Hospital Management System project

Team Members :

Ashish Kumar Gupta	180050013	180050013@iitb.ac.in
Jagdish Suthar	180050039	jsuthar@iitb.ac.in
Kaushal U	180050047	180050047@iitb.ac.in
Shivanshu Gour	180050099	180050099@iitb.ac.in

a) Requirements:

An online solution for managing activities (like booking appointments with a doctor, booking a slot for a test/report like X-ray, CT scan etc) and data (about hospital staff, doctors and patients etc) in a hospital.

Getting insights on various diseases(i.e trends of a disease at particular location) by using the past records of all patients.

Requirements as per User:

1) Doctor
   - Doctor can see the all the upcoming appointments and if it is done then can mark them complete, also doctor can cancel some appointments in emergency situations
   - After diagnosing patients symptoms doctor would write some medicine prescription or doctor would like to see old prescription given to patients and in some situation can modify the prescription
   - Doctors can see reports of their patients
   - Doctors can get their patients information

2) Patient:
   - A patient will be able to see his past history, details of doctors in the hospital.
   - A patient can book an appointment with a **doctor** or for a **Laboratory Test prescribed by a doctor** by paying money from their wallet and can choose a slot from available ones.
   - Patient's account will also have a wallet where they can put their money by recharging and they can spend the money for availing various services of the hospital efficiently.
   - Patient can verify a payment initiated by pharmacy_keeper to purchase medicines

3) Accountant:
   - Add money to patient's wallet
   - give back money from wallet

4) Pharmacy Store Keeper:
   - Pharmacy_store keeper will be able to get a list of prescribed medicines of a patient by entering his ID into the system and can accept payment of medicines from the patient's wallet.
   - Pharmacy_store keeper will be updating the medicine in the database according to the real time activity.

5) Pathologist:
    - Pathologists can add the generated reports to the database.
    - Pathologists can see prescribed tests and medicine and reports of the user generated in the past.

6) Director:
    - Director can add/remove a staff member from the hospital database
    - Can see statistics/analytics of the data stored at the hospital

7) Admin:
    - Can see statistics and analytics of the data stored at the hospital

b) Use Cases:

USE CASES OF EACH ROLES:

1) Doctor

a) Can see,cancel and mark complete each appointments
   i) Can see,cancel and mark complete each appointments
   ii) Doctor can see the all the upcoming appointments and if it is done then can mark them complete, also doctor can cancel some appointments in emergency situations
   iii) Human trigger
   iv) Doctor
   v) Click on see and for each appointment can click on complete/cancel
   vi) No preconditions
   vii) Interaction-> clicking, output-> a message will be displayed saying “marked as complete” or “cancelled”
   viii) No exceptions
   ix) Appointment status may change or remains same

b) Write,modify and see prescription
   i) Write,modify and see prescription
   ii) After diagnosing patients symptoms doctor would write some medicine prescription
   Or doctor would like to see old prescription given to patients and in some situation can modify the prescription
   iii) Triggered by doctor
   iv) Only doctor is interacting with system
   v) **Input->** patient_id, appointment_id **, no constraint**
   vi) there must be prescription associated with that patient_id then only a doctor can modify
   vii) Doctor will feed the patient_id **system->** check constraints and output all prescriptions of that patient with doctor then doctor can select a particular prescription, system will output details of that prescription then doctor can modify that.
   viii) Exceptions-> patient_id doesn’t exist
   ix) New prescription will be added or an old prescription would be modified

c) Can See test reports
   i) Can See test reports

ii) Getting a patient's test report is important for diagnosis of disease and prescription of medicine
iii) Triggered by doctor (human trigger)
iv) Only doctor is interacting
v) **Input**-> patient_id , **Constraints**-> valid patient_id and there is a test a
vi) patients must have gone through test
vii) Doctor will put patient_id then system will output list of tests that patients have gone through then doctor will select a test and report of that will be shown
viii) Exceptions-> when report is pending or test was not done properly(sample has some defect)
ix) System will remain in same state

d) Get Patients info
i) Get Patients info
ii) Age,gender, height, weight and some more critical information are required for diagnosis
iii) Triggered by doctor (human trigger)
iv) Only doctor is interacting
v) Input-> patient_id, Constraints-> valid patient id
vi) That patient must have been registered
vii) Doctor will put patient id and system will show all the information about that patient
viii) If some information is missing (ie gender, weight)
ix) System will remain in same state

2) Patient

*a) A patient can book an appointment with a **doctor** by paying money from their wallet and can choose a slot from available ones.*

1) Booking an appointment with a doctor
2) With this use case a patient can book an appointment to meet with a doctor
3) Will be triggered by a patient (human trigger)
4) Only a patient is interacting with the system
5) ***Inputs*** :- doctor_id, slot_id, date, ***Constraints*** :- money in his wallet must be greater than or equal to the fee of doctor, slot must not be fully filled, date and slot must not correspond in past time.
6) No precondition
7) ***User-***> select a doctor and date, ***System-***> will show availability of slots, ***User-***> choose a slot, ***System-***> will show fee and details about appointment to be booked and pay button ***User-***> click on pay and book button, ***System-***> will check constraints and book an appointment by deducting money from wallet and will show success message on success.
8) Exceptions -> slot is full, not enough balance in wallet (failure message)
9) System will have a new appointment if all goes well else it will remain the same

*b) A patient can book an appointment for a **Laboratory Test prescribed by a doctor** by paying money from their wallet and can choose a slot from available ones.*

1) Booking an appointment for a laboratory test.
2) With this use case a patient can book an appointment for a laboratory test prescribed by a doctor.

3) Will be triggered by a patient (human trigger)
4) Only a patient is interacting with the system
5) ***Inputs*** :- prescription_id, test_id, slot_id, date, ***Constraints*** :- doctor must have prescribed the test in the prescription, money in his wallet must be greater than or equal to the fee of lab_test, slot must not be fully filled, date and slot must not correspond in past time.
6) There must be an entry for the test in the prescription whose report is not available.
7) ***User->*** click on book appoint button of the corresponding test entry in the prescription, **System->** will ask for a date **User->** enter a date ***System->*** will show availability of slots, ***User->*** choose a slot, ***System->*** will show fee and details about appointment to be booked and pay button ***User->*** click on pay and book button, ***System->*** will check constraints and book an appointment by deducting money from wallet and will show success message on success.
8) Exceptions -> slot is full, not enough balance in wallet (failure message), not prescribed by a doctor.
9) System will have a new appointment for lab_test if all goes well else it will remain the same

c) Add money to the wallet from bank

1) Add money to the wallet
2) Patient can load money in his wallet from a bank
3) Human trigger
4) Only a patient is interacting
5) **Inputs->** amount to be added **Constraints->** amount >= 1
6) No precondition
7) **User->** enter amount to be added **System->** create a pay order of given amount and redirect to a payment gateway **User->** will complete his payment and payment gateway will redirect back to the system **System->** will check the status of payment and on success will add money to the wallet.
8) Exceptions-> payment fails, amount outside constraints
9) User will have the amount added to his wallet if all goes well, else no change.

d) Withdraw money from the wallet

1) Withdraw money from the wallet
2) Patient can transfer money from his wallet to a bank account
3) Human trigger
4) Only a patient is interacting
5) **Inputs->** amount to be transferred **Constraints->** amount >= 1
6) No precondition
7) **User->** enter amount to transfer, bank account info(acc_num, IFSC code etc.) **System->** will check for if the user's balance is sufficient and then will make a transaction from the hospital's account to the user's account. If the transaction succeeded, the system would deduct the amount from the user's wallet balance.
8) Exceptions-> payment fails, amount outside constraints, insufficient balance
9) User will have the amount deducted from his wallet if all goes well, else no change.

e) Verify a payment

1) Verify Payment
2) Patient can verify a payment initiated by pharmacy_keeper to purchase medicines
3) System triggered for sending notification to verify payment, Human triggered for verifying payment
4) Patient and Pharmacy_keeper are primary actor
5) **Inputs->** none **Constraints->** balance must be >= amount in payment initiated
6) **Precondition->** payment must be initiated by pharmacy_keeper
7) **User->** click on verify payment **System->** will deduct money specified by payment from the user's wallet
8) Exceptions-> not enough balance in user's wallet
9) On success amount will be deducted from user's wallet and payment will be made, On failure no change

3) Accountant

a) Add money to patient's wallet
   i) Add money to patient's wallet
   ii) If patient have cash then someone is required to add that cash to patients wallet
   iii) Triggered by Accountant(human trigger)
   iv) Accountant
   v) **Input-**> patient_id, amount **Constraints-**> amount should be greater than 0 and valid patient id
   vi) No precondition
   vii) Accountant will input patient_id, amount and system will add that much amount to patient's wallet and then a message for successful transaction will be shown
   viii) Exception-> when transaction is failed
   ix) If transaction is successful then patients wallet will have more money

b) give back money from wallet
   i) give back money from wallet
   ii) Finally when patient is discharged the remaining amount will be given to patient by accountant
   iii) Triggered by Accountant(human trigger)
   iv) Accountant
   v) **Input-**> patient_id, amount **Constraints-**> amount should be greater than 0 and valid patient id
   vi) No precondition
   vii) Accountant will input patient_id, amount and system will deduct that much amount from patient's wallet and then a message for successful transaction will be shown
   viii) Exception-> when transaction is failed
   ix) If transaction is successful then patients wallet will have less money

4) Pathologist

a) Add report to patient's account
   1) Can add report of patients to their account
   2) Pathologists can generate the reports, so they can add it to the database.
   3) Human trigger

4) Pathologist
5) patient_id, report(link to the report file), date
6) Patient should already exist
7) Interaction-> adding inputs to the form and then submit by clicking
8) No exceptions
9) Patients can now see their reports. Doctors can also access the reports of their patients

b) Access prescription and previous reports of patient
1) Access prescription and previous reports of patient
2) Pathologists can see prescribed tests and medicine and reports of the user generated in the past.
3) Human Trigger
4) Pathologist
5) Patient_id
6) Patient should exist
7) interaction->entering patient_id in the form and clicking, output->all the prescription of the patient of the past
8) No exceptions
9) System will remain in same state

5) Pharmacy keeper :

a. Updating the medicine entity for the new supply
a) Updating the medicine entity for the new supply
b) The user will be updating the medicine in the database according to the real time activity.
c) Triggered by the Pharmacy_keeper
d) Pharmacy keeper
e) **Input->** medicine details(for insert) and quantity **Constraints->** inserted medicine must not exist before
f) No precondition
g) Click on insert new_medicine if the medicine is not available and update the availability
h) Exceptions-> medicine already exist
i) new_supply medicine has been updated to the medicine entity.

b. Checking availability of the medicine before purchasing
1) Checking availability of the medicine before purchasing
2) The user will be updating the medicine in the database according to the real time activity.
3) Triggered by the Pharmacy_keeper
4) Pharmacy keeper
5) **Input->** medicine details(for insert) and quantity **Constraints->** quantity >= 0
6) No precondition
7) Putting all the names of medicine and check (availability - requirement) >= 0
8) Exceptions-> medicine already exists, underflow of availability after update.
9) moving to the next step of payment, new_supply medicine have been updated to the medicine entity.

c. Initiate a payment for medicine
    1) Initiate a payment for medicine
    2) Pharmacy_keeper will initiate a payment for medicines which will be verified by the patient and then payment will complete.
    3) Triggered by this Pharmacy_keeper
    4) Pharmacy keeper and Patient
    5) **Input-**> Total amount of the purchased medicine, patient_id **Constraints**:
    6) Prescription for medicines must exist
    7) **User->** initiate a payment with total amount of medicines to corresponding patient **System->** send notification to patient to verify payment and on successful verification by patient make a transaction and deduct amount from patients wallet and reduce the availability in the medicine entity.
    8) Exceptions-> verification bay patient failed
    9) On success a transaction will be added and amount will be deducted form patients wallet else no change.

6) Director

a) Add a new staff member
    i) Add a new staff member to the Hospital
    ii) Director can add a new person as a staff member of the Hospital
    iii) Human trigger
    iv) Only director is interacting
    v) **Inputs->** All details about new staff member, role, time slot, salary **Constraints->** age must be in range specified by Hospital
    vi) No precondition
    vii) **Director->** give input **System->** verify input and add new staff to hospital system
    viii) Exceptions-> age not in range, missing required field etc.
    ix) After success there will be a new staff member in staff.

b) Remove a staff member
    i) Remove a staff member
    ii) Director can remove a staff member from Hospital
    iii) Human trigger
    iv) Director
    v) **Input->** Id of staff member, **Constraints->** id must exist and staff member must be currently working
    vi) No precondition
    vii) **User->** select staff member to remove, date(optional) **System->** will check constraints and add date_of_leave to current date or date supplied.
    viii) Exceptions-> staff member not working currently
    ix) After success staff member will be updated with date_of_leave

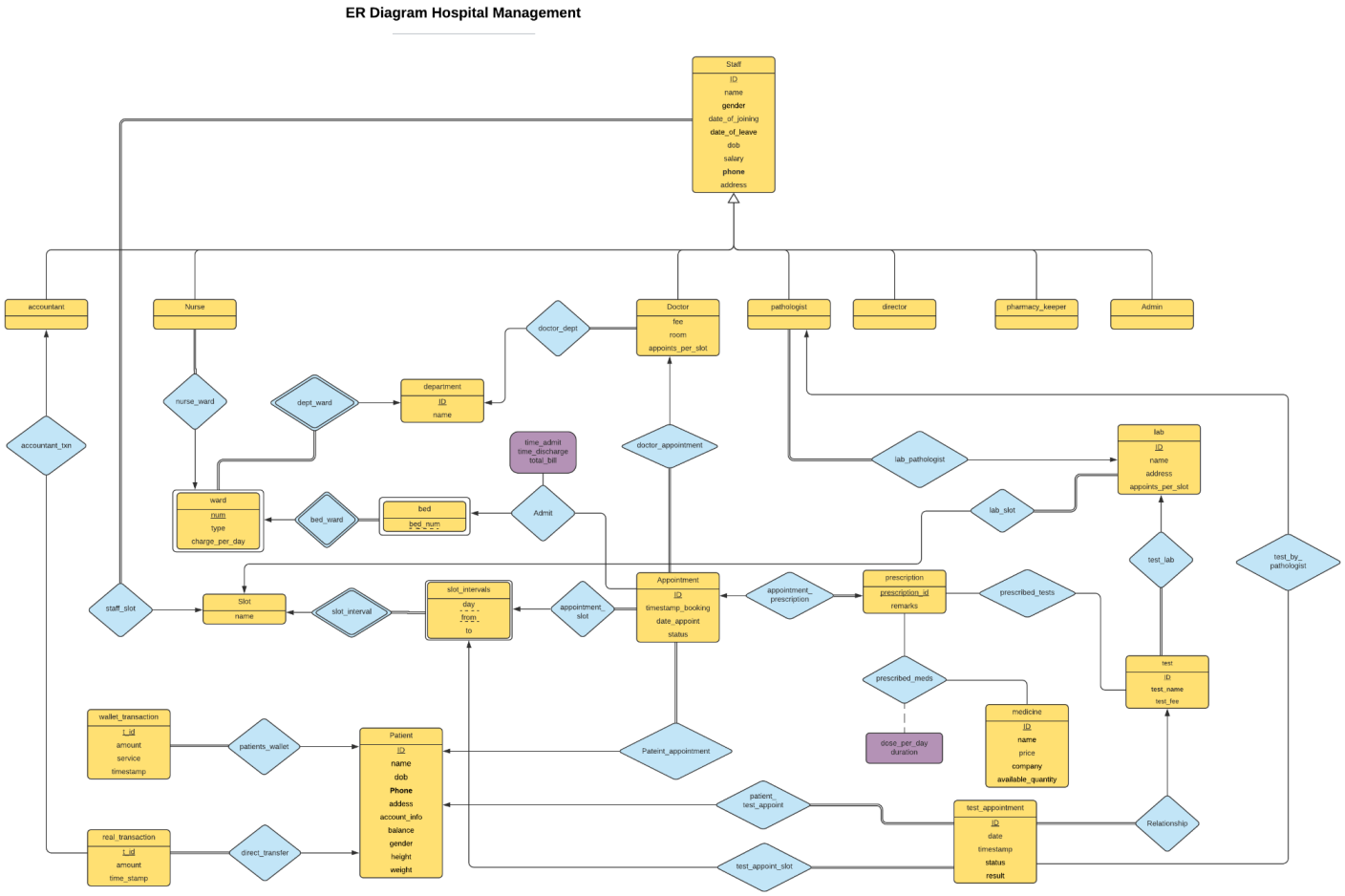
c) Can see statistics and analytics
    i) View Analytics and Statistics
    ii) A user can see analytics and statistics of the system in real time.
    iii) Human triggered for opening analytics page, Details will be updated automatically(automatically triggered)
    iv) Director

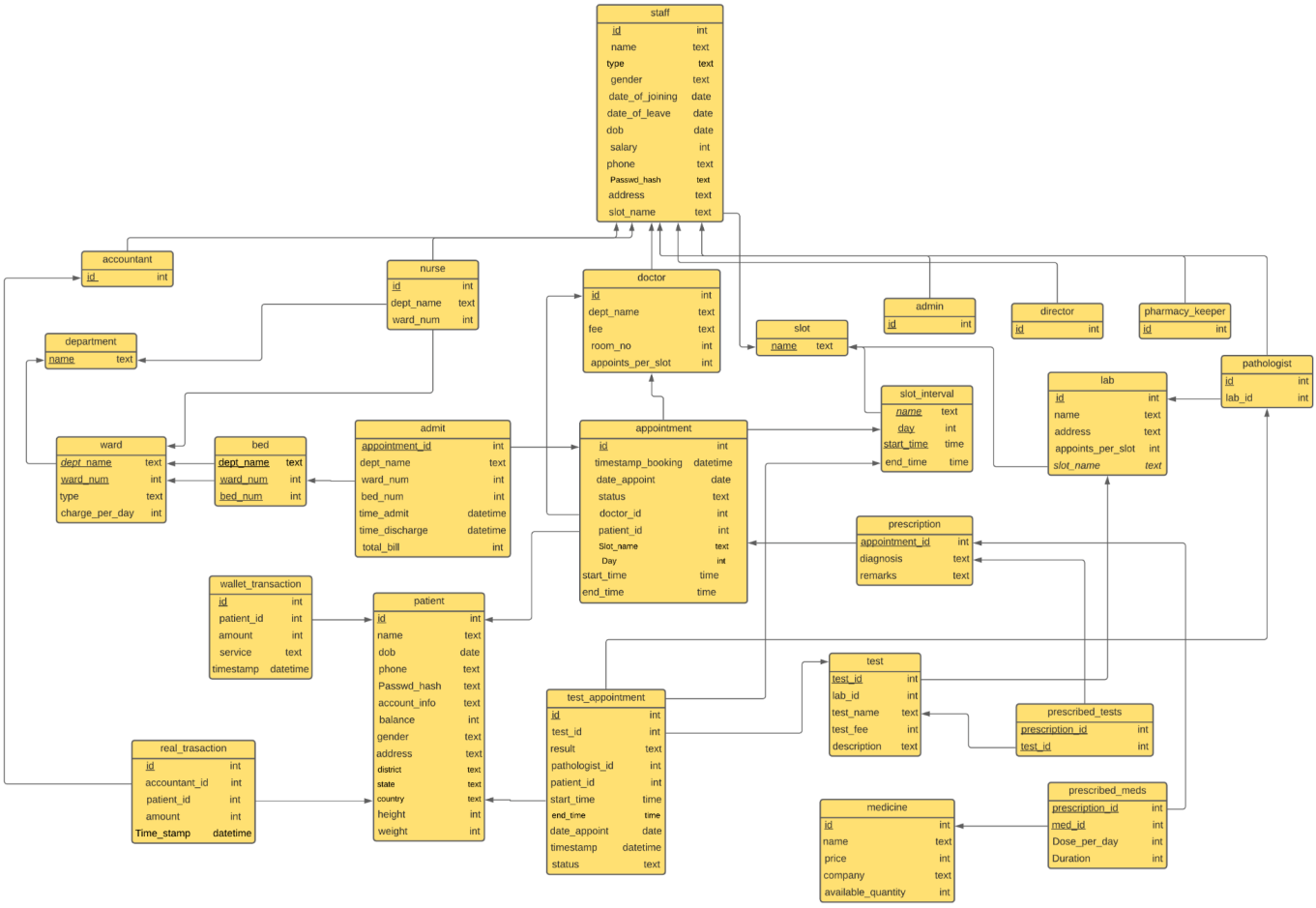
v) No input currently
vi) No preconditions currently
vii) No success path currently
viii) No exception currently
ix) No postconditions currently

7) Admin

a) Can see statistics and analytics
i) View Analytics and Statistics
ii) A user can see analytics and statistics of the system in real time.
iii) Human triggered for opening analytics page, Details will be updated automatically(automatically triggered)
iv) Admin
v) No input currently currently
vi) No preconditions currently
vii) No success path currently
viii) No exception currently
ix) No postconditions currently

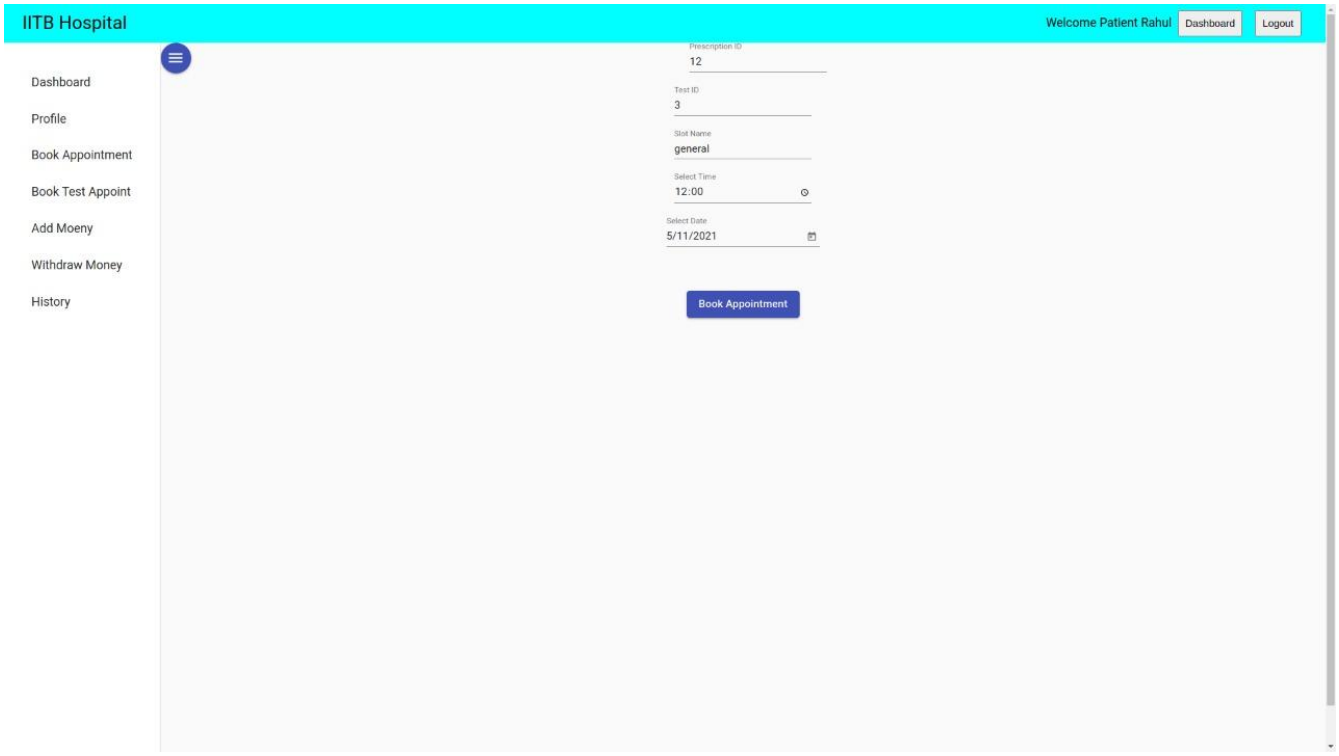
8) Nurse

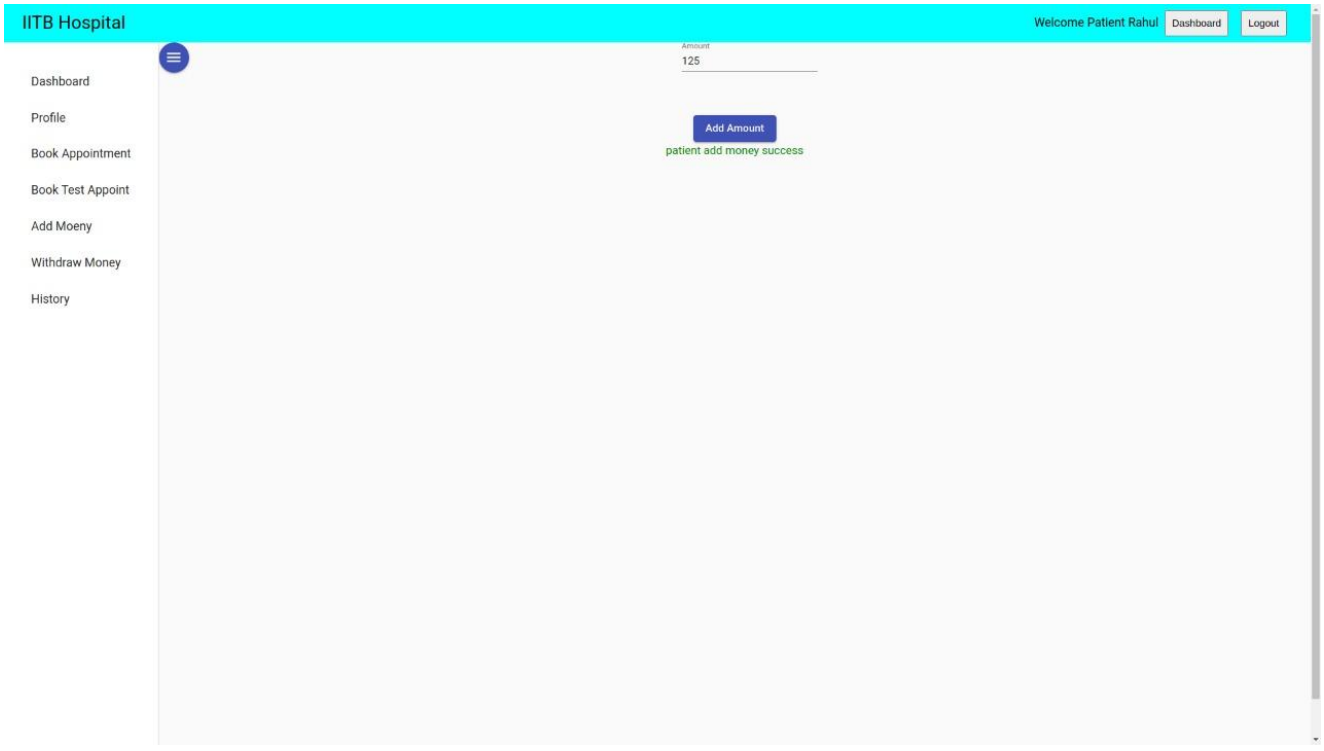


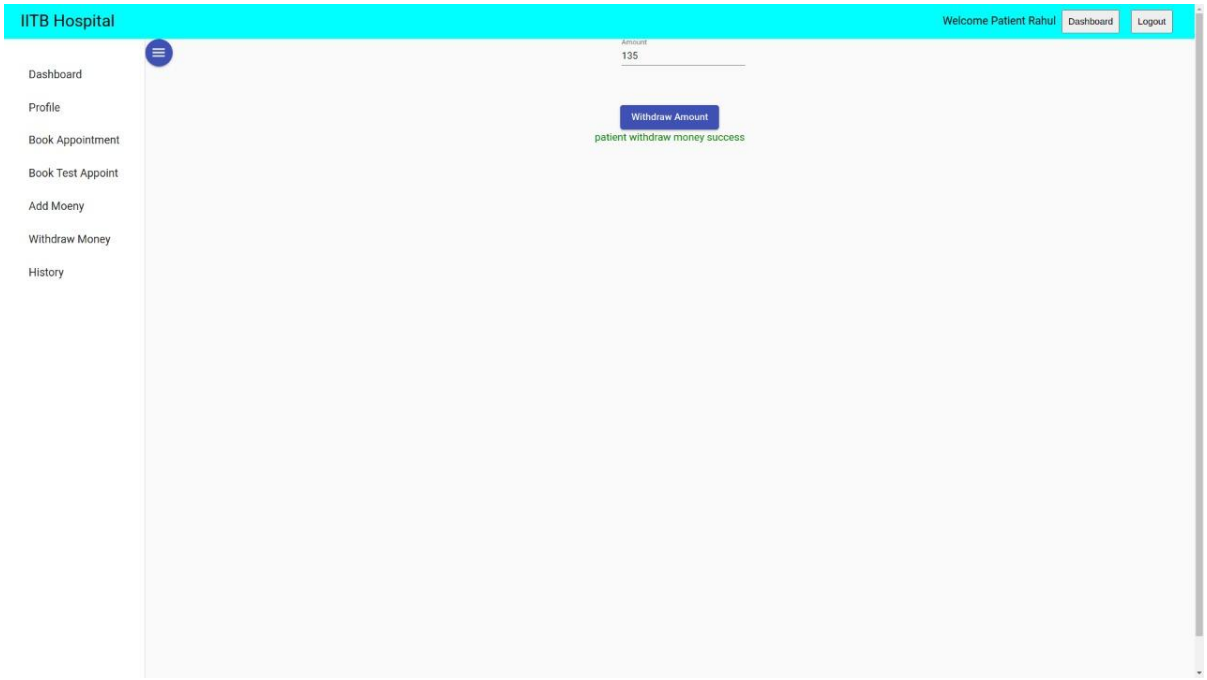


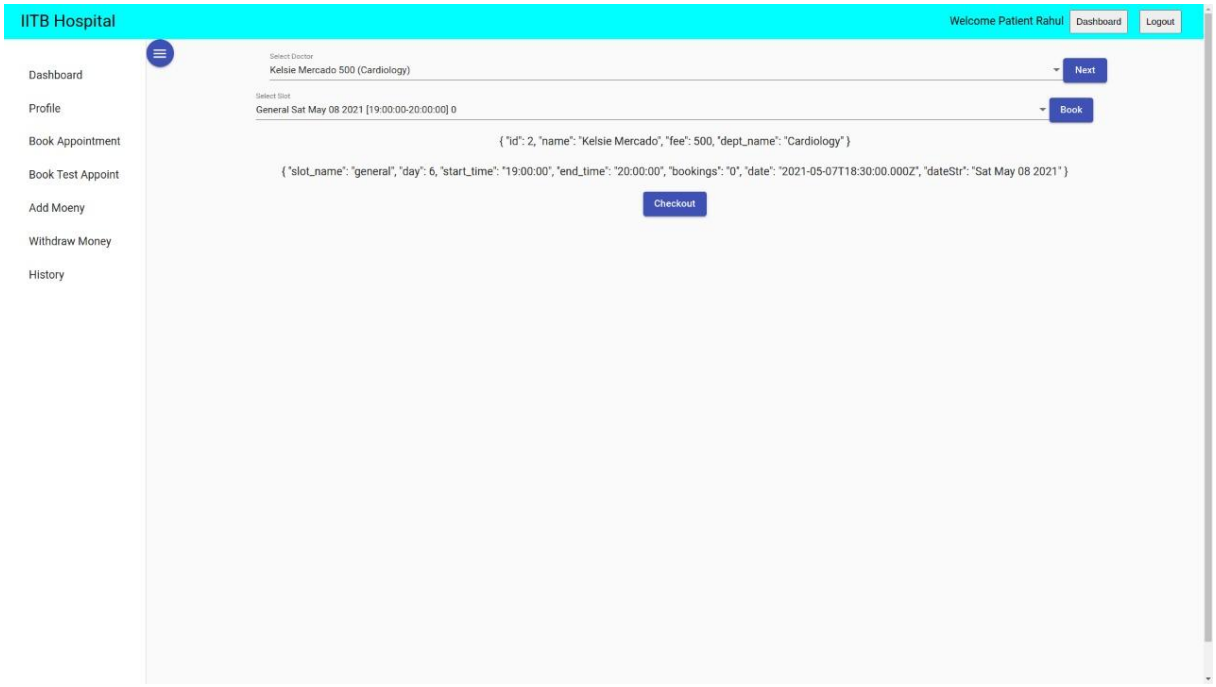
d) Test Results

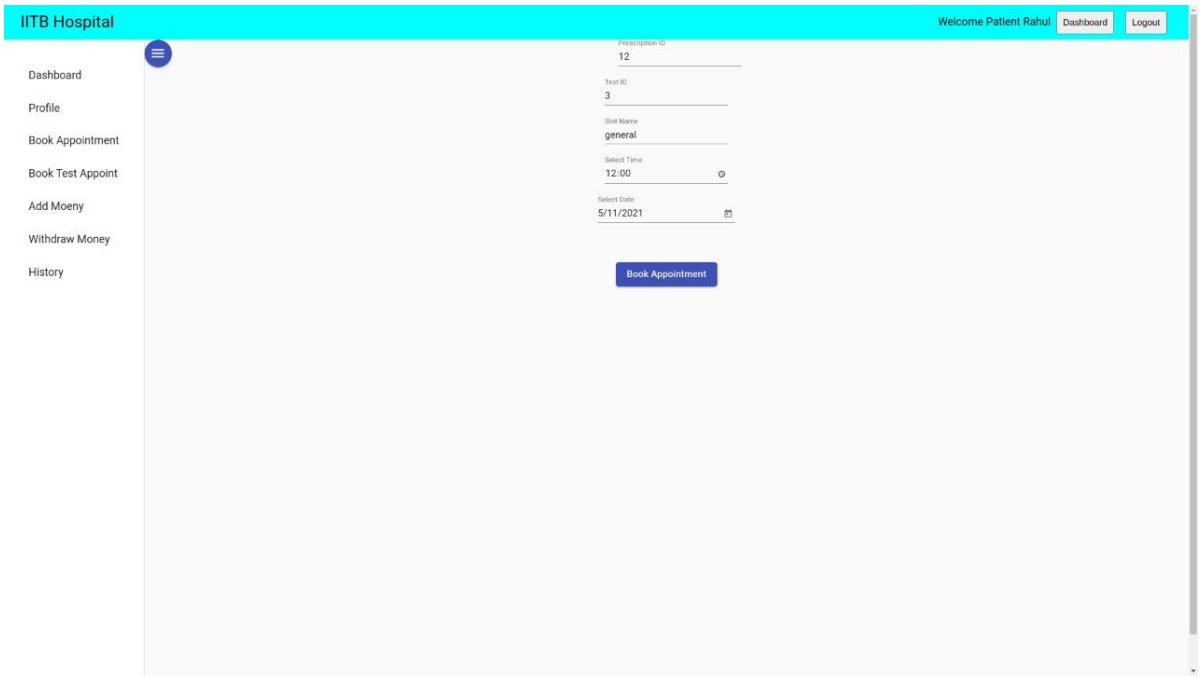
Patient login, book-appointment, add/withdraw money:-


IITB Hospital
Welcome Patient Rahul
Dashboard
Logout
Dashboard
Profile
Book Appointment
Book Test Appoint
Add Moeny
Withdraw Money
History
Prescription ID
12
Test ID
3
Slot Name
general
Select Time
12:00
Select Date
5/11/2021
Book Appointment

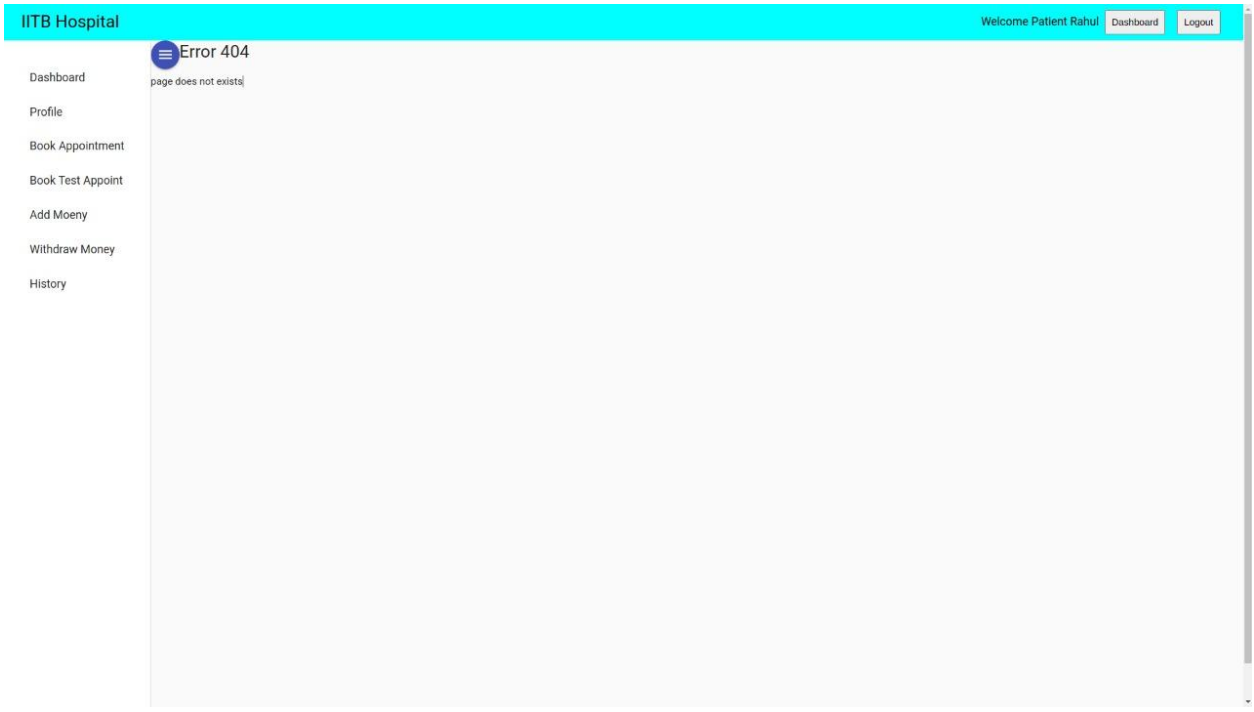

IITB Hospital
Welcome Patient Rahul
Dashboard
Logout
Dashboard
Profile
Book Appointment
Book Test Appoint
Add Moeny
Withdraw Money
History
Amount
125
Add Amount
patient add money success

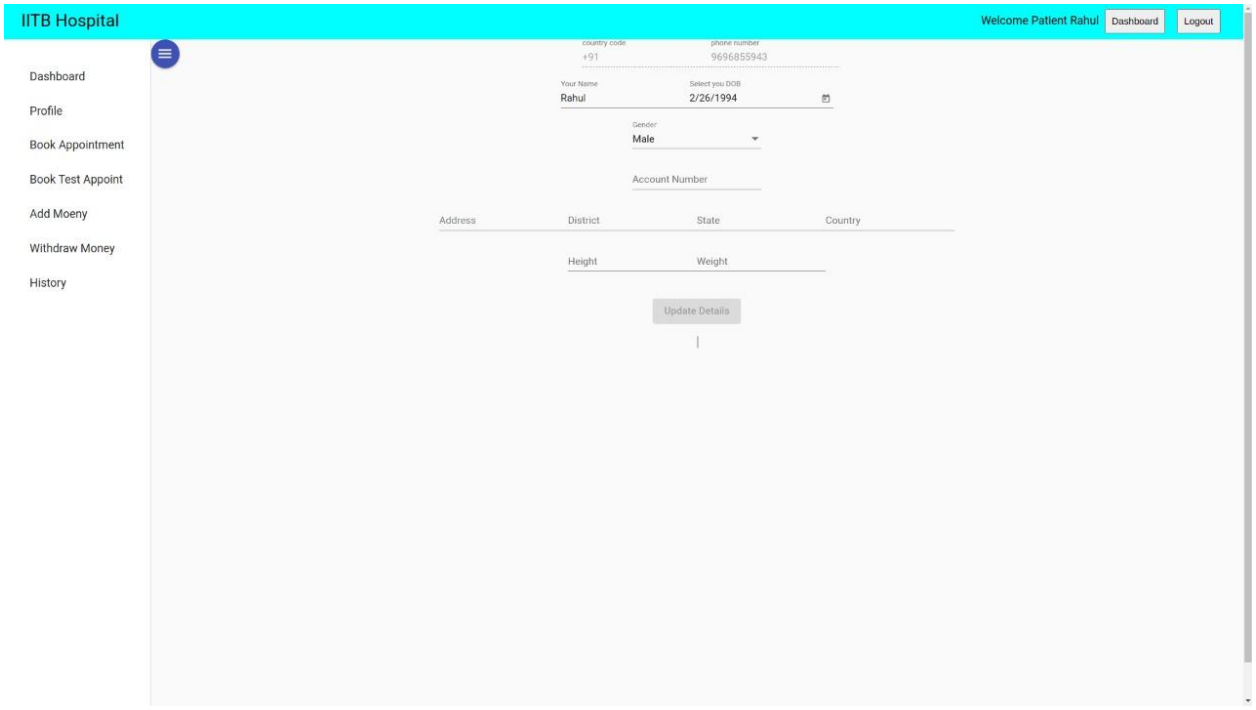

IITB Hospital
Welcome Patient Rahul
Dashboard
Logout
Dashboard
Profile
Book Appointment
Book Test Appoint
Add Moeny
Withdraw Money
History
Amount
135
Withdraw Amount
patient withdraw money success


IITB Hospital
Welcome Patient Rahul
Dashboard
Logout
Dashboard
Profile
Book Appointment
Book Test Appoint
Add Moeny
Withdraw Money
History
Select Doctor
Kelsie Mercado 500 (Cardiology)
Next
Select Slot
General Sat May 08 2021 [19:00:00-20:00:00] 0
Book
{ "id": 2, "name": "Kelsie Mercado", "fee": 500, "dept_name": "Cardiology" }
{ "slot_name": "general", "day": 6, "start_time": "19:00:00", "end_time": "20:00:00", "bookings": "0", "date": "2021-05-07T18:30:00.000Z", "dateStr": "Sat May 08 2021" }
Checkout

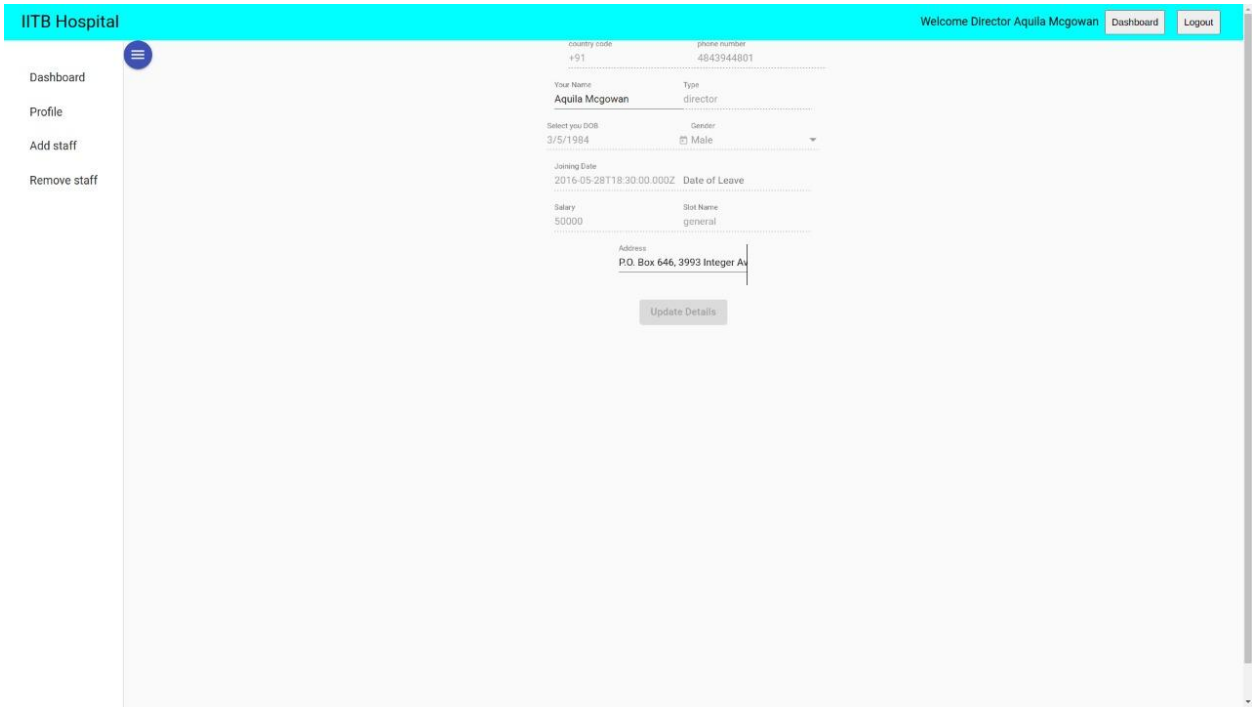

IITB Hospital
Welcome Patient Rahul
Dashboard
Logout
Dashboard
Profile
Book Appointment
Book Test Appoint
Add Moeny
Withdraw Money
History
Prescription ID
12
Test ID
3
Slot Name
general
Select Time
12:00
Select Date
5/11/2021
Book Appointment


IITB Hospital
Login / Register
country code
+91
phone number
9696855943
password
Your Name
Rahul
Select you DOB
2/27/1994
Gender
Male
Register


IITB Hospital
Welcome Patient Rahul
Dashboard
Logout
Dashboard
Profile
Book Appointment
Book Test Appoint
Add Moeny
Withdraw Money
History
Error 404
page does not exists

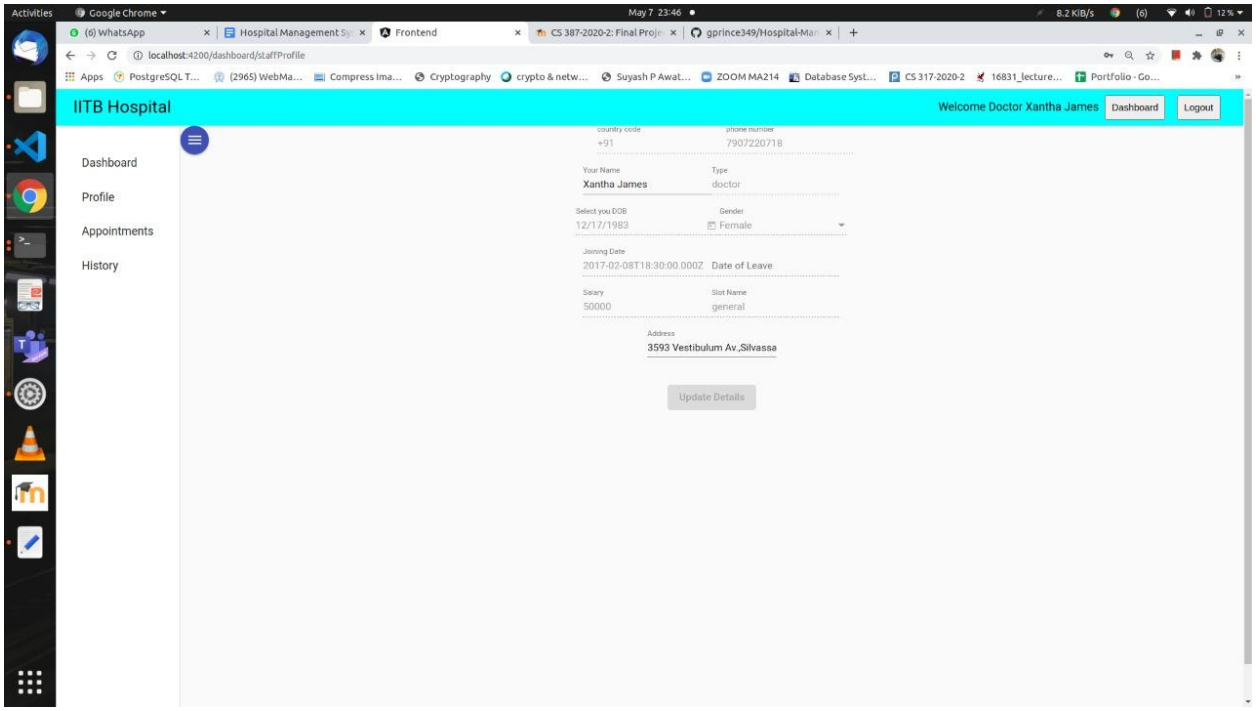

IITB Hospital
Welcome Patient Rahul
Dashboard
Logout
Dashboard
Profile
Book Appointment
Book Test Appoint
Add Moeny
Withdraw Money
History
country code
+91
phone number
9696855943
Your Name
Rahul
Select you DOB
2/26/1994
Gender
Male
Account Number
Address
District
State
Country
Height
Weight
Update Details

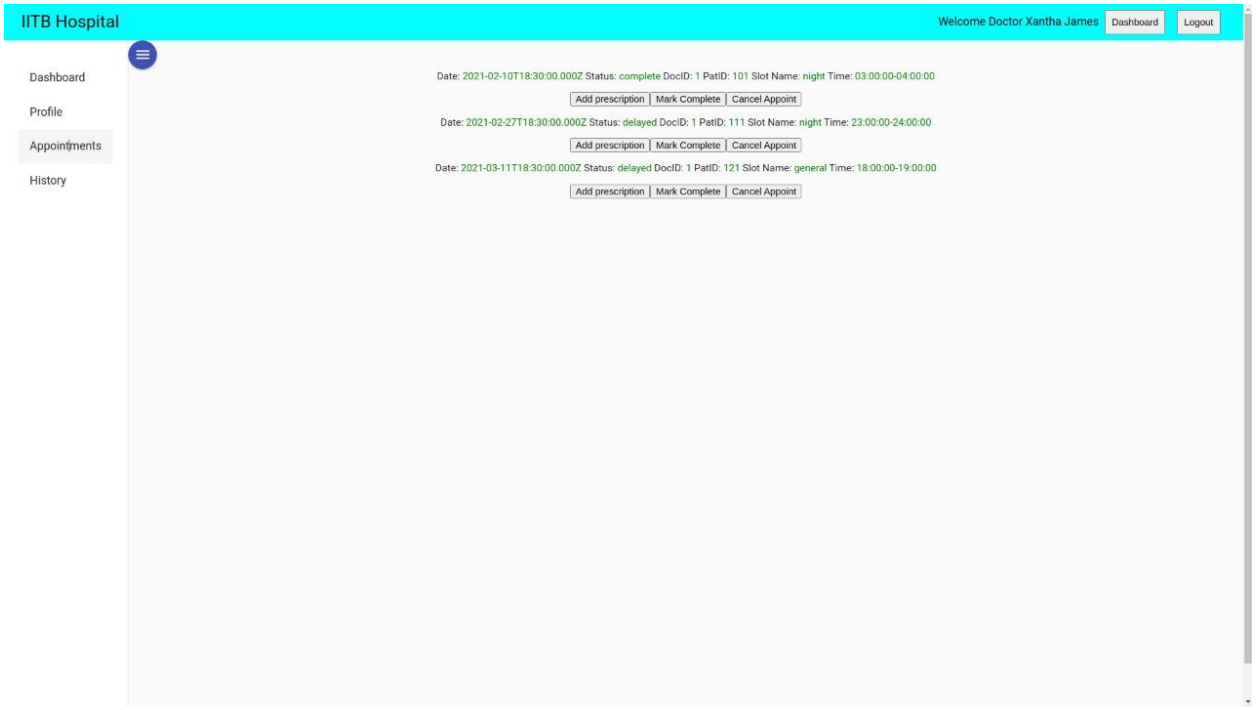

IITB Hospital
Login / Register
Not a Staff? login as a Patient
Login as a "Staff"
country code
+91
phone number
4843944801
password
Submit
Not a user? register


IITB Hospital
Welcome Director Aquila Mcgowan
Dashboard
Logout
Dashboard
Profile
Add staff
Remove staff
country code
+91
phone number
4843944801
Your Name
Aquila Mcgowan
Type
director
Select you DOB
3/5/1984
Gender
Male
Joining Date
2016-05-28T18:30:00.000Z
Date of Leave
Salary
50000
Slot Name
general
Address
P.O. Box 646, 3993 Integer Av
Update Details


IITB Hospital
Login / Register
Not a Staff? login as a Patient
Login as a "Staff"
country code
+91
phone number
4843944801
password
Submit
Not a user? register

Doctor profile , appointments





e) Conclusion

We are able to complete the following:

1. Login and registration
2. Change and update of details by user
3. Booking an appointment by patient
4. Director can add staff members
5. Patient can add and withdraw money
6. Patient can see his details on dashboard
7. Doctor can see list of his appointments

Due to health issues of our team members we weren't able complete the following requirements on time:

1. Front-end interface for the use-cases of Pharmacy Keeper and Pathologist.
2. An interface for the statistics/analytics of the hospital database system.
3. Booking of test appointment
4. Update and modification of appointments by doctor

f) git Link

https://github.com/gprince349/Hospital-Management-System.git

c) Database Design:
DDL of the Schema:

```
DROP TRIGGER IF EXISTS check_delayed on appointment;
DROP TRIGGER IF EXISTS check_test_delayed on test_appointment;

DROP TABLE IF EXISTS    test_appointment;
DROP TABLE IF EXISTS    prescribed_meds;
DROP TABLE IF EXISTS    prescribed_tests;
DROP TABLE IF EXISTS    medicine;
DROP TABLE IF EXISTS    test;
DROP TABLE IF EXISTS    prescription;
DROP TABLE IF EXISTS    admit;
DROP TABLE IF EXISTS    real_transaction;
DROP TABLE IF EXISTS    wallet_transaction;
DROP TABLE IF EXISTS    appointment;
DROP TABLE IF EXISTS    bed;
```

```
DROP TABLE IF EXISTS    doctor;
DROP TABLE IF EXISTS    admin;
DROP TABLE IF EXISTS    director;
DROP TABLE IF EXISTS    pharmacy_keeper;
DROP TABLE IF EXISTS    pathologist;
DROP TABLE IF EXISTS    accountant;
DROP TABLE IF EXISTS    nurse;
DROP TABLE IF EXISTS    staff;
DROP TABLE IF EXISTS    patient;
DROP TABLE IF EXISTS    lab;
DROP TABLE IF EXISTS    ward;
DROP TABLE IF EXISTS    department;
DROP TABLE IF EXISTS    slot_interval;
DROP TABLE IF EXISTS    slot;

-- ================================================
--               Trigger to set id_serial
--              (on manual input value for id) as max()
-- ================================================

CREATE OR REPLACE FUNCTION set_serial_id_seq()
RETURNS trigger AS
$BODY$
 BEGIN
  EXECUTE (FORMAT('SELECT setval(''%s_%s_seq'', (SELECT MAX(%s) from %s));',
  TG_TABLE_NAME,
  TG_ARGV[0],
  TG_ARGV[0],
  TG_TABLE_NAME));
   RETURN OLD;
  END;
$BODY$
LANGUAGE plpgsql;
-- ================================================


-- 1
CREATE TABLE slot (
       name                                    text,
       Primary key (name)
);

-- 2
CREATE TABLE slot_interval (
       name                          text,
       day                               integer check(day >=0 and day < 7),
       start_time                        time,
       end_time                          time   Not null,
       Primary key(name, day, start_time),
       Foreign key(name) references slot(name) on delete Cascade
);
-- 3
CREATE TABLE lab(
       id                                      serial,
       name                              text    Not null,
       address                           text    Not null,
```

```
        appoints_per_slot               integer Not Null
check(appoints_per_slot >= 0 ),
        slot_name                       text    Not null,
        Primary key(id),
        Foreign key (slot_name) references slot
);
-- ================================================
CREATE TRIGGER set_lab_id_seq
AFTER INSERT OR UPDATE OR DELETE
ON lab
FOR EACH STATEMENT
 EXECUTE PROCEDURE  set_serial_id_seq('id');
-- ================================================

-- 4
CREATE TABLE department (
        name                        text,
        Primary key (name)
);

-- 5
CREATE TABLE ward (
        dept_name               text,
        ward_num                    integer,
        type                        text        Not null        check (type in
('General', 'ICU')),
        charge_per_day          integer Not null        check(charge_per_day >=
0 ),
        Primary key (dept_name,ward_num),
        Foreign key(dept_name) references department(name)
);
-- 6
CREATE TABLE patient (
        id                              serial,
        name                        text,
        dob                         date        Not Null,
        phone                       text        Not Null        Unique,
        passwd_hash                             text    Not null,
                account_info            text,
        balance                     integer     Not null        check (balance
>= 0 ) Default 0,
        gender                  text    Not null        Check (gender
in('male','female','other')),
                address                     text,
                district                                text,
                state                                   text,
                country                                 text,
        height                      integer     check (height > 0 or height is
null ) ,
        weight                      integer     check (weight > 0 or weight is
null),


        Primary Key(id)
);
-- ================================================
CREATE TRIGGER set_patient_id_seq
```

```
AFTER INSERT OR UPDATE OR DELETE
ON patient
FOR EACH STATEMENT
 EXECUTE PROCEDURE  set_serial_id_seq('id');
-- ================================================

-- 7
CREATE TABLE staff (
       id                      serial,
       name                    text    NOT NULL,
              type                                    text    Not Null,
       gender                  text    Not null Check (gender
in('male','female','other')),
       date_of_joining                 date    NOT NULL,
       date_of_leave                   date,   /*(null allowed => currently
working)*/
       dob                             date    Not null,
       salary                          integer Not null        check (salary
>= 0),

       phone                           text    NOT NULL UNIQUE,
              passwd_hash                             text    Not null,
       address                         text,
       slot_name                       text,
       Primary Key(id),
       Foreign Key(slot_name) references slot(name)
);
-- ================================================
CREATE TRIGGER set_staff_id_seq
AFTER INSERT OR UPDATE OR DELETE
ON staff
FOR EACH STATEMENT
 EXECUTE PROCEDURE  set_serial_id_seq('id');
-- ================================================

-- 8
CREATE TABLE accountant (
       id                              integer,
       Primary key(id),
       Foreign Key(id) references staff(id) on delete Cascade
);

-- 9
CREATE TABLE nurse (
       id                              integer,
       dept_name                       text,
       ward_num                        integer,
       Primary key(id),
       Foreign Key(id) references staff(id) on delete Cascade,
       Foreign Key(dept_name, ward_num) references ward(dept_name,ward_num)
);

-- 10
CREATE TABLE pathologist (
       id                                  integer,
       lab_id                          integer,
       Primary key(id),
```

```
	Foreign Key(id) references staff on delete Cascade,
	Foreign Key(lab_id) references lab(id)
);

-- 11
CREATE TABLE pharmacy_keeper (
	id 					integer,
	Primary key(id),
	Foreign Key(id) references staff(id) on delete Cascade
);
-- 12
CREATE TABLE director (
	id 					integer,
	Primary key(id),
	Foreign Key(id) references staff(id) on delete Cascade
);
-- 13
CREATE TABLE admin (
	id 					integer,
	Primary key(id),
	Foreign Key(id) references staff on delete Cascade
);
-- 14
CREATE TABLE  doctor(
	id 				integer,
	dept_name 			text,
	fee 				integer Not null 	check (fee >=
0),
	room_no 		integer ,
	appoints_per_slot 		integer Not null 	check(appoints_per_slot
>=0 ),
	Primary key(id),
	Foreign key(id) references staff on delete Cascade,
	Foreign key(dept_name) references department(name)
);


-- 15
CREATE TABLE bed (
	dept_name 			text,
	ward_num 		integer,
	bed_num 		integer,
	Primary key (dept_name,ward_num,bed_num),
	Foreign Key(dept_name, ward_num) references ward(dept_name, ward_num)
on delete Cascade
);


-- 16
CREATE TABLE appointment(
	id 					serial,
	timestamp_ 				timestamp without time zone
Default current_timestamp,
	date_appoint 		date 	Not Null,
	status 			text 	Not Null check(status in
('scheduled', 'complete', 'delayed', 'cancelled by doctor', 'cancelled')),
	doctor_id 			integer 	Not null,
```

```
       patient_id                  integer     Not null,
       slot_name                   text        Not null,
       slot_day                    integer     Not null,
       start_time                      time without time zone  Not null,
       end_time                        time without time zone,

       Primary key(id),
       Foreign Key(doctor_id) references doctor,
       Foreign Key(patient_id) references patient,
       Foreign Key(slot_name, slot_day, start_time) references
slot_interval(name,day,start_time) on delete set Null
);
-- ================================================
CREATE TRIGGER set_appointment_id_seq
AFTER INSERT OR UPDATE OR DELETE
ON appointment
FOR EACH STATEMENT
 EXECUTE PROCEDURE  set_serial_id_seq('id');
-- ================================================

-- 17
CREATE TABLE wallet_transaction (
       id                              serial,
       patient_id                  integer,
       amount                      integer    NOT NULL,
       service                     text               Not Null,
       timestamp_                  timestamp without time zone        Default
current_timestamp,
       Primary Key(id),
       Foreign Key(patient_id) references patient on delete set Null
);
-- ================================================
CREATE TRIGGER set_wallet_transaction_id_seq
AFTER INSERT OR UPDATE OR DELETE
ON wallet_transaction
FOR EACH STATEMENT
 EXECUTE PROCEDURE  set_serial_id_seq('id');
-- ================================================

-- 18
CREATE TABLE real_transaction (
       id                              serial,
       accountant_id            integer,
       patient_id               integer,
       amount                      integer    Not Null,
       timestamp_                  timestamp without time zone        Default
current_timestamp,
       Primary key(id),
       Foreign Key(patient_id) references patient on delete set Null,
       Foreign Key(accountant_id) references accountant(id) on delete set Null
);
-- ================================================
CREATE TRIGGER set_real_transaction_id_seq
AFTER INSERT OR UPDATE OR DELETE
ON real_transaction
FOR EACH STATEMENT
 EXECUTE PROCEDURE  set_serial_id_seq('id');
```

```
-- ================================================

-- 19
CREATE TABLE admit (
       appointment_id          integer,
       dept_name               text,
       ward_num                integer,
       bed_num                 integer,
       time_admit              timestamp without time zone            Default
current_timestamp,
       time_discharge          timestamp without time zone,
       total_bill              integer         check(total_bill>=0),
       Primary key(appointment_id),
       Foreign Key(appointment_id) references appointment on delete set NULL,
       Foreign Key(dept_name,ward_num, bed_num) references bed on delete set
Null,
              Check (time_discharge is not null OR (dept_name is not null AND
ward_num is not null AND bed_num is not null))
              -- make sure that if patient is not discharged then bed should
exist
);

-- 20
CREATE TABLE prescription(
       appointment_id              integer,
       diagnosis                           text    Not null,
       remarks                     text,
       Primary key(appointment_id),
       Foreign key(appointment_id) references appointment on delete Cascade
);


-- 21
CREATE TABLE test (
       test_id                     integer,
       lab_id                          integer         Not Null,
       test_name                   text                Not null,
       test_fee                    integer     Not null        check (test_fee
>= 0 ),
              description                              text,
       Primary key(test_id),
       Foreign Key(lab_id) references lab
);

-- 22
CREATE TABLE medicine (
       id                              serial,
       name                        text                Not null,
       price                       integer     Not null        check(price >=0
),
       company                 text            Not null,
       available_quantity          integer             Not null
check(available_quantity >= 0 ),
       Primary key(id)
);
-- ================================================
CREATE TRIGGER set_medicine_id_seq
```

```
AFTER INSERT OR UPDATE OR DELETE
ON medicine
FOR EACH STATEMENT
 EXECUTE PROCEDURE  set_serial_id_seq('id');
-- ================================================

-- 23
CREATE TABLE prescribed_tests (
       prescription_id             integer,
       test_id                         integer,
       Primary key(prescription_id,test_id),
       Foreign Key(prescription_id) references prescription on delete Cascade,
       Foreign Key(test_id) references Test on delete set Null
);

-- 24
CREATE TABLE prescribed_meds (
       prescription_id             integer,
       med_id                          integer,
       dose_per_day            integer         check(dose_per_day >= 0 OR
dose_per_day is null),
       duration                        integer         check(duration >= 0 OR
duration is null),
       Primary key(prescription_id,med_id),
       Foreign Key(prescription_id) references prescription on delete Cascade,
       Foreign Key(med_id) references medicine on delete set Null
);

-- 25
CREATE TABLE test_appointment (
       id                              serial,
       test_id                     integer             Not null,
       timestamp_                  timestamp without time zone Default
current_timestamp,
       pathologist_id          integer,
       patient_id              integer         Not null,
       slot_name                   text                Not null,
       slot_day                            integer         Not null,
       start_time                  time without time zone,
       end_time                    time without time zone,
       date_appoint                date                Not null,
       status                      text        Not null check(status in
('scheduled', 'sample_taken', 'complete', 'delayed', 'cancelled', 'cancelled
by pathologist')),
       result                  text,                   /*null while result not
published*/

       Primary key(id),
       Foreign Key(test_id) references Test,
       Foreign Key(patient_id) references patient,
       Foreign Key(pathologist_id) references pathologist,
       Foreign Key(slot_name,slot_day,start_time) references
slot_interval(name,day,start_time) on delete set Null
);
-- ================================================
CREATE TRIGGER set_test_appointment_id_seq
AFTER INSERT OR UPDATE OR DELETE
```

```
ON test_appointment
FOR EACH STATEMENT
 EXECUTE PROCEDURE  set_serial_id_seq('id');
-- ================================================



   update test_appointment set status = 'delayed'
   where status = 'scheduled'
   and (date_appoint < current_date) OR (date_appoint = current_date and
end_time < current_time);
   return new;
END;
$$;

create trigger check_test_delayed AFTER insert OR update OR delete
on test_appointment
for each statement
when (pg_trigger_depth() = 0)
execute procedure check_test_appoint_delayed();



-- pay and confirm slot booking GIVEN(date, day, slot_name, start_time,
end_time, patient_id, fee)
-- creating a function to do booking
create or replace function book_appoint(max_appoints int, cur_appoints int,
       appoint_date date, start_time time, end_time time, p_id int, d_id int,
slot_name text, day int, fee int)
returns int
language plpgsql
as
$$
declare
  a integer;
begin
   select 0 into a;
   if (cur_appoints >= max_appoints) then
       raise exception 'slot is full';

   elseif ((select balance from patient where id = p_id) < fee) then
       raise exception 'insufficient balance in your wallet, recharge it';
   else
       insert into appointment(date_appoint, status, doctor_id, patient_id,
slot_name, slot_day, start_time, end_time)
       values (appoint_date, 'schedualed', d_id, p_id, slot_name, day,
appoint_time, end_time);

       update patient set balance = balance - fee where id = p_id;

       insert into wallet_transaction (patient_id, amount, service)
       values (p_id, -fee, 'appointment booking');

       select 1 into a;
   end if;
   return a;
end;
```

```
$$;
```

Indexes:

```
-- ================== Indexes ======================
Create index doct_appointment           on appointment USING btree
(doctor_id);
Create index patient_appointment        on appointment USING btree
(patient_id);
Create index date_appointment           on appointment USING btree
(date_appoint);
Create index st_appointment             on appointment USING btree (status);

Create index patho_test_appointment on test_appointment USING btree
(pathologist_id);
Create index patient_test_appointment on test_appointment USING btree
(patient_id);
Create index date_test_appointment      on test_appointment USING btree
(date_appoint);
Create index st_test_appointment          on test_appointment USING btree
(status);
```

Triggers:

```
-- ================== Triggers ======================
-- trigger to check for any delayed appointment
-- before any INSERT/UPDATE/DELETE
CREATE OR replace FUNCTION check_appoint_delayed()
  RETURNS TRIGGER
  LANGUAGE PLPGSQL
AS $$
BEGIN
  -- trigger logic
   update appointment set status = 'delayed'
   where status = 'scheduled'
   and (date_appoint < current_date) OR (date_appoint = current_date and
end_time < current_time);
   return new;
END;
$$;

create trigger check_delayed AFTER insert OR update OR delete
on appointment
for each statement
when (pg_trigger_depth() = 0)
execute procedure check_appoint_delayed();


-- trigger to check for any delayed test_appointment
-- before any INSERT/UPDATE/DELETE
CREATE OR replace FUNCTION check_test_appoint_delayed()
  RETURNS TRIGGER
  LANGUAGE PLPGSQL
AS $$
BEGIN
  -- trigger logic
```