Project Analysis

Saumya Goyal (180050092)
saumyagoyal@cse.iitb.ac.in

Akkapaka Saikiran (180050005)
psycherun@cse.iitb.ac.in

Abhinav Kumar (180050003)
abhinavkumar@cse.iitb.ac.in

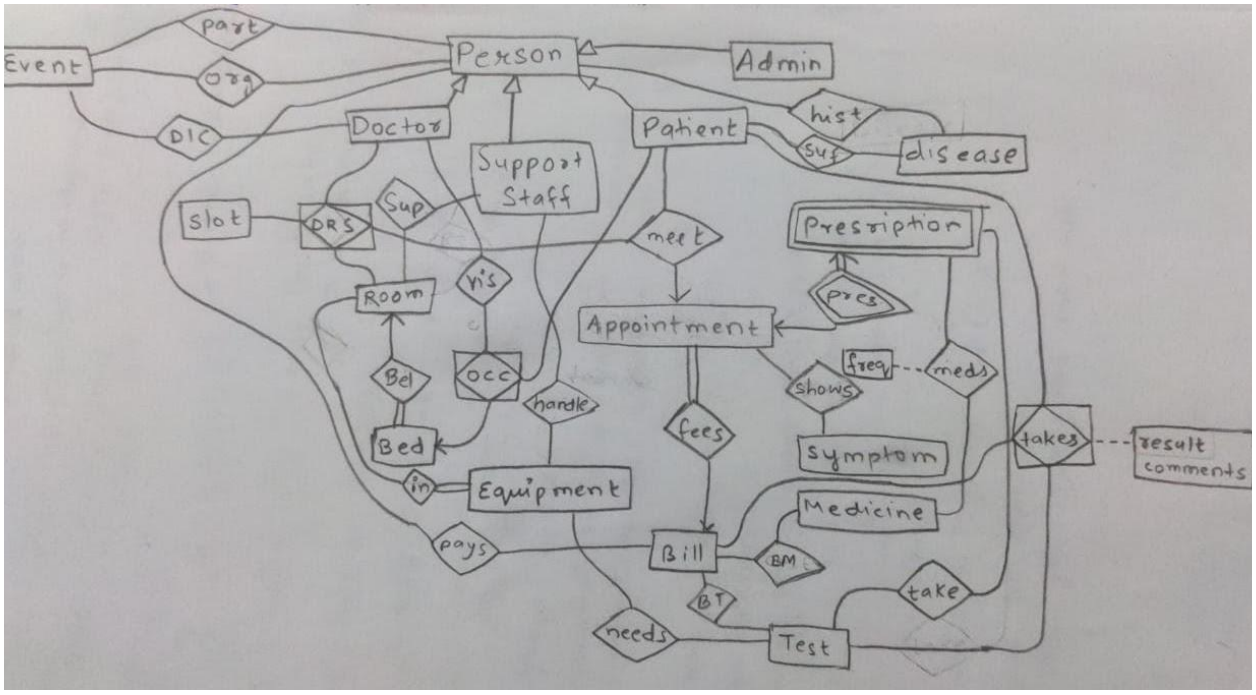
Rishi Agarwal (180050086)
rishiagarwal@cse.iitb.ac.in

Hospital management system (patient-centric)

User Requirements:

An external visitor would like to check the generic information about the hospital such as number of beds available with their types (icu, normal etc), doctors with their speciality, different departments in the hospital, bed charges.
A registered person would like to visit the doctor for either continuation of a previous visit or may visit for checkup for different ailment, pay bills and buy medicines.
A user registering as patient would like to book an appointment for visiting doctor in a slot,, take tests like blood test, ct scan etc, view his/her expenditures for past appointments, disease history, add/edit previous history from a different hospital
The doctor will be able to check their appointments and select/edit slots for a week or a specific day. For a patient visiting the doctor for an appointment, the doctor should be able to see the patient's history (enabling a more thorough diagnosis) and add the prescription and diagnosis of the patient to the database. For a patient who has to be admitted the supervising doctor would select the room and bed type for the patient. He/she will have to visit the admitted patients hence a list of such patients is also required.
Support staff should be able to add test results for a patient's report, attend and generate pharmacy bills. Other than this, bills like that for appointments, etc should be auto-generated when required.
Admin should have many powers ranging from adding/ removing support staffs and doctors, changing their salaries, updating the charges for tests, register new awareness events and select organisers, doctor in-charges, assign duties to support staff such as attending to a bed/room or an equipment, make other user an admin. Allot rooms to doctors for opd/ OTs, add inventories (new equipments/ rooms/ beds etc)

Entity-Relationship diagram:



Person
ID
Name
Address
PIN Code
{Ph no}
gender
age
username
password
email id
Admin
salary

Doctor
speciality
qualification
salary
Permanant
experience
opd charges
OT charges

Support Staff
type
qualification
experience
salary
start_hr
end_hr
days of week

Admin
salary
qualification
role

Patient
patient id

__/__/____

Slot
date
start_time

Room
ID
type

Prescription
ID
instructions

Appointment
ID
type

Symptom
ID
name

Disease
ID
name
info link

Test
ID
name
charges

Bed
no.
type
charges

Equipment
ID
typee

Medicine
ID
name
desc.
manufacturer
price
expiry date
mfg date

Event
ID
name
start_date_time
end_date_time
desc.
link

Bill
ID
paid
purpose

Participate: Person & Event
Person_id, Event_id

Organised: Person & Event
Person_id, Event_id

Doctor-in-charge: Doctor & Event
Doctor_id, Event_id

??Doctor-Room-Slot:
Doctor_id, Room_no, Slot_id

Handle: Support_Staff & Equipment
Support_staff_id, Equipment_id

Meet: Appointment & Patient & Doctor-Room-Slot
Appointment_id, Patient_id, Doctor_id-Room_id-Slot_id, Patient-complaint

History: Person & Disease
Person_id, Disease_id, Previous_history_link, duration

Suffers: Patient & Disease
Patient_id, Disease_id

Prescribes: Appointment & Prescription
Appointment_id, Prescription_id

Shows: Appointment & Symptom
Appointmen_id, Symptom_id

Takes: Patient & Test
Patient_id, Test_id, Result file, Comments, date

Should_Take: Prescription & Test
Prescription_id, Test_id

Bill-Medicine:
Bill_no, Medicine_id, qty

Pays: Person & Bill ??
Person_id, Bill_no, mode of payment

Bed-belongsTo-Room: Bed & Room
Bed_id, Room_id

Facility: Room & Equipment
Room_no, Equipment_id

Occupies: Patient & Bed
Patient_id, Bed_id, Occupied_duration

Meds: Medicine & Prescription
Medicine_id, Prescription_id, Dosage, Frequency

Needs: Test & Equipment
Test_id, Equipment_id

Takes-Bills:
Bill_no, taken_no

Visits: Occupies & Doctor
Patient_id, Bed_id, Doctor_id, time/date of visit, visit notes

Support-staff && room
Support_staff_id, room_id

Add discount % to bill

Use Cases:

1. ***Give admin privileges:***
    1. Title of Use case: “Give admin privileges”
    2. Description: This should allow a user with admin privileges to give some other registered user admin privileges.

3. Trigger event for the use case - Human triggered
4. Primary actor - Admin
5. Inputs to be supplied by primary actor and constraints on input: They have to select the person to whom they want to give the privileges. Constraints are that they can not enter a person who hasn't authorized the admin to view their info.
6. Preconditions for the use case: The person to be given admin privileges should already be a registered user and should have authorized admin to see their basic information
7. Main success path: admin chooses the person_id, person becomes an admin
8. Exceptions that may arise and how they will be compensated: The person has not authorized the admin to view basic info. Person should log in and then authorize admin to avoid this
9. Post conditions: An entry will be created for the person in the admin table, thereby granting them admin privileges

2. **Assign responsibilities to support staff:**
    1. Title of Use case: "Assign responsibilities to support staff"
    2. Description: This allows the admin to assign or unassign various responsibilities to the support staff. Responsibilities that the support staff can take up are to attend to a room (assist doctors and patients in that room for example) and operate equipment (administer tests to patients)
    3. Trigger event for the use case: Human triggered
    4. Primary actor: Admin
    5. Inputs to be supplied by primary actor and constraints on input: The admin should first select the support staff (from those already registered as support staff) and then select the responsibility(/ies) that should be assigned to or unassigned from the staff.
    6. Preconditions for the use case: Responsibilities can only be given to users who are already registered as support staff
    7. Main success path: Admin selects the required fields, and the system adds or removes the new responsibilities for the staff
    8. Exceptions that may arise and how they will be compensated: No exceptions are expected to arise since the admin always chooses from a set of possible input values
    9. Post conditions: corresponding entries will be added to or removed from the "supervise" and "handles" relations

3. ***View/modify personal info***
    1. Title of Use case: "View/modify personal info"
    2. Description: This use case is meant to be something like a dashboard for every registered user. They should be able to see information like the appointments they

have booked/attended and the prescriptions issued, the bills issued in their name and their payment status, the tests they have taken and their results, the events they have registered for and attended, overview of times they were admitted at the hospital and disease history in addition to information such as name, etc. User should also be allowed to edit their personal information such as phone number and update history.

3. Trigger event for the use case: Human triggered
4. Primary actor: Any user
5. Inputs to be supplied by primary actor and constraints on input: In case the user wishes to edit their personal information, the new value should make sense. For example, phone number should be of 10 digits only. User cannot modify history of diseases identified at the hospital since they are linked to appointments and need doctor's approval. For past history the user will only be able to choose from diseases already present in the system and remove history items which are already present
6. Preconditions for the use case: Should be a registered user
7. Main success path: The user should be able to see their dashboard once they login
8. Exceptions that may arise and how they will be compensated: The user might enter an invalid value. They will be forbidden to do so by displaying an error message
9. Post conditions: The state of the database will change only if the user edits. If the edit about personal information is sane, it is reflected in the user's entry in the table. If the edit about history is sane, change will be reflected in the past_history relation

4. ***Book/cancel appointment***
    1. Title of Use case: "Book/cancel new appointment"
    2. Description: This should allow the user to book a new appointment or cancel an already scheduled appointment with a doctor.
    3. Trigger event for the use case: Human triggered
    4. Primary actor: User
    5. Inputs to be supplied by primary actor and constraints on input: The user should select the doctor and the time slot during which they want to schedule an appointment. If they wish to cancel, they need only select an already scheduled appointment. An obvious constraint here is that the doctor must be available during the slot the user chooses
    6. Preconditions for the use case: Should be a registered user
    7. Main success path: The user first selects whether they wish to book or cancel an appointment. They are taken to the relevant page where they are asked for the relevant input
    8. Exceptions that may arise and how they will be compensated: No exceptions are expected to arise at the time of booking/cancelling an appointment

9. Post conditions: If the patient wishes to book an appointment: First of all, a new entry in the patient table is made for the user, ie the user is assigned a new patient_id for every *fresh* appointment that they make. Then, a new entry is added to the “appointment” relation. If the patient wishes to cancel an appointment then the appropriate entry will be removed from the “appointment” relation

5. *Book/cancel follow up*
    1. Title of Use case: “Book/cancel follow up”
    2. Description: It should allow a user to book or cancel a follow up appointment for an appointment they have already made in the past
    3. Trigger event for the use case: Human trigger
    4. Primary actor: User
    5. Inputs to be supplied by primary actor and constraints on input: The user will first have to select the “case” for which they want to book/cancel a follow up. Then the procedure is similar to booking/cancelling fresh appointments
    6. Preconditions for the use case: The registered user should already have made at least one appointment with a doctor.
    7. Main success path: The user first selects the “case” then they select whether they wish to book or cancel a follow up. They are taken to the relevant page where they are asked for the relevant input
    8. Exceptions that may arise and how they will be compensated: no exception are expected to arise at the time of booking/cancelling a follow up
    9. Post conditions: Corresponding entry is added/removed from the appointments relation. Note that no change is made to the patient table in this case

6. Book/cancel tests
    1. Title of Use case: “Book/cancel tests”
    2. Description: It should allow a user to book or cancel a test at the hospital
    3. Trigger event for the use case: Human trigger
    4. Primary actor: User
    5. Inputs to be supplied by primary actor and constraints on input: The user should first select either a pre-existing case under which they wish to book a test or choose to open a new case for the test. The user then selects the test which they want to book. If they wish to cancel, they need only select an already booked test
    6. Preconditions for the use case: Should be a registered user
    7. Main success path: The user first selects either a pre-existing case under which they wish to book a test or choose to open a new case for the test. They are then taken to a book/cancel test page where they are asked for relevant input
    8. Exceptions that may arise and how they will be compensated: no exception are expected to arise at the time of booking/cancelling a test

9. Post conditions: If the user has opened a new case then a new entry is added to the patient table for the user. Appropriate entry is then added/removed from the “takes” relation

7. Add test results
    1. Title of Use case: “Add test results”
    2. Description of what the use case is meant to accomplish for the user: Allow user to add the results obtained from the various medical tests performed (like blood test, MRI scanning)
    3. Trigger event for the use case - Human triggered
    4. Primary actor - Support staff
    5. Inputs to be supplied by primary actor and constraints on input: The reports (pdfs, image files, etc.) generated by the test, a comment like - “Normal”/”Abnormal” describing the outcome of the test. Report size must be limited by some upper bound (like 5 Mb). The exact limit will be decided based on performance.
    6. Preconditions for the use case - state of data - what is the world state that the use case is relevant for: Concerned patient must have a pending request to perform the concerned test.
    7. Main success path - what is the set of interactions with the system and the final output: User uploads files, gives a short comment, enters the cost and clicks “Save changes”. A message indicating success pops up, or an error message pops up to indicate failure.
    8. Exceptions that may arise and how they will be compensated: File size may exceed the upper limit. User will be indicated about the large file size. Unless the user reduces file size, one cannot proceed with the transaction.
    9. Post conditions - after executing this use case, what will be the state of the system: The request for this test is marked completed, and the inputs given by the support staff get saved.
    10. `insert into takes values ($1, $2, $3, $4, $5,$6);` note: $6 is bill_no, so optional

8. ***Add disease***
    1. Title of Use case: Add a new disease to db
    2. Description of what the use case is meant to accomplish for the user.: Allow doctor to add a new disease in the database
    3. Trigger event for the use case - Human triggered. Doctor clicks on option to add disease
    4. Primary actor - Doctor
    5. Inputs to be supplied by primary actor and constraints on input.: Disease name, description, link to webpage containing information about the disease

6. Preconditions for the use case - state of data - person adding the disease must be a doctor
7. Main success path - what is the set of interactions with the system and the final output. : User enters the details about the disease, provide necessary links for the disease then click "Save changes".
8. Exceptions that may arise and how they will be compensated.: Previously added disease may have been given again as input in which case, no changes to the database will be made
9. Post conditions - Required disease will be added to the database
10. `insert into disease (id, name, info_link) values ($1, $2, $3);` (note: $1 auto-incremented?)

9. ***Add symptoms***
    1. Title of Use case: "Adding new symptom"
    2. Description of what the use case is meant to accomplish for the user.: Doctor adds a new symptom to db
    3. Trigger event for the use case - human triggers
    4. Primary actor - Doctor
    5. Inputs to be supplied by the primary actor and constraints on input. : Symptom name
    6. Preconditions for the use case -: - Person adding symptom must be a doctor
    7. Main success path - what is the set of interactions with the system and the final output. :Doctor enters the symptom name and then click save changes.
    8. Exceptions that may arise and how they will be compensated.: If the symptom already exists in the db, current request will be denied and no changes to db. Returned to the previous state at home page
    9. Post conditions - after executing this use case, what will be the state of the system?: A new symptom has been added
    10. `insert into symptom (id, name, desc) values ($1, $2, $3);`

10. ~~Set up event~~
    1. ~~Title of Use case: Adding a details for a new awareness event to be organised~~
    2. ~~Description of what the use case is meant to accomplish for the user.: The event manager who is an admin, should be able to add a new event and details.~~
    3. ~~Trigger event for the use case - human triggers~~
    4. ~~Primary actor - admin~~
    5. ~~Inputs to be supplied by primary actor and constraints on input. : Admin fills the event details, select the organisers of the event, select doctor in-charge for the event~~
    6. ~~Preconditions for the use case - state of data - Person must be a admin~~
    7. ~~Main success path - On clicking "add new event", the admin enters necessary details like date, time, name of the event, link to the event and its description and then clicks "save"~~

~~which shows success on successful addition of an event to event list. Then chooses the organisers from the registered user list, selects a doctor in-charge for the event.~~

8. ~~Exceptions that may arise and how they will be compensated.: if another event is already scheduled during same time, system warns user about the same and asks for a new time,date~~
9. ~~Post conditions - The event has been successfully added to the event table~~

11. ***Allot room for appointment***
    1. "Allot room for appointment"
    2. Allow user to allot a room for the appointment
    3. Human triggered
    4. Primary actor - Admin
    5. The user specifies the room (room_id), appointment (appo_id). The room should not be occupied for another appointment in the same time slot.
    6. There must be a valid pending appointment.
    7. System asks which appointment to schedule. User enters the id. System asks which room to allot. User enters the id and clicks 'Ok'. System outputs a success message.
    8. If the room is already occupied in that time slot, a message indicating this is thrown at the user. The user must change the room to proceed.
    9. The room gets allotted for the concerned appointment.
       Edit: Made it "change room of a row in DRS table"
    10. `(select room_no from room) except (select room_no from doctor_room_slot where (dat, start_time) = ($1, $2));` => to display free rooms in the selection box.
        `update doctor_room_slot set room_no = $1 where (doc_id, dat, start_time) = ($2, $3, $4);` => to update the row in the table

12. ***Decide slot:***
    1. "Decide slots for doctor"
    2. Doctor can choose/edit his/her slot for the week from the given slots
    3. human triggers
    4. Primary actor - Doctor
    5. Date for which slot needs to be chosen and then click on required slots
    6. Preconditions for the use case - state of data - what is the world state that the use case is relevant for.: Can only select for upcoming days
    7. Main success path - Doctor first selects "Select slot" and then provide the week/date for which slot will be decided and then selects the slots out of given options.
    8. **Exceptions** that may arise and how they will be compensated.: If doctor changes the slot in some day for which a patient has already booked appointment, then the appointment

will be cascaded, patient will get a system generated mail about the change in slot and he/she can call reception (for priority re-scheduling) to select a new slot

9. Post conditions - Slot have been successfully selected for the doctor
10. Add slot:
    a. with free_rooms (room_no) as ((select room_no from room) except (select room_no from doctor_room_slot where (dat, start_time) = ($1, $2))) select count(*) from free_rooms; => check whether there is a room available for the slot the doc wants to add. If not, say sorry to the doctor and that there is no room available.
    b. If rooms are available, `with free_rooms (room_no) as ((select room_no from room) except (select room_no from doctor_room_slot where (dat, start_time) = ('2020-01-01', '10:30'))), chosen_room(room_no) as (select min(room_no) from free_rooms) insert into doctor_room_slot select 1, room_no, '2020-01-01', '10:30' from chosen_room;`

13. ***Register/remove Doctor/ support staff, modify salary, charges etc***
    1. "Register/remove Doctor/ support staff, modify salary, charges"
    2. Description of what the use case is meant to accomplish for the user.: Admin can do any of the following 1) Add/ remove doctor/support staff 2) Modify their salaries 3) Update the charges for various tests
    3. Trigger event for the use case - human triggers
    4. Primary actor - Admin
    5. Inputs to be supplied by primary actor and constraints on input. : Choose from available options to either add/remove staff/doctor, provide input for salary changes or update the rate of tests. Salary and test charges must be positive.
    6. Preconditions for the use case - state of data: User must be a admin
    7. Main success path - what is the set of interactions with the system and the final output. : User selects the operation then for adding doctor/staff, user selects from registered persons and then fills their attributes and the clicks "Save" For removing staffs, select from either doctor or staff list and then click "ok". For modifying rates of tests, select the item, choose "update" fill in required details the click "ok".
    8. Exceptions that may arise and how they will be compensated.: In case staff is already added, error message "Person already is a staff" will appear. For salary updates or test charges, non-positive values will give error and new value will be required to be added
    9. Post conditions - Staff has been added/ salary-test charges have been updated

14. Fill prescriptions, symptom, disease
    1. "Fill prescriptions, symptoms, disease for a patient"

2. Description of what the use case is meant to accomplish for the user.: Doctor must be able to fill in the symptoms, disease and prescriptions for the patient
3. Trigger event for the use case - human triggers
4. Primary actor - Doctor
5. Inputs to be supplied by primary actor and constraints on input. : Select the patient, choose add prescriptions, symptoms, disease each and then fill in the necessary details.
6. Preconditions for the use case - Patient has booked appointment with doctor for present slot and has been through checkup by doctor.
7. Main success path - Doctor selects the patient in current slot, click on "Add prescription", provide necessary doses for medicines, ask for required test reports, then save it. "Add symptoms and diseases": fill in the necessary information
8. Exceptions that may arise and how they will be compensated.: NA
9. Post conditions - Prescriptions and disease, symptom has been added to the appointment of person

15. Allot beds to admitted patients
    1. Title of Use case: "Allot beds to admitted patients"
    2. Description of what the use case is meant to accomplish for the user.: Support staff provides a room and bed to a to be admitted patient
    3. Trigger event for the use case - human triggered
    4. Primary actor - Doctor
    5. Inputs to be supplied by primary actor and constraints on input.: Patient details, type of room, type of bed
    6. Preconditions for the use case - doctor will allot a bed to the patient and the same doctor will be the visiting doctor for that patient-bed
    7. Main success path - Doctor staff selects "Get Bed" option then selects the appointment id of the patient, room type, bed type and then the system will automatically find a bed with given requirements and the room number for that.
    8. Exceptions that may arise and how they will be compensated.: In case no bed with required type is available then system shows a pop-up for the same and shows number of other types of beds available
    9. Post conditions - patient has been assigned a bed successfully
    10. `select distinct type from room where type is not null;` => shows all room types
        `select distinct type from bed where type is not null;` => shows all room types
        ```
        with free_beds(bed_no) as ((select bed_no from bed natural join room where bed.type = '1' and room.type = '2') except (select bed_no from bed natural join room natural join occupies where bed.type = '1' and room.type = '2' and end_dt is null)) select
        ```

```
count(*0) from free_beds; => if this returns 0, display pop-up.
Else,
with free_beds(bed_no) as ((select bed_no from bed natural join
room where bed.type = '1' and room.type = '2') except (select
bed_no from bed natural join room natural join occupies where
bed.type = '1' and room.type = '2' and end_dt is null)) insert
into occupies select ‘patient_id’, min(bed_no), cur_date' from
free_beds;
=> this adds an entry into the occupies table. Current date and
patient id need to be passed to it. Bill id and end dt will be
taken care of later.
```

16. ***Update inventory***
    1. “Update inventory”
    2. Allow the user to update inventory (like add new equipment, add new rooms/beds, add new medicines, etc.)
    3. Human triggered
    4. Primary actor - Support staff
    5. The user specifies the type of goods, type of change(addition/deletion). The user then specifies the name of (the new) good
    6. NA
    7. System lists down the type of goods in the hospital. User either clicks on one of them, system asks whether user wants to delete it or user adds a new good. System shows a form to enter info about new item. User enters it, clicks “Ok”. System shows a success message.
    8. If new good has already been added before (i.e. no longer new), alert the user.
    9. The list of goods is saved.

17. ~~Register for event~~
    1. ~~“Register for event”~~
    2. ~~Allow user to register for an event.~~
    3. ~~Human triggered~~
    4. ~~Primary actor - Person~~
    5. ~~The user clicks on the event which one wants to attend.~~
    6. ~~This event must be a part of a list of events.~~
    7. ~~System shows a list of events. User clicks on one. System asks for verification to register for the event. User clicks “Ok”.~~
    8. ~~No exceptions.~~
    9. ~~The person is registered for the event.~~

18. ***Register as a person***
    1. “Register as member of the hospital”
    2. Allow user to register as a member of the hospital.
    3. Human triggered
    4. Primary actor - External Person
    5. User specifies name, email, DOB, address, new_password. DOB must be valid (not from future), email should be valid (at least it should look like a valid email id), new_password may have some constraints to improve security (these are not decided yet).
    6. No prerequisites.
    7. System shows a button to “Sign up”. User clicks and is shown a form to enter the details.
    8. System alerts the user to enter valid input in either of the exceptions.
    9. The person is registered as a member of the hospital with a login id and password.

19. ***View generic information***
    1. “View general details about the hospital”
    2. Allow anyone to view the details of the hospital (like doctors, staffs, facilities, departments, number of beds).
    3. Human triggered
    4. Primary actor - Person (even unregistered)
    5. User will have click a tab and apply some filters.
    6. No prerequisites.
    7. System shows some tabs. User clicks one of them (say Doctors), applies some filters like (Doctors specialized for nervous disorders). System outputs the doctors along with their information.
    8. NA
    9. No change in the database.

20. ***Payment***
    1. “Make payment”
    2. Allow user to pay for the services, medicines.
    3. Human triggered
    4. Primary actor - Person
    5. User verifies the bill, confirms to make the payment. System redirects the user to payment gateway.
    6. There should be an outstanding bill.
    7. System shows a list of pending bills. User clicks on one of them. System shows the details of the bill. User verifies the bill, clicks “Ok”.

8. In case the user denies the bill, system asks the user to specify what is invalid in the bill. System asks if the user wants to proceed and send an email to the hospital regarding the same. User clicks “Yes”, system sends an email to the concerned authority.
9. The bill has been paid. This purchase is also added to the person’s history.

21. ***Disease-symptom***
    1. “Query the symptoms arising due to a disease”
    2. Allows anyone to find out which symptoms usually are indicators of a disease.
    3. Human triggered.
    4. Person
    5. Disease name, must be known to the DB
    6. Patients must have visited the appointments previously facing the disease - only then will their symptoms be available to the system
    7. User enters disease name, an applications runs in the backend and delivers symptoms of the disease
    8. The disease may not exist in the DB, a simple message will be shown to user expressing the system’s regret. If enough data is unavailable, then also the system may choose to withhold the partial analytics and show the message
    9. No change in DB.

22. ***Disease-area***
    1. “Query the most common disease in an area (represented by pincode)”
    2. Allows anyone to find out which areas are prone to which disease and take the necessary preventive steps.
    3. Human triggered.
    4. Person
    5. Area code must be known to the DB
    6. Patients must have visited the appointments previously facing the disease - only then will their diseases be available to the system.
    7. User enters area code, an applications runs in the backend and delivers diseases
    8. The area code may not exist in the DB, a simple message will be shown to user expressing the system’s regret. If enough data is unavailable, then also the system may choose to withhold the partial analytics and show the message
    9. No change in DB.

23. ***Bill generation***
    1. Title of Use case: "Bill generation"
    2. Description of what the use case is meant to accomplish for the user.: Support staff with special privileges should be able to generate the bill for some activity
    3. Trigger event for the use case - Human triggered

4. Primary actor - Support staff with billing privileges
5. Inputs to be supplied by primary actor and constraints on input.: The activity (buying of medicines, appointment or test) for which bill has to be generated. In case of a pharmacy bill (buying medicines) the user should specify what medicines were brought and in what quantity and then click on “Generate bill”. In other cases, they need only select the activity and standardised bill would be produced
6. Preconditions for the use case - Support staff should have “billing” privileges. If bill for appointment or test taken is being generated, appropriate entry for the test taken and appointment should already exist.
7. Main success path -User selects the appropriate inputs at the “Bill generation” pane and clicks on generate bill. A bill for the entity is generated and a print option is presented to the user.
8. Exceptions that may arise and how they will be compensated.: No exceptions are expected to arise during bill generation since we enforce that the user can click on “generate bill” only once
9. Post conditions - Corresponding entry is added to the bill table. Relationships of the bill to appointments, tests, etc are also filled accordingly.

Technology choices:

On the backend side we would like to commit to using Flask for the server and PostgreSQL for the database. The reason for our choice is the versatility that Flask and PostgreSQL provide. On the frontend side we do not have any technology preferences yet due to lack of experience.