



**Hyacinth**  
Hospital Management

# Hyacinth – Hospital Management

*Yogesh R, 111801047, Team-2*

# Contents

---

<b>INTRODUCTION .....</b>	<b>3</b>
<b>ENTITIES .....</b>	<b>4</b>
USER ENTITIES.....	4
MANAGEMENT ENTITIES .....	4
RELATIONSHIP SETS .....	5
<b>AUTHENTICATION.....</b>	<b>6</b>
EMPLOYEE LOGIN.....	6
PATIENT LOGIN.....	7
<b>OPERATIONS ON USERS.....</b>	<b>8</b>
PATIENT REGISTRATION.....	8
ADD NURSE/DOCTOR/DRIVER/ADMIN .....	8
DELETE EMPLOYEE .....	9
<b>VIEWS, ROLES AND AUTHORIZATION .....</b>	<b>10</b>
<b>WORKING IN UNISON .....</b>	<b>11</b>
PRESENTATION .....	11
APPLICATION.....	11
DATABASE .....	11
FLOW-CHART .....	11
<b>DASHBOARDS .....</b>	<b>12</b>
ADMIN.....	12
DOCTOR.....	13
PATIENT .....	13
NURSE .....	14
DRIVER .....	14
VENDOR.....	14
<b>DETAILED PAGES.....</b>	<b>15</b>
CONSULTATION DETAILS.....	15
PRESCRIPTION DETAILS .....	15
PROFILE PAGE .....	16
<b>CONTRIBUTIONS.....</b>	<b>17</b>
<b>APPENDIX .....</b>	<b>18</b>

## Introduction

---

Hospitals are the essential part of our lives, providing the best medical facilities to people suffering from various ailments, which may be due to change in climatic conditions, increased workload, emotional trauma stress, etc. The hospital must keep track of its day-to-day activities & records of its patients, doctors, nurses, and other staff personnel that keep the hospital running smoothly & successfully. But keeping track of all the activities and their records on paper is very cumbersome and error-prone. It also is very inefficient and a time-consuming process, observing the continuous increase in population and number of people visiting the hospital. It is also not economically & technically feasible to maintain these records on paper. Thus, keeping the working of the manual system as the basis of our project. We will develop an automated version of the manual system, named “**Hyacinth**”. The main aim of our project is to provide a paperless hospital.

Hyacinth is a website with a simple, intuitive user interface powered by a durable, scalable, and efficient backend. It offers features like employee management, supply-chain management and eases the workload of record maintenance. It maintains details like profiles, consultations, appointments, pharmacy, ambulance, etc.

This website is made using APIs as a data provider instead of rendering HTML inside backend itself, thus providing versatility for automation software and technologies to empower its fullest. Docker technology is used to make it more scalable and easily deployable.

## Entities

---

### User Entities

#### Employee

The employee is the inheritance entity for the following four user entities.

S.NO	Name	Description
1	Admin	Admin has complete control over the website
2	Doctor	A doctor can view their consultations and the details that follow it.
3	Nurse	A nurse can view the list of diagnostics done by them
4	Driver	A driver can view the list of journeys done by them

#### Patient

The patient can view all their consultations and the details that follow them.

### Management Entities

S.NO	Name	Description
1	Vendor	The vendor supplies the drug to the pharmacy
2	Pharmacy	Pharmacy stores organize and provide drugs.
3	Diagnostics	The diagnoses of patients are stored
4	Consultation	It is a collection of sessions with the doctor on a particular problem.
5	Appointment	A singular interaction with the doctor under one consultation. A patient can have multiple appointments under the exact consultation.
6	Invoice	It is the total amount the patient has to pay for his appointment.
7	Prescription	It has all the drugs and diagnostics advised by the doctor during the appointment.
8	Ambulance	It will be assigned to a driver in case of emergency pickups.

## Relationship Sets

S.NO	Name	Type	Description
1	Attachment	One-to-One	Each appointment has a prescription given by the doctor.
2	Payment	One-to-One	Each appointment has one invoice that needs to be paid.
3	Participates	One-to-Many	Every patient that comes to the hospital gets a consultation from a doctor.
4	Performs	One-to-Many	A doctor can have multiple consultations, but one consultation cannot have various doctors.
5	Sessions	One-to-Many	A consultation will have many appointments in it. And each appointment will be under a session.
6	Assist	Many-to-Many	Every nurse can assist with any diagnosis. Also, every diagnostic needs a nurse, and hence there is full participation from the Diagnostics entity.
7	Contains	Many-to-Many	A Prescription may contain multiple diagnostics from the Diagnostics entity, and the same diagnostics can be performed in multiple prescriptions. Also, there is no full participation from either of the entities because there can be a prescription with no diagnostics. Diagnostics can indeed not be mentioned in any of the prescriptions.
8	Drugs	Many-to-Many	Every prescription can have multiple items from the pharmacy, and pharmacy items can be assigned to multiple prescriptions.
9	Journey	Many-to-Many	A driver can drive many ambulances, and many drivers can drive one ambulance.
10	Supplies	Many-to-Many	A particular Drug in the pharmacy can be from several different vendors, and a specific vendor can supply multiple drugs to the pharmacy

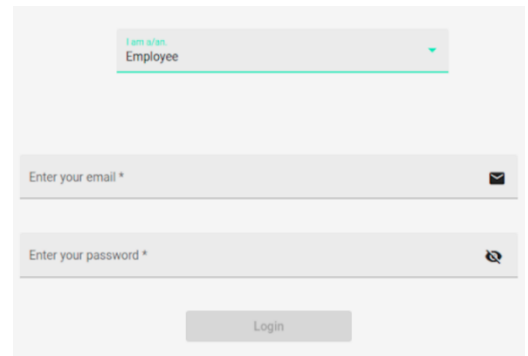
A visual representation can be seen by the [ER diagram](#) and [Schema diagram](#)

# Authentication

## Employee Login

There exists no employee registration as only the admin has the authorization to create new employees. The admin is using the add<employee> procedure.

The employee Login requires two fields: email address and password. These details are given to the procedure “[Employee Login](#)” which does the following:



- Check if the user exists using the “[User Exists](#)” function.
- If yes, then check the password; if wrong, reject it; else, return the role using “[Employee Role](#)” function.
- If no then check if it exists in employee table, if no then reject, else create new user (implies first time login) and return the role using “[Employee Role](#)” function.

We can note that if we get the role then our authentication is successful.

Figure 1: Employee Login

```
create procedure EmployeeLogin(in email_ varchar(255), in password_
varchar(255), out role varchar(11), out id int)
begin
    -- If user exists, then check password
    if user_exists(email_)
    then
        set @password = (select password from Employee where
email=email_);
        if (PASSWORD(password_)=@password) then
            set role=employee_role(email_);
            set id = (select employeeID from Employee where email=email_);
        else
            set role='None';
            set id = -1;
        end if;
    -- If user does not exist, check if the admin has added him/her as an
employee.
    elseif exists(select * from Employee where email=email_)
    then
        set @create_user = concat('create user \'',
substring_index(email_, '@', 1),
'\@\'localhost\' identified by \'', password_, '\');');
        call exec_query(@create_user);
        set @update_user = concat('update Employee set password=\'',
PASSWORD(password_), '\\' where email=\'', email_, '\');');
        call exec_query(@update_user);
        set role=employee_role(email_);
        set id = (select employeeID from Employee where email=email_);
    -- If the admin has not added the Employee, don't allow registration.
    else
        set role='None';
        set id = -1;
    end if;
end; //
```

Figure 2: Employee Role

```
create function employee_role(email_ varchar(255))
returns varchar(20)
begin
    if exists(
        select * from Doctor
        where doctorID=(select employeeID from Employee where
email=email_)
    )
    then
        return 'Doctor';
    elseif exists(
        select * from Nurse
        where nurseID=(select employeeID from Employee where email=email_)
    )
    then return 'Nurse';
    elseif exists(
        select * from Driver
        where driverID=(select employeeID from Employee where
email=email_)
    )
    then return 'Driver';
    elseif exists(
        select * from Admin
        where adminID=(select employeeID from Employee where email=email_)
    )
    then return 'Admin';
    else return 'Invalid';
    end if;
end //
```

Figure 3: User Exists

```
create function user_exists(email varchar(255))
returns bool
begin
    return (
        exists(select User from mysql.user
        where User=substring_index(email, '@', 1))
    );
end //
```

## Patient Login

The patient login is similar to the employee login. It takes in the email and passwords. But unlike the employee login, we reject the patient if not found in the patient entry, in with case the user will be prompted to register before attempting to login. This is done using “[Patient Login](#)” procedure

As show in the figure the user can switch to registration if it's the first time in this website.

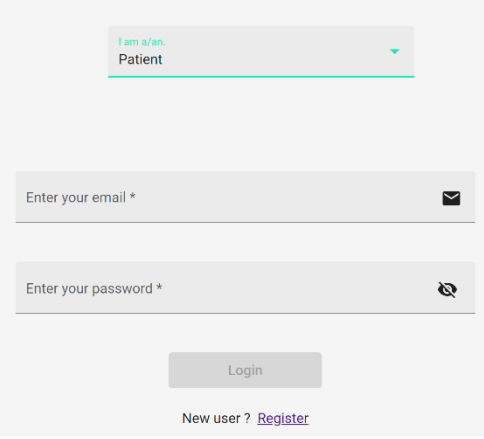
A screenshot of a web form for patient login. At the top, there is a dropdown menu with the text "I am a/an. Patient" and a green arrow pointing down. Below this are two input fields: "Enter your email \*" with an envelope icon on the right, and "Enter your password \*" with an eye icon on the right. At the bottom, there is a "Login" button and a link that says "New user ? [Register](#)".

Figure 4: Patient Login

```
create procedure PatientLogin(in email_ varchar(255), in password_ varchar(255), out id int)
begin
    if user_exists( email: email_)
    then
        set @password = (select password from Patient where email=email_);
        if (password(password_)=@password) then
            set id = (select patientID from Patient where email=email_);
        else
            set id = -1;
        end if;
    elseif exists(select * from Patient where email=email_)
    then
        set @create_user = concat('create user \'', substring_index(email_, '@', 1),
        '\@\'localhost\' identified by \'', password_, '\';');
        call exec_query( query_str: @create_user);
        set @update_user = concat('update Patient set password=\'', PASSWORD(password_),
        '\\' where email=\'', email_, '\');');
        call exec_query( query_str: @update_user);
        set id = (select patientID from Patient where email=email_);
    else
        set id = -1;
    end if;
end //
```

## Operations on Users

The patients are added using the registration process and the employees are added by the admin. There are some integrity constraints placed on these entities as follows:

1. *PhoneCheck*: This ensures the phone number is a valid Indian number; implemented using regular expressions.
2. *SexCheck*: This makes sure that only 'Male', 'Female', or 'Others' strings are entered.
3. *EmailCheck*: This ensures the email is a valid; implemented using regular expressions.

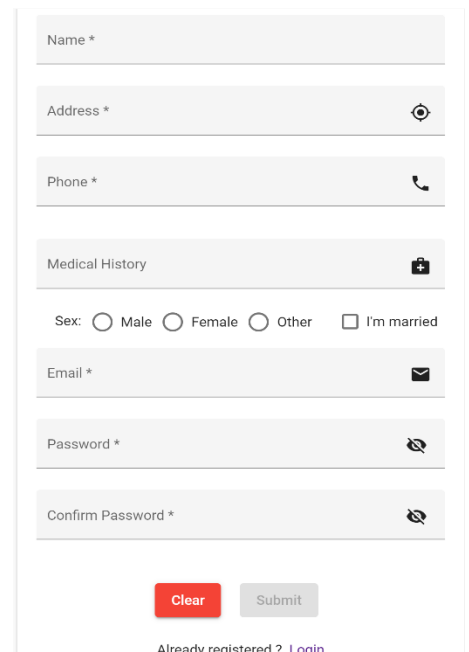
There are many more integrity constraints implemented on other entities and can be found in [schema.sql](#). For implementational proof see "[Appendix](#)"

### Patient Registration

The patient registration takes the following as inputs: Name, Address, Phone number, Email, Medical History (optional), Sex, Password, Confirm password. Once the user has entered all the information, the "[Patient Registration](#)" procedure is called. This procedure does the following:

1. The new patient record is added with these details.
2. A new User is created with the role patient.

If the user already exists then this procedure fails and the user is prompted to login.



The form contains the following fields and controls:

- Name \*
- Address \* (with a visibility toggle icon)
- Phone \* (with a phone icon)
- Medical History (with a plus icon)
- Sex: ☐ Male ☐ Female ☐ Other ☐ I'm married
- Email \* (with an email icon)
- Password \* (with a visibility toggle icon)
- Confirm Password \* (with a visibility toggle icon)
- Buttons: Clear (red), Submit (grey)
- Footer: Already registered? [Login](#)

Figure 5: Patient Registration

```
create procedure PatientRegistration(name_ varchar(45), email_ varchar(255), password_ varchar(255),
                                   phone varchar(20), address_ varchar(255), sex varchar(20),
                                   medicalHistory varchar(255), marital bool, out id int
)
begin
    if user_exists(email_ email_)
    then
        set id=-1;
    else
        set @email = substring_index(email_, '@', 1);
        set @create_user = concat('create user \'', @email,
                                   '\'\@localhost\' identified by \'', password_, '\';');
        call exec_query( query_str: @create_user);
        set @insert_user = concat(
            'insert into Patient (name, email, phone, password, \' ||
            'address, sex, medicalHistory, marital) values (\'',
            '\', name_, '\',', '\', email_, '\',', '\', phone, '\',',
            '\', PASSWORD(password_), '\',', '\', address_, '\',',
            '\', sex, '\',', '\', medicalHistory, '\',', '\', marital, '\')'
        );
        call exec_query( query_str: @insert_user);
        set id = (select patientID from Patient where email=email_);
    end if;
end; //
```



## Add Nurse/Doctor/Driver/Admin

There are procedure corresponding to each role, which first insert all the basic details into the employee table (parent table), then using the last inserted id we insert it into the corresponding role table (child table, inheritance).

One such example is adding doctor as show below. Other procedures can be found in [procedure.sql](#)

Figure 6: Add Doctor

```
create procedure addDoctor(name_ varchar(45), password_ varchar(256), phone_ varchar(45),
                           email_ varchar(255), address_ varchar(45), sex_ varchar(45),
                           salary_ varchar(45), qualification_ varchar(50), license_ varchar(50),
                           bio_ varchar(255), available_ bool, specialization_ varchar(255), out id int)
begin
    insert into Employee (name, password, phone, email, address, sex, salary)
    values (name_, password(password_), phone_, email_, address_, sex_, salary_);
    set id = last_insert_id();
    insert into Doctor (doctorID, qualification, license, bio, available, specialization)
    values (id, qualification_, license_, bio_, available_, specialization_);
end //
```

## Delete Employee

The procedure corresponding to this operation deletes the record matching the input ID from the employee table, the cascading effect also removes the entry from the roles.

Figure 7: Delete Employee

```
create procedure delEmployee(id int)
begin
    set @email = (select email from Employee where employeeID=id);
    if user_exists(email: @email)
    then delete from mysql.user where User=substring_index(@email, '@', 1);
    end if ;
    delete from Employee where employeeID=id;
end //
```

## Views, Roles and Authorization

View is created to restrict user scopes and simplify complex queries. Every user has their *<user>Info* view which gives the profile information of the user. There is other view based on the relations named as “*<entity1><entity2>Info*” or “*<relation>Info*”.

There are five roles corresponding to each user: Admin, Doctor, Nurse, Driver, and Patient. Admin corresponds to the root user in database. The other roles have their authorizations as follows.

S.NO	Role	Authorization
1	Doctor	Select on <i>doctorInfo</i> and <i>doctorPatientInfo</i> views
2	Nurse	Select on <i>nurseInfo</i> and <i>nurseProfile</i> views
3	Driver	Select on <i>DriverInfo</i> and <i>journeyInfo</i> views
4	Patient	Select on <i>PatientInfo</i> and <i>PatientDoctorInfo</i> views

An example of such view is as follows for doctor. Other views can be seen in [views.sql](#)

```
create view DoctorPatientInfo as (  
    select * from DoctorInfo  
        inner join Consultation C using (doctorID)  
        inner join Appointment A using (consultationID)  
        inner join Prescription Pr using (prescriptionID)  
        inner join Drugs D using (prescriptionID)  
        inner join Pharmacy Ph using (pharmacyID)  
);
```

```
create view DoctorInfo as (  
    select d.*, name as doctorName,  
           phone, email, address, sex  
    from Doctor d  
        inner join Employee e on d.doctorID = e.employeeID  
);
```

The insert, update and delete operation on other entity can be done only by the admin.

Non-admin users can only view the detail of their own entities and relations. These are achieved by simple procedures of the form “*<entity>profile (in ID int)*” with their body using the appropriate views mentioned above. One such example is as follows. Other “userProfiles” can be seen in [procedure.sql](#)

```
create procedure DoctorProfile(ID int(11))  
begin  
    select * from DoctorInfo where doctorID = ID;  
end; //
```

## Working in unison

---

Hyacinth follows the three-tier architecture: presentation, application and database.

### Presentation

The presentation module was done using Angular, a TypeScript-based open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. The versatility and good documentation made us choose Angular as the frontend development framework.

### Application

The application module was done using FastAPI, a modern, fast (high-performance), web framework for building APIs. The simplicity, and robustness of FastAPI made it a prominent choice for backend development.

### Database

The database was done using MariaDB, a community developed, open-source relational database management system.

### Flow-Chart



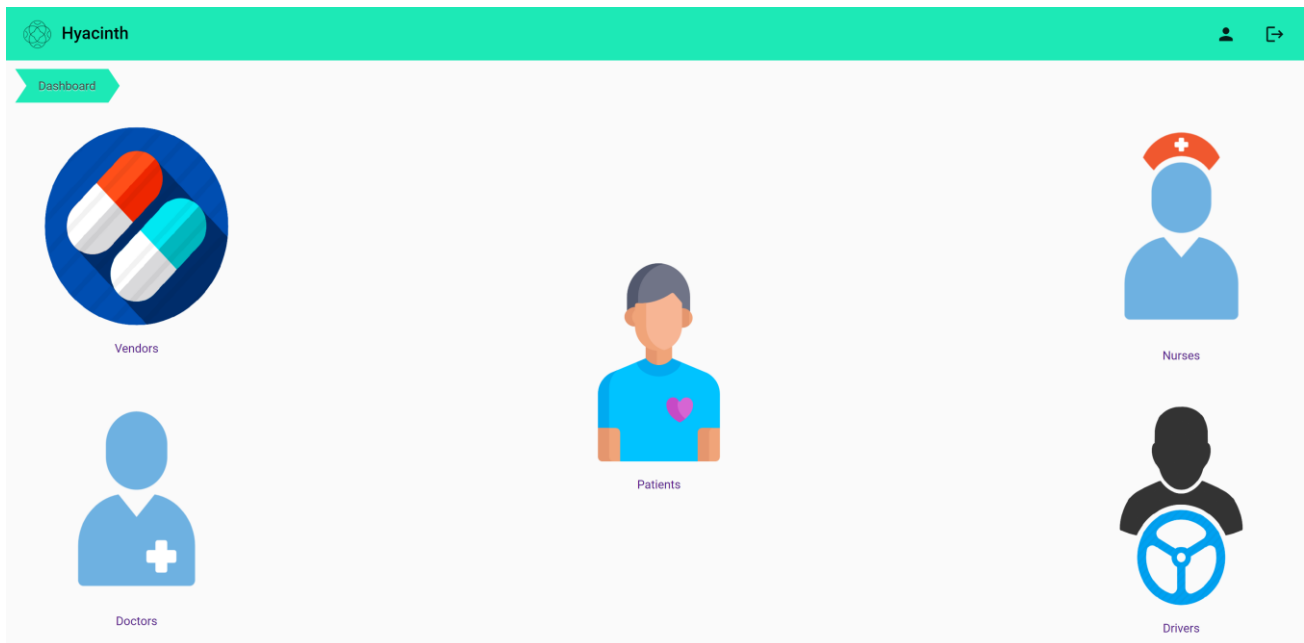
## Dashboards

Every role has their personal dashboard showing the information necessary for that role.

### Admin

The admin dashboard has five icons corresponding to doctor, patient, nurse, driver, vendor.

Figure 8: Admin Dashboard



Clicking on any of those icons, directs the user to the list of those entities. E.g., clicking on the doctor icon, will show a page with all the doctor in the hospital.

Figure 9: List of Doctors

The screenshot shows the 'List of Doctors' page in the Hyacinth application. It features a green header with the 'Hyacinth' logo and a user profile icon. Below the header is a breadcrumb showing 'Dashboard' and 'Doctor'. The main content area has a title 'Doctors' and an 'Add' button. A table displays a list of doctors with columns for Employee ID, Name, Phone, Email, Sex, and Actions. The table contains 11 rows of data.

Employee ID	Name	Phone	Email	Sex	Actions
2	Dana Bethea	+91 6276875343	Nance@nowhere.com	Male	
3	Jeromy Vanburen	+91 8420416516	ShannonAbney@example.com	Male	
4	Adan May	+91 7770548164	Celina_Cota428@example.com	Male	
5	Charline Berman	+91 9055163400	Stephens6@example.com	Male	
6	Roosevelt Lloyd	+91 8863501623	GainesM@nowhere.com	Male	
7	Carson Burley	+91 9445684470	Aguirre@example.com	Male	
8	Gabriel Cheatham	+91 8816502572	Tomlinson@nowhere.com	Male	
9	Elfrieda Fennell	+91 7375552166	Lanny.Boehm592@nowhere.com	Male	
10	Antone Coles	+91 9210631867	KylaBoehm@nowhere.com	Male	
11	Adelaida Mcfall	+91 7750357597	Boyles241@nowhere.com	Male	

Clicking on a row directs the user to that user's dashboard. Clicking on the user icon on each row directs the user to that user's profile page. As mentioned earlier, only admin can add, and delete users. The add button in the top of the table and the trash button on every row corresponds to insertion and deletion of the user. Clicking on the add brings a popup (as shown in [Figure 10](#)) where all the necessary details can be given. This calls the "[Add Doctor](#)" procedure

Figure 10: Add doctor

Doctors Details

Name \*

Password \*

Phone \*

Email \*

Address \*

Sex: ☐ Male ☐ Female ☐ Other

Salary \*

Pin \*

Close Submit

## Doctor

The doctor's dashboard contains all the consultations done by this doctor.

Figure 11: Doctor Dashboard

Hyacinth

Doctor Dashboard

Welcome Dana Bethea to your dashboard

Consultation ID	Patient ID	Patient	Problem
5	50	Charlena Rasmussen	Nobis impedit quisquam. A.
16	46	April Calabrese	Neque natus deleniti... Magnam.

Click on a row directs the user to the [Consultation Details](#).

## Patient

The patient's dashboard contains all the consultations done for this patient.

Figure 12: Patient Dashboard

Hyacinth

Dashboard Patient Patient Dashboard

Charlena Rasmussen

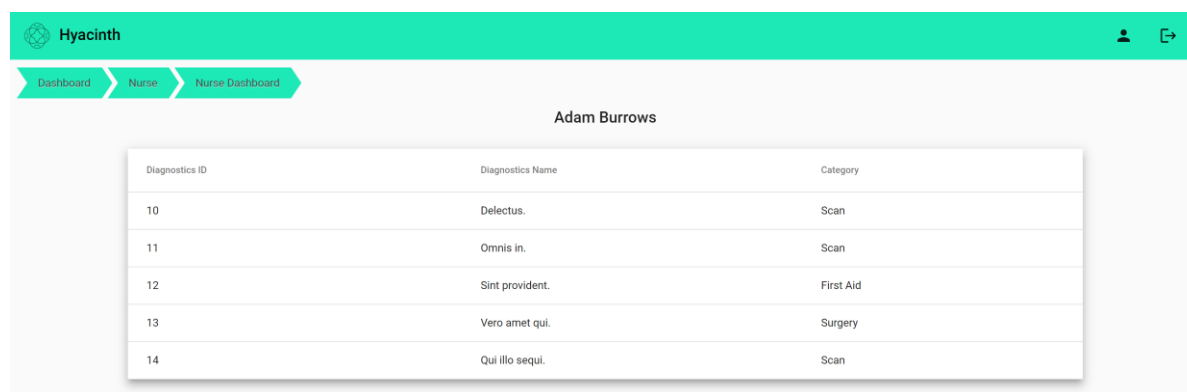
Consultation ID	Problem	Doctor ID	Doctor Name	Specialization
5	Nobis impedit quisquam. A.	2	Dana Bethea	Family Practice
6	Culpa saepe facere.	4	Adan May	Family Practice
7	Dolor sit sint. Repellat quia.	10	Antone Coles	Cardiology

Click on a row directs the user to the [Consultation Details](#)Page.

## Nurse

The nurse's dashboard contains all the diagnostics done by them.

Figure 13: Nurse Dashboard



Hyacinth

Dashboard > Nurse > Nurse Dashboard

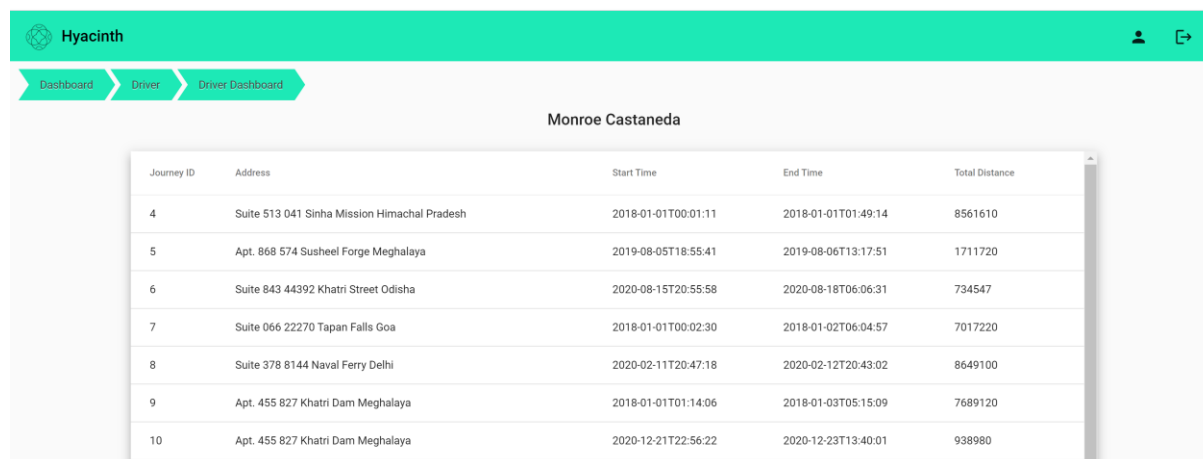
Adam Burrows

Diagnostics ID	Diagnostics Name	Category
10	Delectus.	Scan
11	Omnis in.	Scan
12	Sint provident.	First Aid
13	Vero amet qui.	Surgery
14	Qui illo sequi.	Scan

## Driver

The driver's dashboard contains all the journeys done by that driver

Figure 14: Driver Dashboard



Hyacinth

Dashboard > Driver > Driver Dashboard

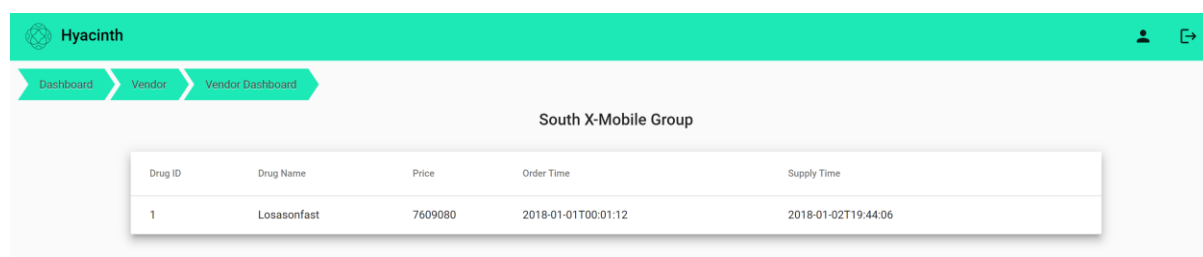
Monroe Castaneda

Journey ID	Address	Start Time	End Time	Total Distance
4	Suite 513 041 Sinha Mission Himachal Pradesh	2018-01-01T00:01:11	2018-01-01T01:49:14	8561610
5	Apt. 868 574 Susheel Forge Meghalaya	2019-08-05T18:55:41	2019-08-06T13:17:51	1711720
6	Suite 843 44392 Khatri Street Odisha	2020-08-15T20:55:58	2020-08-18T06:06:31	734547
7	Suite 066 22270 Tapan Falls Goa	2018-01-01T00:02:30	2018-01-02T06:04:57	7017220
8	Suite 378 8144 Naval Ferry Delhi	2020-02-11T20:47:18	2020-02-12T20:43:02	8649100
9	Apt. 455 827 Khatri Dam Meghalaya	2018-01-01T01:14:06	2018-01-03T05:15:09	7689120
10	Apt. 455 827 Khatri Dam Meghalaya	2020-12-21T22:56:22	2020-12-23T13:40:01	938980

## Vendor

The vendor's dashboard contains all the drugs sold by that vendor to this hospital.

Figure 15: Vendor Dashboard



Hyacinth

Dashboard > Vendor > Vendor Dashboard

South X-Mobile Group

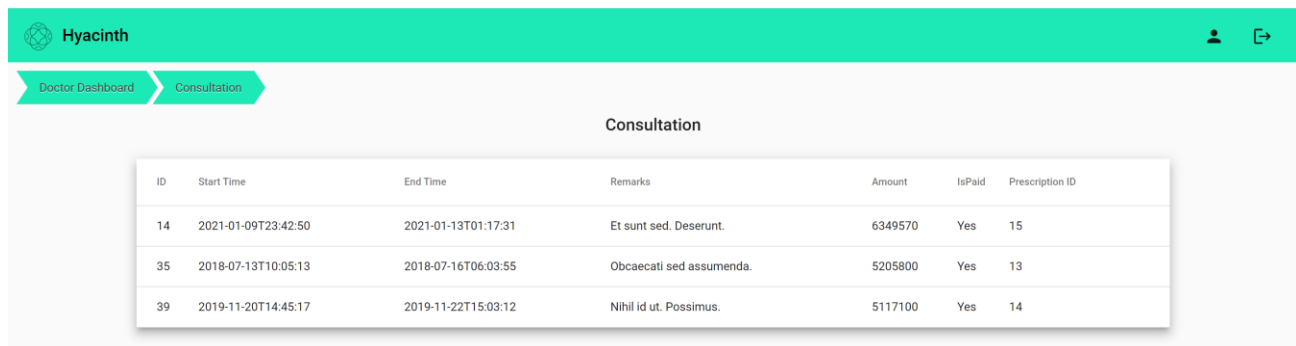
Drug ID	Drug Name	Price	Order Time	Supply Time
1	Losasonfast	7609080	2018-01-01T00:01:12	2018-01-02T19:44:06

## Detailed Pages

### Consultation Details

The consultations page shows all the appointments pertaining to this consultation.

Figure 16: Consultation Details



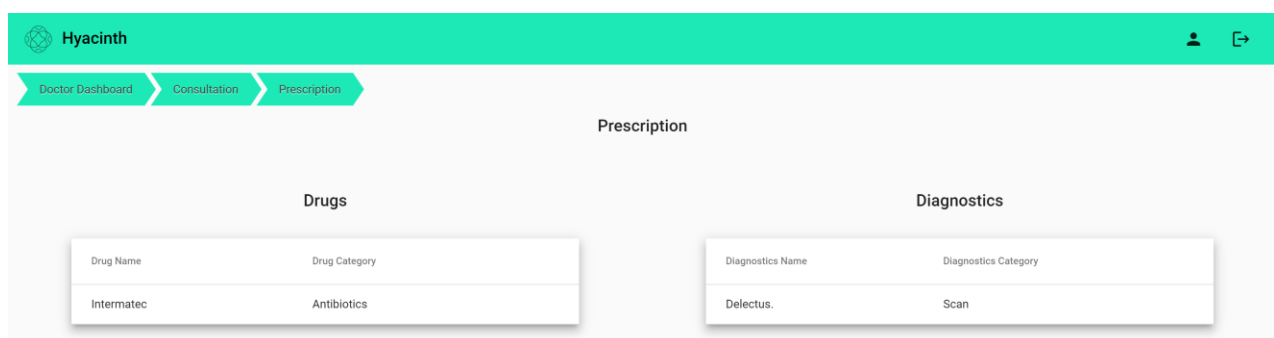
ID	Start Time	End Time	Remarks	Amount	IsPaid	Prescription ID
14	2021-01-09T23:42:50	2021-01-13T01:17:31	Et sunt sed. Deserunt.	6349570	Yes	15
35	2018-07-13T10:05:13	2018-07-16T06:03:55	Obcaecat sed assumenda.	5205800	Yes	13
39	2019-11-20T14:45:17	2019-11-22T15:03:12	Nihil id ut. Possimus.	5117100	Yes	14

Clicking on an appointment directs the user to the [Prescription details](#) attached to this appointment.

### Prescription details

The prescription page shows the details of the prescription.

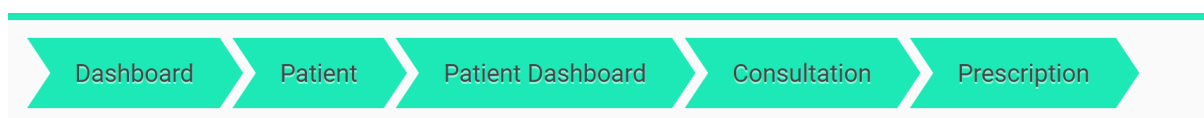
Figure 17: Prescription Details



Drugs		Diagnostics	
Drug Name	Drug Category	Diagnostics Name	Diagnostics Category
Intermatec	Antibiotics	Delectus.	Scan

Every prescription has two categories: Drugs, and Diagnostics. The tables corresponding to these categories are shown above.

Note: As we move pages deep, the problem one may face is navigation across page. This is handled by CSS breadcrumbs as shown below.



Dashboard	Patient	Patient Dashboard	Consultation	Prescription
-----------	---------	-------------------	--------------	--------------

Clicking on any breadcrumb leads to that specific page, thus making it elegant to navigate.

## Profile Page

Every user has their own profile pages. It contains all their personal information as shown below. The view/edit can be toggled using the edit button in the top right. Note: Editing requires the confirmation of password again.

View Mode	Edit Mode
<div><h3>Profile</h3><div><div>Name Dana Bethea</div><div>Address Railway Arch 27C</div><div>Phone +91 6276875343</div><div>Sex: <input checked="" type="radio"/> Male <input type="radio"/> Female <input type="radio"/> Other</div><div>Email Nance@nowhere.com</div></div></div>	<div><div><div>Name * Dana Bethea</div><div>Address * Railway Arch 27C</div><div>Phone * +91 6276875343</div><div>Sex: <input checked="" type="radio"/> Male <input type="radio"/> Female <input type="radio"/> Other</div><div>Email * Nance@nowhere.com</div><div>Password *</div><div>Confirm Password *</div></div><div><div>Clear</div><div>Submit</div></div></div>



## Contributions

---

Hyacinth – hospital management website was done by four developers.

S.NO	Name	Roll Number
1	Yogesh R	111801047
2	Subhash S	111801042
3	Alisetti Sai Vamsi	111801002
4	KM Raswanth	111801056

Me, Yogesh R predominantly worked on making APIs using FastAPI that gives scripts to MariaDB and converts it into json. I made around five APIs for each role and some general APIs to integrate the same. I also worked on developing the ER/Schema diagram, wrote multiple views, functions, and procedures. In the frontend, I made dashboard component and breadcrumbs for navigation. For ease of development, I made the docker-compose file for backend and database.

## Appendix

---

1. [Proof of implementation](#)
2. [Entity Relationship Diagram](#)
3. [Schema Diagram](#)
4. [Schema creation](#)
5. [Data population](#)
6. [View Creation](#)
7. [Functions](#)
8. [Procedures](#)
9. [Backup](#)
  - a. [With data](#)
  - b. [Without data](#)
10. [Complete Source Code](#)