy to use, powerful and flexible and is designed to deliver real conceivable benefits to hospitals. It also provides excellent security of data at every level of user-system interaction and provides robust and reliable storage facilities.

- User View
- Logical View
- Physical View

### Logical View :

- ☐ DDL commands for tables
- ☐ Trigger Implementation
- ☐ SQL queries for fetching data
- ☐ Query Optimization

### DDL commands :

Tables :

- 1) **Doctor** : Doctors registered in the system

dr\_code : Primary key which uniquely identifies each doctor present in the system

name : Character type attribute (single valued) which stores the name of doctors

gender : Character type attribute storing gender for a doctor which can be F, M, O, Female, Male or Other

address : Character attribute to store address of the doctor

designation : Name given to each doctor's position for the job

- 2) **Patient** : Patients registered in the system
  - pat\_code : Primary key which uniquely identifies each patient registered in the system
  - name : Character type attribute (single valued) which stores the name of patients
  - gender : Character type attribute storing gender for a patient which can be F, M, O, Female, Male or Other
  - address : Character attribute to store address of the patient
  - age : Int valued attribute for storing age of patient
  - phone : Character type attribute for contact information of patient
  - dr\_code : Foreign key referenced from Doctor table for logical retrieval of data
- 3) **Staff** : Staff members of the hospital working under supervision of a doctor
  - staff\_id : Primary key which uniquely identifies each member of staff registered in the system
  - name : Character type attribute (single valued) which stores the name of staff members
  - department : Character type field specifying department under which the member is working
  - gender : Character type attribute storing gender for a staff member which can be F, M, O, Female, Male or Other
  - address : Character attribute to store address of the staff member
  - dr\_code : Foreign key referenced from Doctor table for logical retrieval of data
- 4) **Diagnosis** : Diagnostic results of a patient under examination of a doctor
  - diagnosis\_no : Primary key which uniquely identifies each diagnostic test performed
  - pat\_code : Primary key which uniquely identifies each patient
  - pat\_name : Character type attribute (single valued) which stores the name of patients
  - diagnosis : Field that stores the results of diagnostic tests performed
  - dr\_code : Foreign key referenced from Doctor table for logical retrieval of data
  - date : Datetime attribute that stores the date of diagnosis
  - remark : Any remarks or suggestions for the patient by the doctor
  - other : A Null acceptable field which tells if the doctor has some other comments or advice for the patient.
- 5) **Bill** : Statement of money that patient owes for services
  - billno : Primary key which uniquely identifies each bill generated
  - pat\_code : Foreign key referenced from Patient table for logical retrieval of data

pat\_name : Character type attribute (single valued) which stores the name of patients.

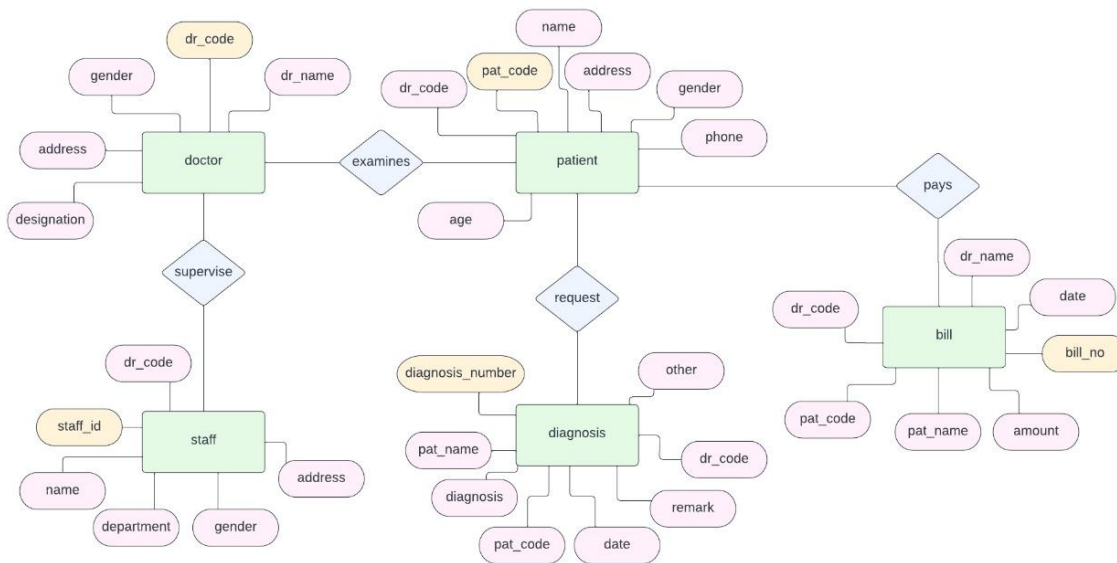
dr\_code : Foreign key referenced from Doctor table for logical retrieval of data.

dr\_name : Character type attribute (single valued) which stores the name of doctors.

date : Datetime attribute that stores the date of bill generation

amount : Integer value for the total charge for services provided to the patient

## ER Diagram



## Schema :

### Doctor

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 <b>dr_code</b> 🔑	varchar(10)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/>	2 <b>name</b>	varchar(100)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/>	3 <b>gender</b>	varchar(10)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/>	4 <b>address</b>	varchar(1000)	utf8mb4_general_ci		Yes	NULL			Change  Drop  More
<input type="checkbox"/>	5 <b>designation</b>	varchar(1000)	utf8mb4_general_ci		Yes	NULL			Change  Drop  More

## Patient

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	pat_code 🔑	varchar(10)	utf8mb4_general_ci		No	None			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	2	name	varchar(100)	utf8mb4_general_ci		No	None			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	3	gender	varchar(10)	utf8mb4_general_ci		No	None			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	4	address	varchar(1000)	utf8mb4_general_ci		Yes	NULL			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	5	age	int(11)			Yes	NULL			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	6	phone	varchar(12)	utf8mb4_general_ci		Yes	NULL			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	7	dr_code 🗝	varchar(10)	utf8mb4_general_ci		Yes	NULL			Change  Drop <a href="#">More</a>
























## Diagnosis

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	diagnosis_number 🔑	varchar(20)	utf8mb4_general_ci		No	None			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	2	pat_code 🗝	varchar(10)	utf8mb4_general_ci		Yes	NULL			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	3	pat_name	varchar(100)	utf8mb4_general_ci		Yes	NULL			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	4	diagnosis	varchar(1000)	utf8mb4_general_ci		Yes	NULL			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	5	date	datetime			Yes	NULL			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	6	remark	varchar(1000)	utf8mb4_general_ci		Yes	NULL			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	7	other	varchar(1000)	utf8mb4_general_ci		Yes	NULL			Change  Drop <a href="#">More</a>

## Staff

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	staff_id 🔑	varchar(10)	utf8mb4_general_ci		No	None			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	2	name	varchar(100)	utf8mb4_general_ci		No	None			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	3	department	varchar(100)	utf8mb4_general_ci		No	None			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	4	gender	varchar(10)	utf8mb4_general_ci		No	None			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	5	address	varchar(100)	utf8mb4_general_ci		Yes	NULL			Change  Drop <a href="#">More</a>
<input type="checkbox"/>	6	dr_code 🗝	varchar(10)	utf8mb4_general_ci		Yes	NULL			Change  Drop <a href="#">More</a>

## Bill

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	bill_no 	varchar(20)	utf8mb4_general_ci		No	None			 Change  Drop  More
<input type="checkbox"/>	2	pat_code 	varchar(10)	utf8mb4_general_ci		Yes	NULL			 Change  Drop  More
<input type="checkbox"/>	3	pat_name	varchar(100)	utf8mb4_general_ci		Yes	NULL			 Change  Drop  More
<input type="checkbox"/>	4	dr_code 	varchar(10)	utf8mb4_general_ci		Yes	NULL			 Change  Drop  More
<input type="checkbox"/>	5	dr_name	varchar(100)	utf8mb4_general_ci		Yes	NULL			 Change  Drop  More
<input type="checkbox"/>	6	date	datetime			Yes	NULL			 Change  Drop  More
<input type="checkbox"/>	7	amount	int(11)			Yes	NULL			 Change  Drop  More

## Trigger Implementation :

### 1) Adding dr\_code to Patient's table whenever a diagnosis test is done

```
// creating trigger to add dr_code in patient table after every insert in diagnosis table
snprintf(query6, 1024, "CREATE TRIGGER `update_dr_code` AFTER INSERT ON `diagnosis` FOR EACH ROW UPDATE patient SET patient.dr_code = new.other WHERE patient.pat_code = new.pat_code;");
int createTriggerStatus = mysql_query(conn, query6);
if (createTriggerStatus != 0) {
    cout<<"Error while creating trigger: "<<mysql_error(conn)<<endl;
}
```

### 2) Creating instance for Bill for every new diagnosis test

```
// trigger to create instance of bill after every insert in diagnosis table
snprintf(query7, 1024, "CREATE TRIGGER `create_bill` AFTER INSERT ON `diagnosis` FOR EACH ROW INSERT INTO bill (`pat_code`, `pat_name`, `dr_code`, `dr_name`, `date`, `amount`) values (new.pat_code, new.pat_name, new.dr_code, (select name from doctor where doctor.dr_code = new.dr_code), new.date, (SELECT RAND()*1000));");
createTriggerStatus = mysql_query(conn, query7);
if (createTriggerStatus != 0) {
    cout<<"Error while creating trigger: "<<mysql_error(conn)<<endl;
}
```

## SQL Queries :

Creating Table 'Doctor' (other tables were created in similar way)

```
// creating table
snprintf(query1, 1024, "CREATE TABLE `%s` (`dr_code` int primary key AUTO_INCREMENT,
`dr_name` varchar(100) not null, `gender` varchar(10) not null,
CONSTRAINT `check_gender` CHECK (gender IN ('M', 'Male', 'F', 'Female' , '0' , 'Other')),
`address` varchar(1000) null, `designation` varchar(1000));", tableName1);

int createTableStatus = mysql_query(conn, query1);
if (createTableStatus != 0) {
    cout<<"Error while creating table: "<<mysql_error(conn)<<endl;
}
```

### 1) All the patients a doctor with given name is examining

```
// All the patients a doctor with given name = A is examining
const char* query01 = "SELECT patient.pat_code, patient.name, patient.gender, patient.age
from patient cross join doctor where doctor.dr_code = patient.dr_code and doctor.dr_name = 'A'";
cout << "All the patients a doctor with given name = A is examining" << endl;
display(query01);
cout << endl;
```

### 2) Staff details working under supervision of a doctor

```
// Staff details working under supervision of a doctor
const char* query02 = "Select staff.staff_id, staff.name
from staff cross join doctor where doctor.dr_code = staff.dr_code and doctor.dr_name = 'A'";
cout << "Staff details working under supervision of a doctor" << endl;
display(query02);
cout << endl;
```

### 3) List of doctors from a specific department

```
// List of doctors from a specific department
const char* query03 = "select doctor.dr_name from doctor cross join staff where staff.department = 'Bhs'";
cout << "List of doctors from a specific department" << endl;
display(query03);
cout << endl;
```

#### 4) Diagnostic remarks for particular Patient

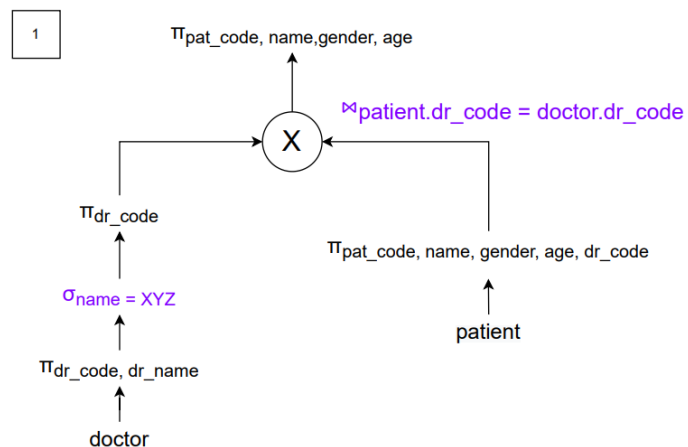
```
// Diagnostic remarks for particular Patient
const char* query04 = "SELECT diagnosis.diagnosis, diagnosis.remark, doctor.dr_name
from diagnosis, doctor where diagnosis.pat_name = 'XYZ' and diagnosis.dr_code = doctor.dr_code;";
cout << "Diagnostic remarks for particular Patient" << endl;
display(query04);
cout << endl;
```

#### 5) Total amount a patient has to pay

```
// Total amount a patient has to pay
const char* query05 = "select sum(bill.amount)
from bill cross join patient where patient.pat_code = bill.pat_code and patient.name = 'ABC';";
cout << "Total amount a patient has to pay " << endl;
display(query05);
cout << endl;
```

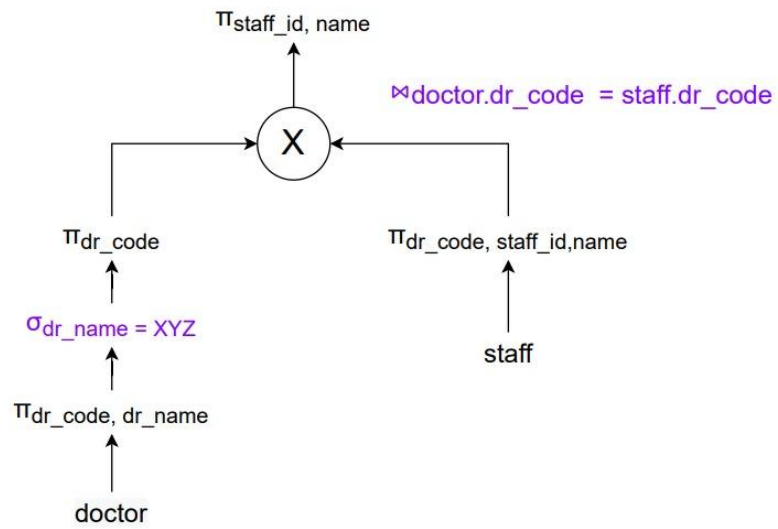
### Query Optimisation : Query Tree

#### 1) All the patients a doctor with given name is examining



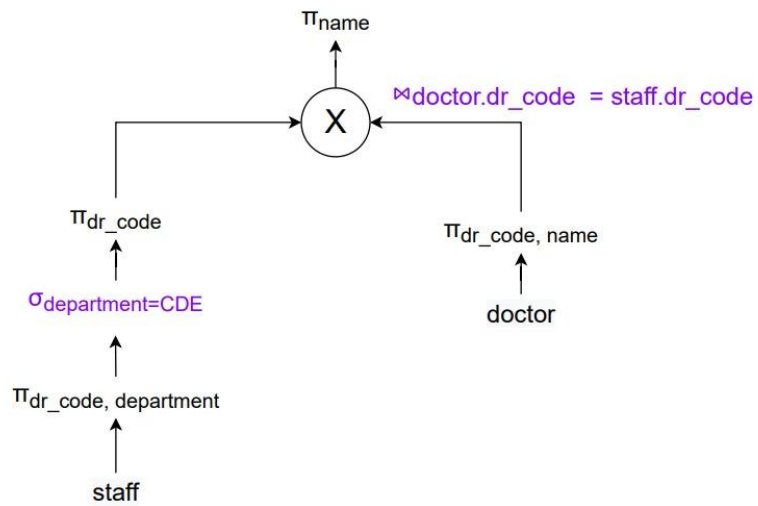
#### 2) Staff details working under supervision of a doctor

2



### 3) List of doctors from a specific department

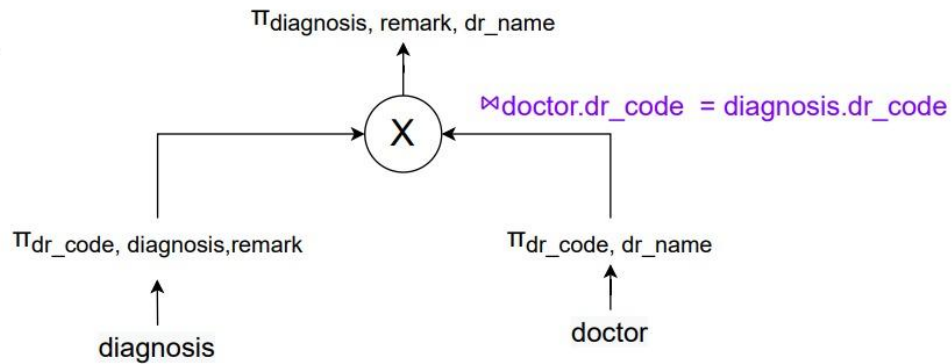
3



### 4) Diagnostic remarks for particular Patient

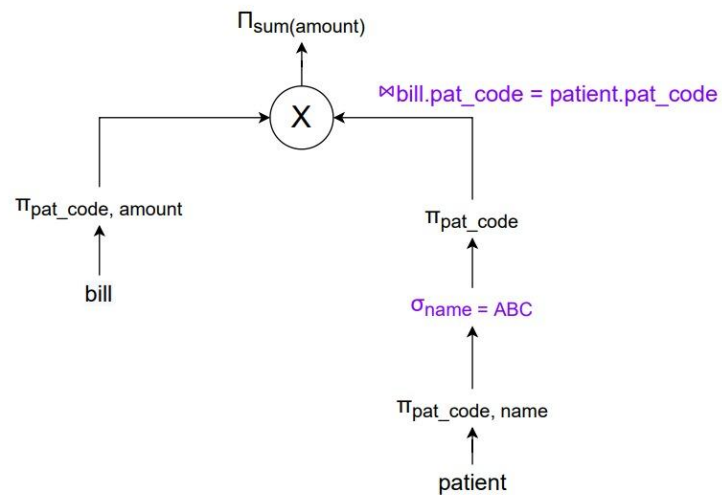


4



### 5) Total amount a patient has to pay

5



### User View :

We make a website where the user can view the details of various tables.

### Techstack:

Frontend- HTML,CSS and jinja text

Backend-Django (python)

Database-PostgreSQL

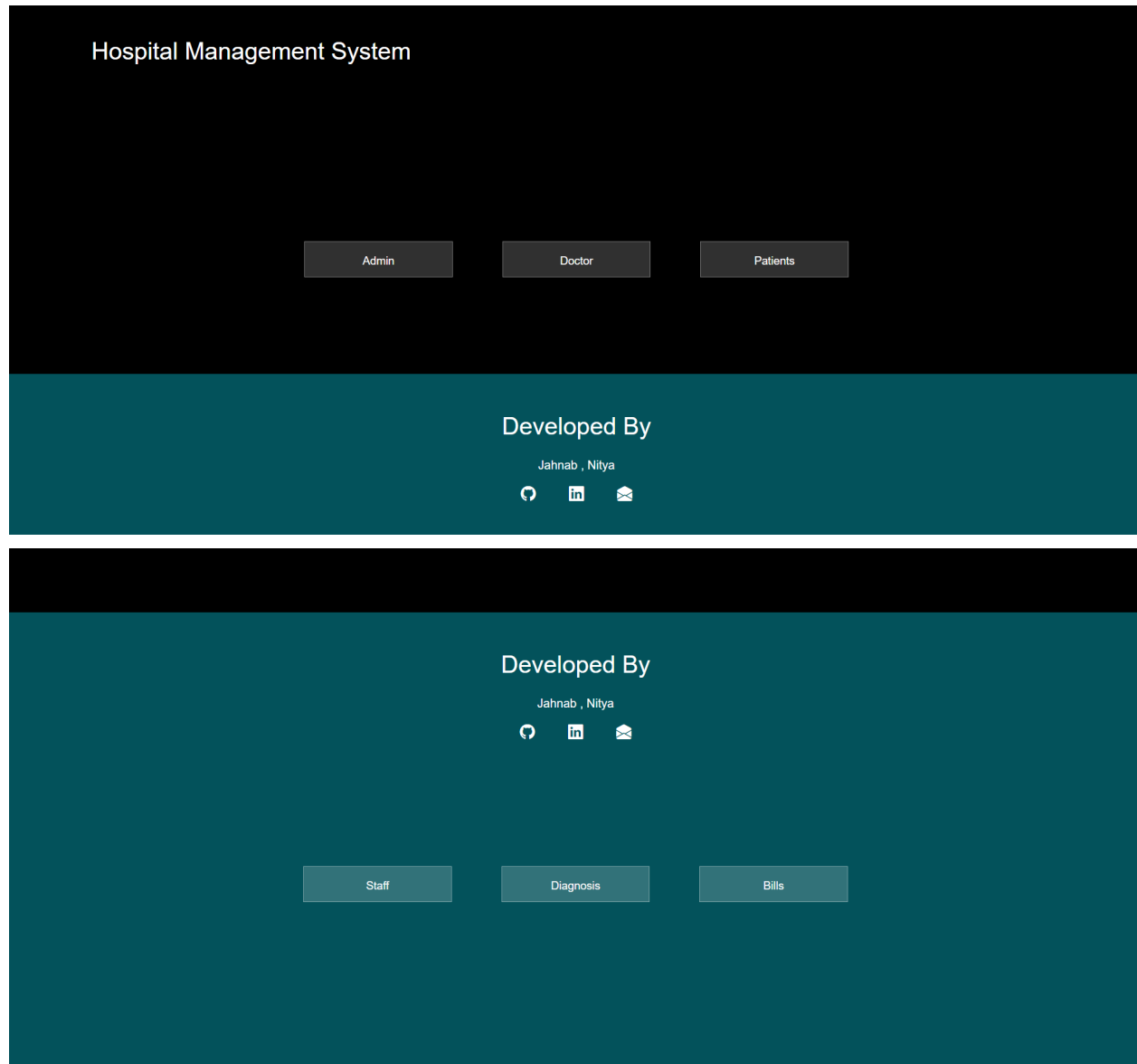
### Methodology:

- Used django models.py to create tables corresponding to the schema created in logical view.
- Used functions in views.py to render html web along with the data.
  - For example,
  - To make the doctors page, we make doctors.html.
  - We host 'doctor.html' from the functions 'doctors' in views.py

- We pass the database object as an argument to the webpage.
- We use jinja text to parse the data items to create the table visualization in the frontend.
- Implemented django admin panel in the website itself to give access to hospital admin to change the database.

## Demonstration:

Welcome page:



The doctors page:

List of Doctors			
<div>Filter</div> <div>Enter search ...</div>			
Doctor Code	Name	Gender	Specialization
0USMHD	Rohan Anish	M	Dermatology
7FB9S2	Anuja Prabhu	F	Genetics
AQPSI6	A. Mishra	M	Ortho
D0Q8E0	Anuj Dinesh	M	Anesthesiology
OV6MF3	Vimal Ramesh	M	ENT

Corresponding table in the postgres database:

Data output Messages Notifications					
	dr_code [PK] character varying (10)	name character varying (100)	gender character varying (10)	address character varying (1000)	designation character varying (100)
1	0USMHD	Rohan Anish	M	Hostel G5, IIT Jodhpur H...	Dermatology
2	7FB9S2	Anuja Prabhu	F	HS. NO.29,2nd FLOOR	Genetics
3	AQPSI6	A. Mishra	M	IIT Jodhpur, Karwar	Ortho
4	D0Q8E0	Anuj Dinesh	M	Hostel G5, IIT Jodhpur H...	Anesthesiology
5	OV6MF3	Vimal Ramesh	M	IIT Jodhpur, Karwar	ENT

Similarly we have pages for all the tables.

We also have an admin page:

Django administration		WELCOME, root. VIEW SITE / CHANGE PASSWORD / Log out
Home · Website · Staffs		
«		
Start typing to filter...		
AUTHENTICATION AND AUTHORIZATION		
Groups		+ Add
Users		+ Add
WEBSITE		
Bills		+ Add
Diagnosis		+ Add
Doctors		+ Add
Patients		+ Add
Staffs		+ Add

Addition of new doctor via admin:

## Add Doctor

Dr code:	<input type="text" value="FJY4RY"/>
Name:	<input type="text" value="Apoorva Aishwarya"/>
Gender:	<input type="text" value="female"/>
Address:	<input type="text" value="Hostel G5, IIT Jodhpur Hostel Complex F4F85"/>
Designation:	<input type="text" value="Dermatology"/>

Value displayed in postgres database and website as well:

List of Doctors			
Filter			
Enter search ...			
Doctor Code	Name	Gender	Specialization
0USMHD	Rohan Anish	M	Dermatology
7FB9S2	Anuja Prabhu	F	Genetics
AQPSI6	A. Mishra	M	Ortho
D0Q8E0	Anuj Dinesh	M	Anesthesiology
FJY4RY	Apoorva Aishwarya	F	Dermatology
OV6MF3	Vimal Ramesh	M	ENT

	dr_code [PK] character varying (10)	name character varying (100)	gender character varying (10)	address character varying (1000)	designation character varying (100)
1	0USMHD	Rohan Anish	M	Hostel G5, IIT Jodhpur H...	Dermatology
2	7FB9S2	Anuja Prabhu	F	HS. NO.29,2nd FLOOR	Genetics
3	AQPSI6	A. Mishra	M	IIT Jodhpur, Karwar	Ortho
4	D0Q8E0	Anuj Dinesh	M	Hostel G5, IIT Jodhpur H...	Anesthesiology
5	FJY4RY	Apoorva Aishwarya	F	Hostel G5, IIT Jodhpur H...	Dermatology
6	OV6MF3	Vimal Ramesh	M	IIT Jodhpur, Karwar	ENT

Value added in 2nd last row.

Features:

1. We can create a tuple for 'staff' or 'patients' without assigning them the foreign key. Later we can assign the doctor via some function or post request.
2. We can create a diagnosis with the proper doctor and patient ids and it will update the patient of the patient table with the correct dr\_code.
3. We can search in a table using all the attributes for that table.
4. Admin panel to change the database without having to make actual sql queries.
5. Visualization of the data in pgAdmin of postgresSQL database.
6. We can easily migrate from one type of database to another by just changing the config file. We can use-
  - a. Mysql server
  - b. Sqlite db
  - c. Microsoft SQL
  - d. PostgresSQL

(explained in the video properly)

## Physical View :

Creation of a simulation to show storing of data from an entity into the database with hashing.

Table chosen- 'doctors'

Hashing method- 'Extendible hashing'

Bucket Size- variable.

No. of initial Entries-16

Hash key- primary\_key

## Methodology:

- Used rolling hash function on 'dr\_code' to create hash keys.
- Used 'Row' data structure to simulate a tuple.
- Created to map<string,vector<Row>> to simulate Hash Table.
- Made extendible hashing class with insert, find and print facility.
- Created interactive simulation for user to user to interact with the database.
- Added method for user to analyze the effect of bucket size and number of splits.

## Demonstration:

Upon running 'doctors.cpp' file we have the following prompt-

```
Choose 1 if you want to analyse hashing for differnt bucket sizes
Choose 2 if you want to simulate extendible hashing
```

Let us choose '2' to simulate extendible hashing-

```
2
Enter the maximum bucket size : 4
1. Insert
2. Search
3. Print
4. Exit
```

Let us print first.

```
0 :

00 :

000 :
Dr. Code : D2AT6Y
Dr. Name : Dr. A.Mishra
Gender: M
Address : B-1/2, Sector-5, Noida
Designation : Consultant

01 :
Dr. Code : D2AT79
Dr. Name : Dr. L.Mishra
Gender: M
Address : B-1/2, Sector-5, Noida
Designation : Consultant

1 :

10 :
Dr. Code : D2AT70
Dr. Name : Dr. C.Mishra
Gender: M
Address : B-1/2, Sector-5, Noida
Designation : Consultant
Dr. Code : D2AT71
Dr. Name : Dr. D.Mishra
Gender: M
Address : B-1/2, Sector-5, Noida
Designation : Consultant
```

We can see there are no entries in the bucket with '0' or '00' or '1' but there are entries in '000' and '10'.

Let us now search for a non existent value-

```
2
Enter the dr_code to search : D2ABCD
Not Found
1. Insert
2. Search
3. Print
```

We can see that it is not found. We can add the value and then search again.

```
1
Enter the details of the doctor
Dr. Code : D2ABCD
Dr. Name : A.Poori
Gender :F
Address : 22_nd_Stree
Designation : ortho
```

Now let's search again.

```
Enter the dr_code to search : D2ABCD
Found
Details :
Dr. Code : D2ABCD
Dr. Name : A.Poori
Gender: F
Address : 22_nd_Stree
Designation : ortho
```

As we can see that the data is found.

Now let us analyze with different bucket sizes and count the number of splits.

```
Enter the number of instances : 5
Enter the bucket size : 1
Enter the bucket size : 2
Enter the bucket size : 3
Enter the bucket size : 4
Enter the bucket size : 16
Bucket Size : 1 Number of Splits : 18
Bucket Size : 2 Number of Splits : 11
Bucket Size : 3 Number of Splits : 8
Bucket Size : 4 Number of Splits : 3
Bucket Size : 16 Number of Splits : 0
```

### Tabulation and analysis:

Bucket size	Number of splits
1	18
2	11
3	8
4	3
16	0

We can see that the number of splits and bucket size are inversely proportional. However, as bucket size increases the total time for search also increases. Therefore, 4 is an optimal bucket size.

### Procedure for project implementation :

- Agile methodology of development
- First phase consisted of designing the schema and normalization
- Second phase consisted of the creation of a logical view for implementation of schema.
- Third phase consisted of creation of user view creation using principles from logical view.
- Third phase consisted of the creation of a physical view to analyze the earlier views.

### Contribution :

- Jahnab Dutta (B20CS091) -Physical view and User view hands on creation and help in logical view design
- Nitya Anand Shah (B20CS039) - Logical view implementation and code review of physical and user view.
- Niharika Manhar (B20CS038) - Trigger creation for logical view and used Lucidcharts for query trees.

Thank you.