

HOSPITAL MANAGEMENT SYSTEM

DBS PROJECT 6

Parth Patel 2020A7PS0026P

Anish Kulkarni 2020A7PS0975P

1. System requirement specification (SRS)

1. Nodejs ([download](#))

2. MySQL Workbench 8.0 CE

3. MySQL

4. use 'npm i' in VSCode to install all required packages of package.json

For the frontend of the website HTML, CSS, javascript is used and backend is done using node.js framework. The database management is done using MySQL

To run the server install node.js from above link

We will have to install npm modules by going into DBS_PR_06_Code_2020A7PS0975P folder and running 'npm i'.

To use the server we need to initialize the database by changing the username and password. Then run the queries

```
' ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '1234';  
flush privileges; '
```

INSTRUCTIONS

1. extract folder from zip.

2. Make sure node is installed by typing node -v in cmd.

3. copy DBS_PR_06_SQL_2020A7PS0975P.sql file contents in MySQL Workbench (make sure username is set to 'root' and password to '1234'. Query for this is already written in the .mysql file)

[For reference queries

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '1234';
```

flush privileges;

]

4. Select all queries of DBS_PR_06_SQL_2020A7PS0975P.sql file and execute them.

5. Open folder DBS_PR_06_Code_2020A7PS0975P in vscode.

6. run 'npm i' in the vscode cmd. This will install various files.

7. run 'node index.js' in vscode terminal

8. open 'localhost:3000' in browser.

2. System Modelling

a) Entity-Relationship (ER) diagram

We can see from the ER diagram that there is total participation from the entity staff and it has a many to one relationship with room.

Due to this we can combine the relation 'alloted_to' and staff to create just one table.

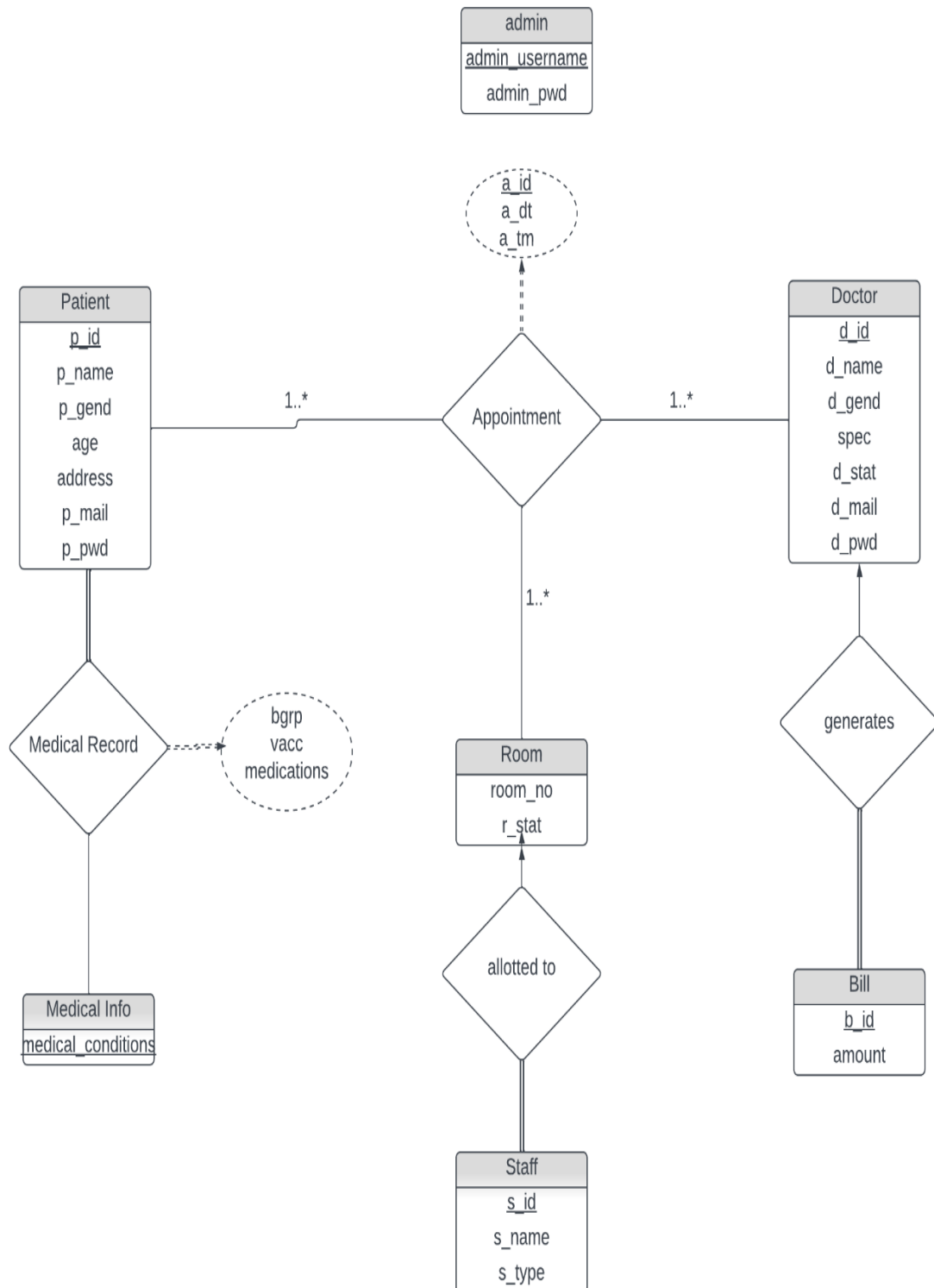
Similarly, we can combine entity 'bill' and relation 'generates' into just one table.

A double-line show total participation.

Appointment is a ternary relationship between Patient, Doctor and Room.

HOSPITAL MANAGEMENT SYSTEM

(2020A7PS0026P , 2020A7PS0975P)



b) Schema Design & Data Normalization

1. *patient(p_name, p_gend, age, address, p_id, p_pwd, p_mail)*

patient(A,B,C,D,E,F,G)

Prime Attributes: E,G

Non-Prime Attributes: A,B,C,D,F

Functional Dependencies: E->ABCD FG, G->ABCDEF

For every non-trivial functional dependency LHS is CK/ SuperKey so table is in 3NF

Here (A,B,C,D,E,F,G) correspond to (p_name ,p_gend , age, address, p_id ,p_pwd ,p_mail)

2. *doctor(d_name, d_gend, spec, d_id, d_pwd, d_mail, d_stat)*

doctor(A,B,C,D,E,F,G)

Prime Attributes: D, F

Non-Prime Attributes: A,B,C,E,G

Functional Dependencies: D->ABCEFG, F->ABCDEG

For every non-trivial functional dependency LHS is CK/ SuperKey so table is in 3NF

Here (A,B,C,D,E) correspond to (d_name ,d_gend , spec, d_id ,d_pwd ,d_mail,d_stat)

3. *admin(admin_username, admin_pwd)*

Only 1 functional dependency from admin_username -> admin_pwd

Table is in 3nf

4. *appointment(a_id, a_dt, a_tm, p_id, d_id, room_no)*

a_id is primary key.

Table constraints ensure that the same doctor or room cannot be reselected at a given date and time while booking an appointment. Each slot is of 1hr

No transitive or partial dependencies exist. Table in 3nf

5. *bill(b_id, d_id, amount)*

b_id primary key. No other functional dependencies. 3nf.

6.medrecords(p_id,bgrp,vacc,medications,medical_conditions)

This table has a transitive dependency from p_id to vacc(status) and bgrp . This can easily be removed by adding vacc and brp to patient table. However, while registering using the HTML form we don't want the user to enter information which is not necessary at that point of time. Blood group and vaccination status can be added later to the patients medical record in case he has to take further steps on advice of the doctor

7.medical_info(medical_conditions)

Only one column. Fully normalized.

8.room(room_no, r_stat)

room_no is PK. Normalized.

9.staff(s_id,room_no,s_type, s_name)

s_id is primary key. No functional dependencies violate 3nf.

c) List of Tables

1) Admin Table

Field	Type	Null	Key	Default
admin_username	varchar(45)	NO	PRI	NULL
admin_pwd	varchar(45)	NO		NULL

- It has the details of the username and password of the admin.
- Admin can login using the above details and keep in check the overall management and functioning of the hospital.

2) Patient Table

Field	Type	Null	Key	Default	Extra
p_name	varchar(15)	NO		NULL	
p_gend	varchar(10)	NO		NULL	
age	int	NO		NULL	
address	varchar(45)	NO		NULL	

p_id	int	NO	PRI	NULL	auto_increment
p_pwd	varchar(10)	NO		NULL	
p_mail	varchar(45)	NO		NULL	

- Stores all the personal details of the patient, namely name, gender, age, address, ID, mail and password.
- This table will be populated whenever the register patient form is filled and submitted on the frontend. Procedure for the same is also made.
- Instructions like insert and delete patient can also be executed.

3) Medical Records Table

Field	Type	Null	Key	Default
p_id	int	NO	PRI	NULL
bgrp	varchar(3)	NO		NULL
vacc	int	YES		NULL
medications	varchar(3)	NO		NULL
medical_conditions	varchar(45)	NO	PRI	NULL

- This table is of a weak relation medical record, whose supporting strong entity is Patient table.
- The primary key of the strong entity is also included in the primary key of this table.
- This table arises due to a many to many relationship between patient and medical info.
- This table stores the medical history of the patient including his blood group, vaccination status, or any medications.
- Other insert, delete and update operations can be executed easily.

4) Medical Info Table

Field	Type	Null	Key	Default	Extra
medical_conditions	varchar(45)	NO	PRI	NULL	

- Stores all diseases or medical conditions that the hospital is currently or has treated patients on in the past.
- Is a part of the primary key of the medrecords relation.

5) Doctor Table

Field	Type	Null	Key	Default	Extra
d_name	varchar(15)	NO		NULL	
d_gend	varchar(10)	NO		NULL	
spec	varchar(15)	NO		NULL	
d_id	int	NO	PRI	NULL	auto_increment
d_pwd	varchar(10)	NO		NULL	
d_mail	varchar(45)	NO		NULL	
d_stat	varchar(5)	YES		Yes	

- Stores all the personal details of the doctor, namely name, gender, area of specialization, ID, mail and password.
- It will be populated whenever a doctor gets affiliated with the hospital.
- It also gives the status of the doctors with respect to their corresponding ID numbers. This will clearly give an insight about which doctors are free/unavailable at a given point of time.
- Other instructions like insert and delete doctor statements can also be executed.

6) Appointment Table

Field	Type	Null	Key	Default	Extra
a_id	int	NO	PRI	NULL	auto_increment
a_dt	int	NO	MUL	NULL	
a_tm	int	NO		NULL	
p_id	int	YES	MUL	NULL	
d_id	int	YES	MUL	NULL	
room_no	int	NO	MUL	NULL	

- Stores information about the appointments and their respective details like date, time, the corresponding patient and doctor IDs and the room in which they will meet.
- An appointment will be created/inserted when a registered patient books an appointment for a specific doctor and room of his choice. A patient enters the date and time he wishes to book the appointment on. The he can see the list of doctors having the desired specialization who are available on that time slot. Similarly, he can choose from available rooms and book the appointment.
- There are two table-level constraints for Appointment table that ensure a unique combination of date, time and room values & date, time, doctor ID values.
- This ensures that a doctor cannot be selected for 2 different rooms at the same date/time.

- The appointment time is an integer from (6 to 24) which corresponds to the kth hour of the day on a given date.
- [An assumption here forth is that an appointment can last for at most an hour.]

7) Room Table

Field	Type	Null	Key	Default	Extra
room_no	int	NO	PRI	NULL	auto_increment
r_stat	varchar(5)	NO		NULL	

- Gives the room status along with the room's corresponding room number.
- This helps in allocating rooms to appointments, while checking that the room is free or not.

8) Bill Table

Field	Type	Null	Key	Default	Extra
b_id	int	NO	PRI	NULL	auto_increment
amount	int	NO		NULL	
d_id	int	YES	MUL	NULL	

- Generates a bill for the patient.
- It has a constraint that ensures that a bill can only be generated by one doctor.
- It has its own bill ID, the amount to be paid and the corresponding doctor ID.

9) Staff Table

Field	Type	Null	Key	Default	Extra
s_id	int	NO	PRI	NULL	auto_increment
room_no	int	YES	MUL	NULL	
s_type	varchar(10)	NO		NULL	
s_name	varchar(45)	NO		NULL	

- Stores details of the workers/professionals other than doctors.

- It has the attributes namely, staff ID, name, type of worker and the room number he is associated with.

d) Additional Components

- Procedures

1) p_insert Procedure

- It is called when a patient tries to register himself in the hospital database.
- Patient fills out the form (on the website), and when submit button is clicked, p_insert procedure is called with the IN parameters being the contents of the Patient table.
- The procedure call statement is as follows:

```
CALL `project`.`p_insert`(<{IN pname VARCHAR(15)}>, <{IN pgender VARCHAR(10)}>,
<{IN age INT}>, <{IN paddr VARCHAR(45)}>, <{IN ppwd VARCHAR(10)}>, <{IN pmail
VARCHAR(45)}>);
```

2) d_insert Procedure

- It is called when the admin tries to enrol a new doctor in the hospital database.
- Admin fills out the form (on the website), and when submit button is clicked, d_insert procedure is called with the IN parameters being the contents of the Doctor table.
- The procedure call statement is as follows:

```
CALL `project`.`d_insert`(<{IN dname VARCHAR(15)}>, <{IN dgender VARCHAR(10)}>,
<{IN spec VARCHAR(15)}>, <{IN dpwd VARCHAR(10)}>, <{IN dmail VARCHAR(45)}>);
```

3) medrec_insert Procedure

- It populates the medical history of the patient with his corresponding patient ID.
- The patient fills up a form that requires his ID, blood group, vaccination dose, medications, or any previous illnesses for the medrec_insert IN parameters.
- The procedure call statement is as follows:

```
CALL `project`.`medrec_insert`(<{IN pid INT}>, <{IN bg VARCHAR(3)}>, <{IN vacc
INT}>, <{IN medic VARCHAR(3)}>, <{IN medhist VARCHAR(30)}>, <{IN previll
VARCHAR(3)}>);
```

4) medinfo_insert Procedure

- It keeps a record of which all diseases or medical conditions, the hospital is currently (or in the past) treating on.
- The procedure call statement is as follows:

```
CALL `project`.`medinfo_insert`(<{IN medcond VARCHAR(45)}>);
```

5) a_insert Procedure

- It forms just *a part of* the entire functionality of Booking an appointment.
- a_insert is called to insert a row in the Appointment table.
- IN parameters of date, time, and the respective patient & doctor IDs and room number are passed.
- The procedure call statement is as follows:

```
CALL `project`.`a_insert`(<{IN adt DATE}>, <{IN atm INT}>, <{IN pid INT}>, <{IN did INT}>, <{IN rno INT}>);
```

6) room_insert Procedure

- It does the simple action of inserting into the hospital database a room, with its room number and status details.
- The procedure call statement is as follows:

```
CALL `project`.`room_insert`(<{IN rno INT}>, <{IN rstat VARCHAR(5)}>);
```

7) s_insert Procedure

- Allots a specific staff worker to a room he is supposed to work in. Parameters passed are room number, type and name of staff.
- The procedure call statement is as follows:

```
CALL `project`.`s_insert`(<{IN rno INT}>, <{IN stype VARCHAR(10)}>, <{IN sname VARCHAR(45)}>);
```

8) bill_insert

- Populates the bill table with an entry of the corresponding appointment number's bill and amount.
- The procedure call statement is as follows:

```
CALL `project`.`bill_insert`(<{IN amt INT}>, <{IN aid INT}>);
```

9) choose_doc_spec Procedure

- Procedure to get a list of doctors of a particular input specialization and their details and status.
- It can be useful for admin or patients to check and choose doctors.
- The procedure call statement is as follows:

CALL `project`.`choose_doc_spec`(<{IN dspec VARCHAR(15)}>);

10) hosp_patient_hist Procedure

- Procedure to find the history of that particular patient with different specialization doctors in the hospital
- Useful for the admin or doctor to check the history/illnesses of the patient and which all doctors he has consulted.
- The procedure call statement is as follows:

CALL `project`.`hosp_patient_hist`(<{IN pid INT}>);

11) hosp_doc_hist Procedure

- Procedure to find the history of a particular doctor and all the patients he has treated
- Useful for the admin or doctor to check his work and record of patients and appointments
- The procedure call statement is as follows:

CALL `project`.`hosp_doc_hist`(<{IN did INT}>);

12) previous_appointments Procedure

- Procedure to get a list of all previous appointments with the respective patient and doctor.
- Useful for the doctor to keep an update regarding the recovery with successive appointments with his patient.
- The procedure call statement is as follows:

CALL `project`.`previous_appointments`(<{IN pid INT}>, <{IN did INT}>);

• Views

1) p_login

- View to get a table of mail and password details of registered patients.
- Admin can use it in case patient wants to view login details – email-id or forgets password.

2) d_login

- View to get a table of mail and password details of registered doctors.
- Admin can use it in case doctor wants to view login details – email-id or forgets password.

3) doc_status

- View to get a table of availability of doctors with their specialization and status.
- Also, there is a column of associated appointment ID, if doctor is not free.

4) vacc_status

- View to get a table of vaccination status of all registered patients.
- Admin can use it if the hospital plans to organize a vaccination drive for the second dose.
- Doctor can use it to advise based on whether the patient is doubly or singly vaccinated.

5) room_work

- View to get a table of all patients, doctors, staff associated with a room.
- Useful for the admin to check which all professionals/workers & patients are allotted in which room.

• DML Operations

1) Delete operations

- A doctor leaves and is no longer associated with the hospital.
- The admin needs to delete his entry, giving the doctor ID and delete him from the database.
- A patient feels well and wants to cancel his appointment.
- The appointment table will reflect that the appointment on that date, at that time, with the respective patient is deleted.

- A room in the hospital is undergoing maintenance
- Admin wants to delete that room, because it would not be available

- A staff is on leave/ or has joined another hospital.
- The admin wants to delete the entry of this particular staff member.

2) Update operations

- Patient wants to update his email-id and password details.
- Admin can update them, based on the particular patient ID

- Doctor wants to update his email-id and password details.
- Admin can update them, based on the particular doctor ID

- Staff's allotted room needs to be changed.
- Admin can reallocate certain staff IDs, staff types to a different room

- A patient needs to reschedule his appointment.
- He can change the appointment details to a different date and time.

- The patient also bought medicines from the doctor.
- Doctor needs to add medicine charges to consultation fees to update the billing amount.

- Room allotted/ booked for an appointment is now free
- Admin wants to update its status to available

- Transactions

- 1) Register Patient transaction

- The registration process of a patient needs to be executed fully, or else there will be some blank fields.
- A transaction ensures atomicity, and that it will be committed completely.

START TRANSACTION;

CALL p_insert(p_name, p_gend, age, address, p_pwd, p_mail);

COMMIT;

- 2) Book an Appointment transaction

- The appointment booking needs to be isolated for every patient.
- Also, it needs to be executed completely.
- A transaction for displaying the available options and then booking an appointment will be followed.

- Triggers

- 1) Add_GST Trigger

- Hospital services generally incur a GST of 18%
- This trigger will automatically include and add the GST charges into the final bill amount.
- Trigger statement is as follows:

SET NEW.amount = NEW.amount*1.18;

- 2) Doc_status_change Trigger

Before booking an appointment and after deleting an appointment, this trigger will change the status of room and doctor.

UPDATE doctor SET doctor.d_stat='Yes'

WHERE doctor.d_id=OLD.d_id;

UPDATE room SET room.r_stat='Yes'

WHERE room.room_no=OLD.room_no;

3) Frontend Development


Attached below are the screenshots of the website, with execution of some functionalities:

Hospital

Log InAdminLinks

Register / Login

Already have an account? Login.



Register here

Name

Your name..

Gender

Male

Age

Your age..

Address

Your address..

E-mail

Your mail..

Password

Your password..

Submit

Hospital

Log InAdminLinks

Login here


E-Mail

Your email..

Password

Your password..

Submit




Doctors

Hospital

Log InAdminLinks

High quality.



Admin Login only

Username

Your username..

Password

Your password..

Submit

Please Enter

E-Mail

ok

Please Enter

Specialization

Patient ID

Choose date and time for appointment

Date

Time

Check available doctors

Check available rooms

Check available doctors

ok

Check available rooms

ok



Book an appointment

PatientID

DoctorID

RoomNo.

Submit



Show patient details

ok

- (NAME , GENDER , AGE , ID , PASSWORD)
- (Anish , Male , 12 , 1 , anish179 ,)
- (Rani , Female , 24 , 1522 , rani69 ,)
- (Ram , Male , 82 , 1523 , ramroxx ,)
- (Sachin , Male , 71 , 1524 , sachinsuxx ,)
- (Shreya , Female , 21 , 1525 , shreyas ,)
- (Hans , Male , 26 , 1526 , hansraj ,)
- (Mia , female , 25 , 1527 , 1234a ,)
- (Mia , female , 25 , 1528 , 1234a ,)

Show doctor details

ok

- (NAME , GENDER , SPECIALIZATION , ID , PASSWORD)
- (Narendra , Male , Ortho , 100 , pranam17 ,)
- (Nimbol , Male , General , 813 , nimbroxx ,)
- (Tejomoy , Male , Neuro , 814 , tejo131 ,)
- (Mitali , Female , Dentist , 815 , mitaliop ,)
- (Tanmoy , Male , Ophthalmology , 816 , moytan ,)
- (Ramesh , Male , Homeopathy , 817 , ramesh12 ,)
- (Shruti , Female , General , 818 , shruti00 ,)
- (Ila , Female , Dentist , 819 , ilailu ,)
- (sidhi , female , Neuro , 820 , sid66 ,)

Show Appointments

ok

- (AppointmentID , Date , Time , PatientID , DoctorID , RoomNo)
- (152 , 2022-11-15 , 7 , 1523 , 813 , 2204)
- (153 , 2022-11-15 , 8 , 1524 , 814 , 1233)
- (154 , 2021-10-11 , 12 , 1525 , 815 , 2204)
- (158 , 2021-10-11 , 12 , 1526 , 816 , 1233)
- (161 , 2021-05-04 , 11 , 1523 , 815 , 6101)
- (162 , 2022-11-30 , 6 , 1526 , 100 , 1233)