CS39202

DBMS Lab

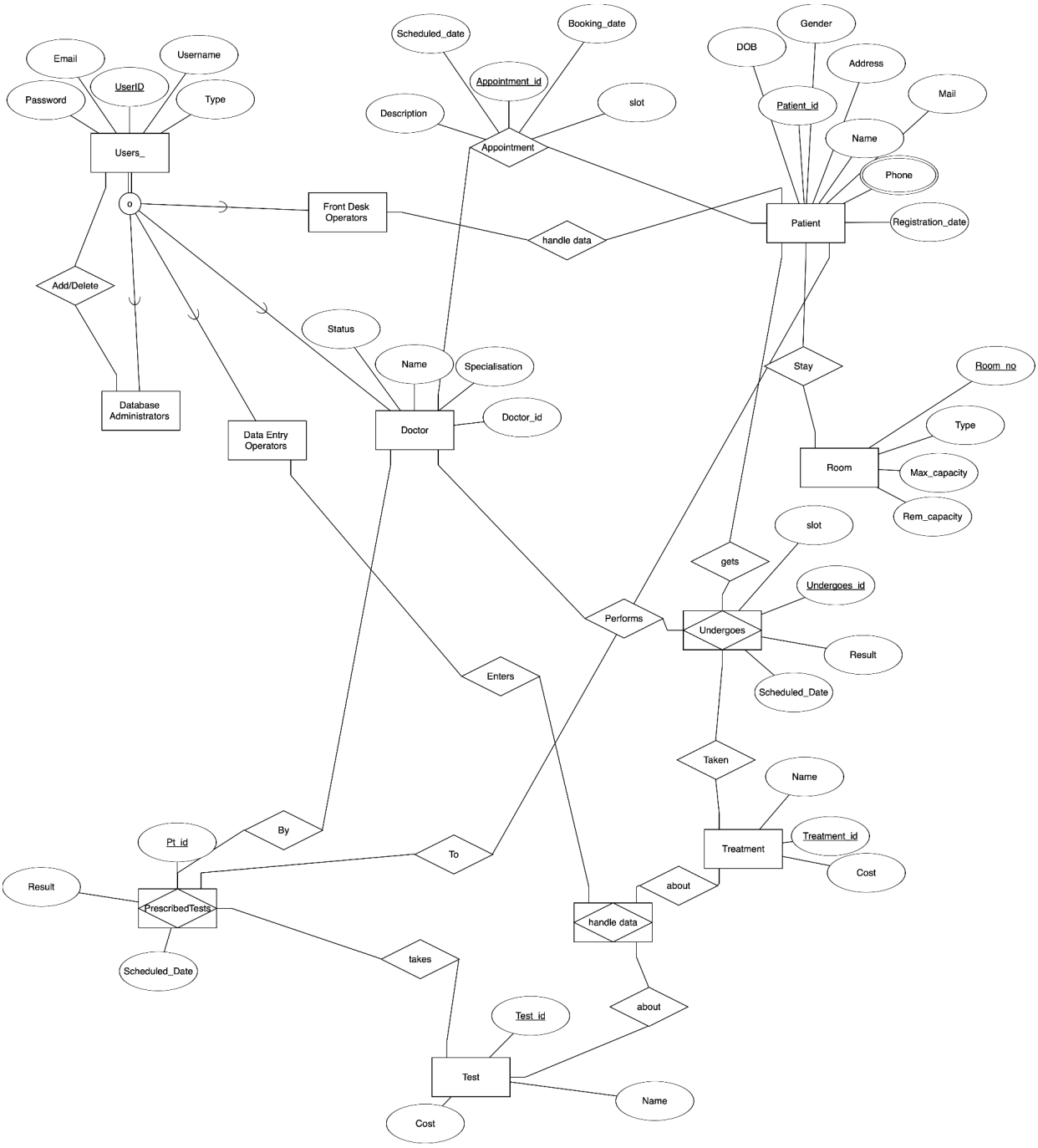
Mini Project

Project Report

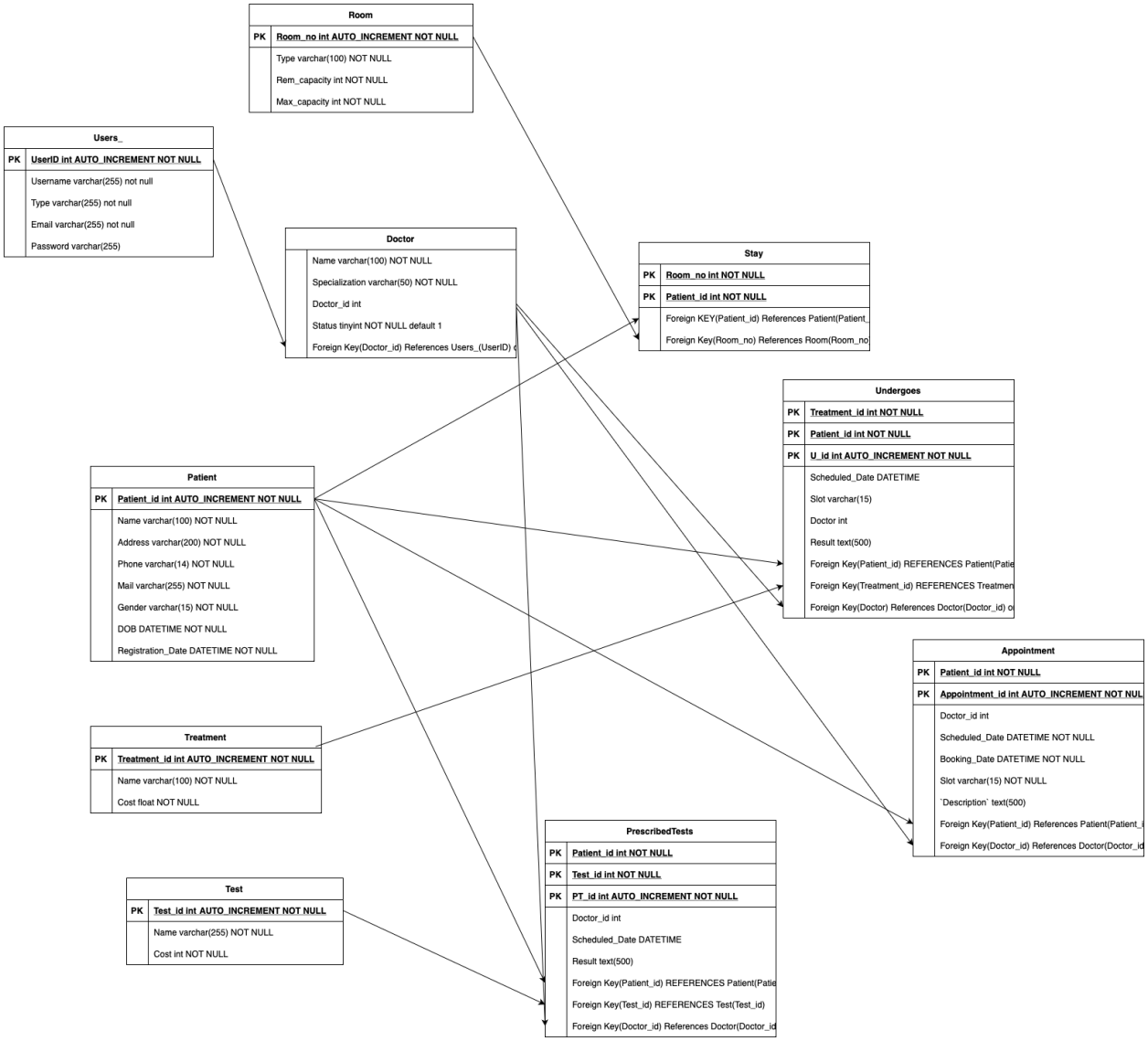
Group-Info

Kunchala Chandra Sekhara Azad- 20CS10032
Budda Mohan Chandu - 20CS10017
Varanasi Vachan Siddarth - 20CS10070
Daravath Yuvraj - 20CS10025
Vemuri Harsha - 20CS10071

ER Diagram:


Email
Username
Password
UserID
Type
Users_
o
Add/Delete
Database Administrators
Data Entry Operators
Front Desk Operators
Scheduled_date
Booking_date
Appointment_id
Description
slot
Appointment
handle data
Status
Name
Specialisation
Doctor_id
Doctor
DOB
Gender
Address
Mail
Patient_id
Name
Phone
Patient
Registration_date
Stay
Room_no
Type
Max_capacity
Rem_capacity
Room
slot
gets
Undergoes_id
Performs
Undergoes
Result
Scheduled_Date
Enters
Taken
Name
Treatment_id
Treatment
Cost
about
handle data
Pt_id
By
To
Result
PrescribedTests
Scheduled_Date
takes
Test_id
about
Test
Name
Cost

Relational Schema:



Languages and Tools Used:

JavaScript:

This is the elemental language used in both front-end and back-end development.Specifically it was utilized because of its vast library support .

NodeJS:

This tool is used in back-end development. It is a runtime environment used for building server side application.Specifically it was utilized for server handling.

ExpressJS:

This tool is used in back-end development. It is a web application framework for NodeJS.Specifically it was utilized for server routing.

ReactJS:

This tool is used in front-end development.It is a JavaScript library for building interfaces. Specifically it was utilized in routing , navigating and page component rendering.

HTML:

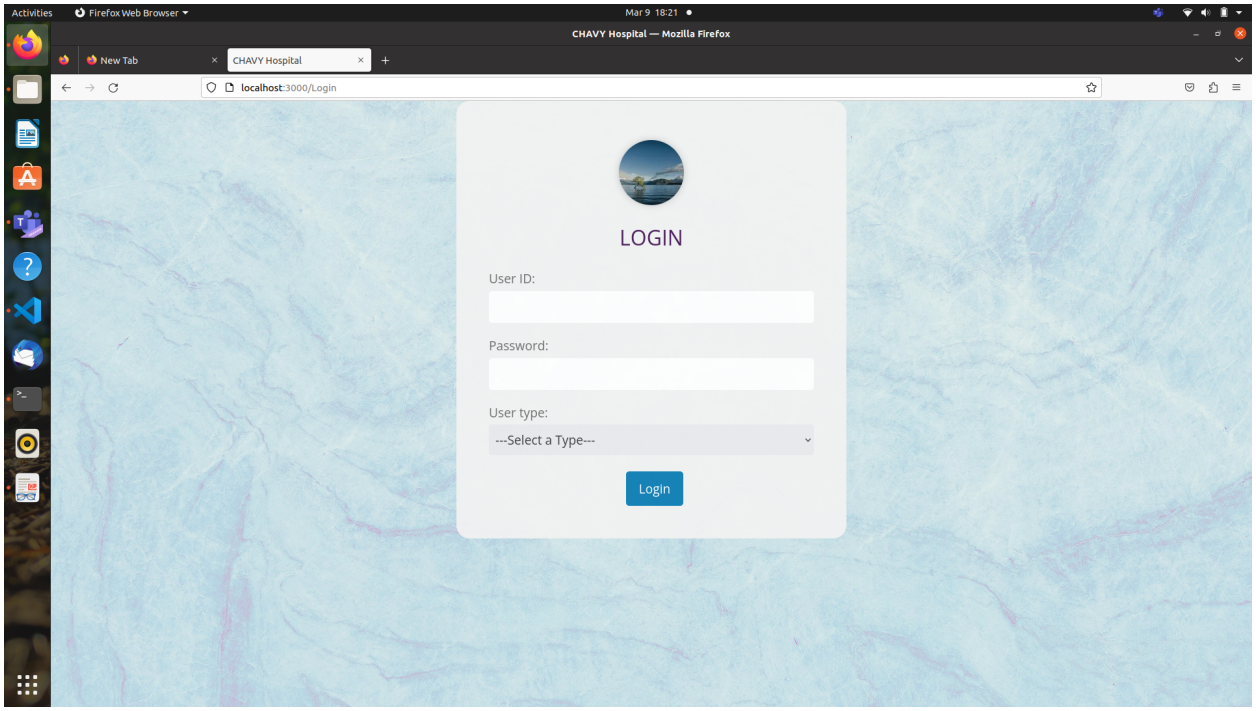
This language is used in front-end development.Specifically it was utilized in structuring pages for the application.

CSS:

This language is used in front-end development.Specifically it was utilized in styling the pages for in the application.

Triggers and Workflows:

Login:



This page prompts the person to enter his user id along with password and his type.

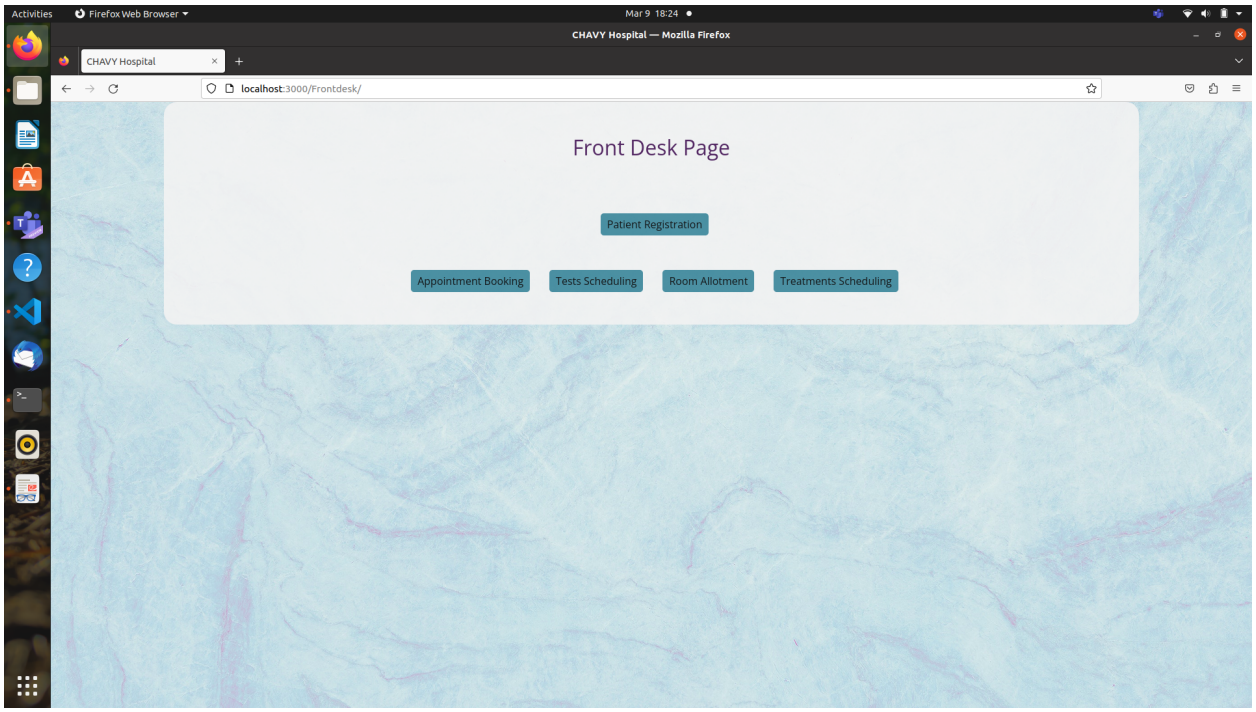
There are four types of users provided:

(1)Front Desk Operators
(2)Data Entry Operators
(3)Doctor
(4)Database Administrator

There are different functionalities to be implemented for each and are described as such below:

(i) Front Desk Operator:

Home Page:

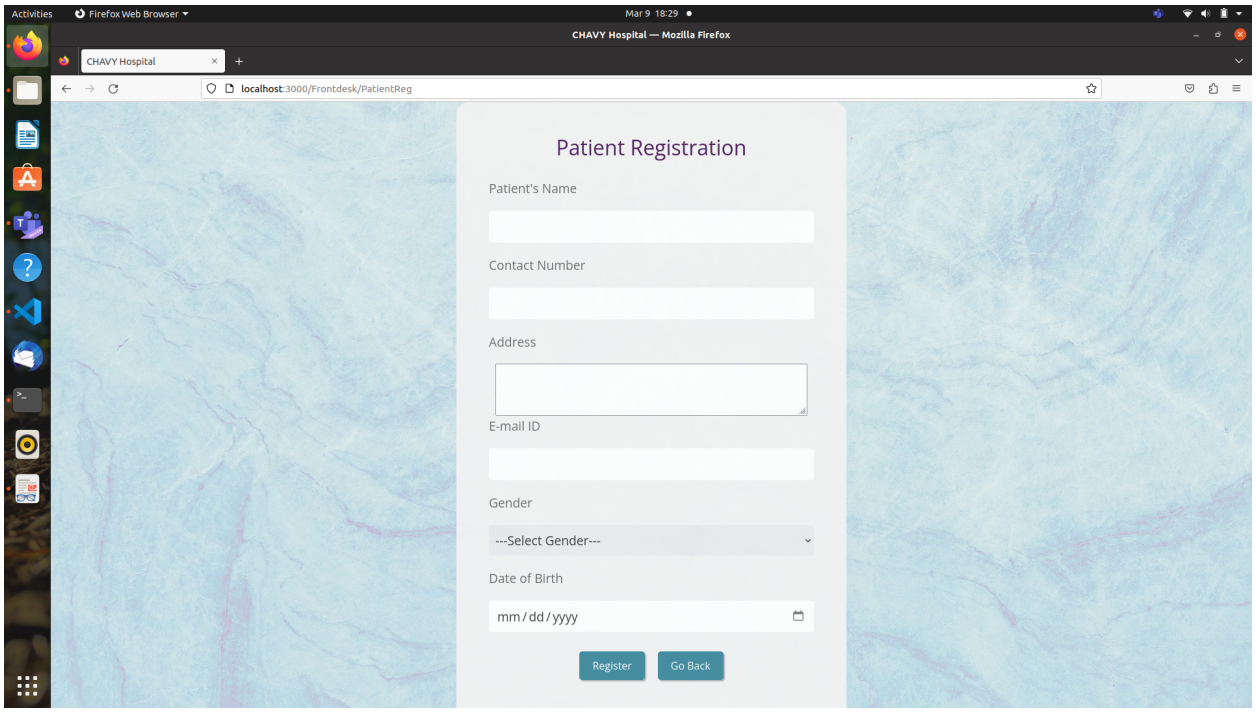


All the functions(listed below)of a frontdesk operator are in the home page. They are :

Patient Registration
Appointment Booking
Test Scheduling
Room Allotment
Treatments Scheduling

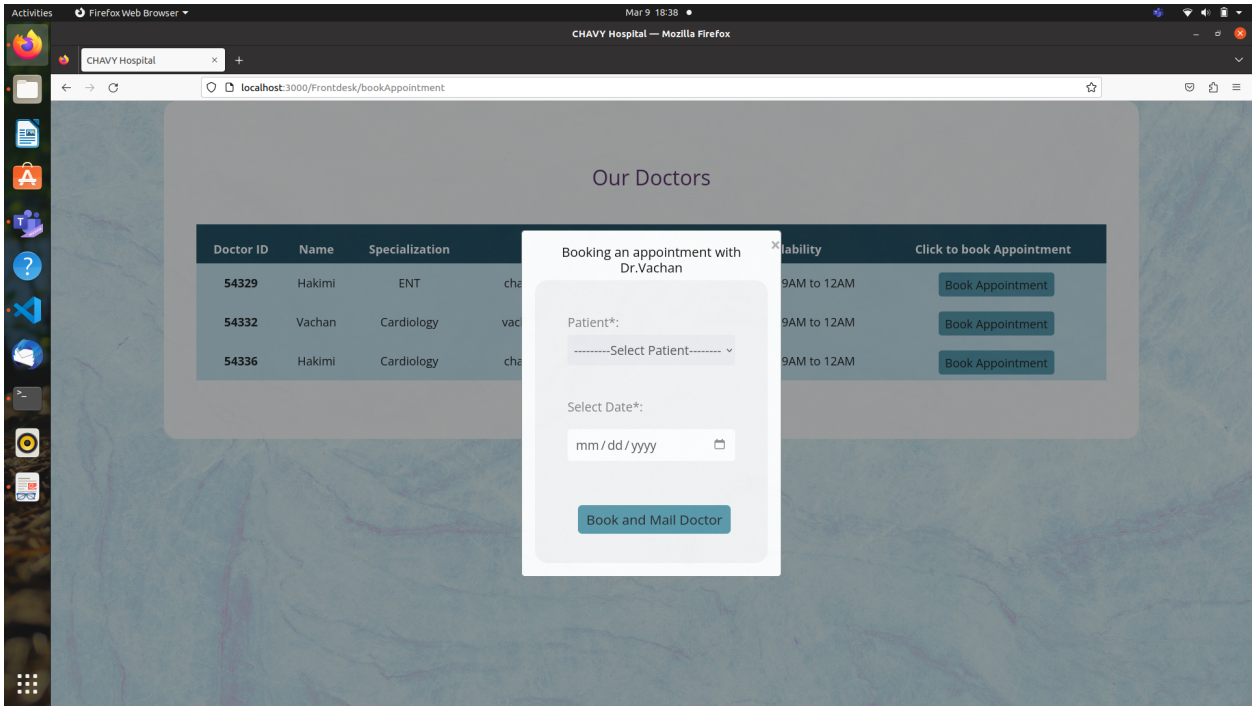
All these are functionalities in response to a patient's request at frontdesk.

Patient Registration:



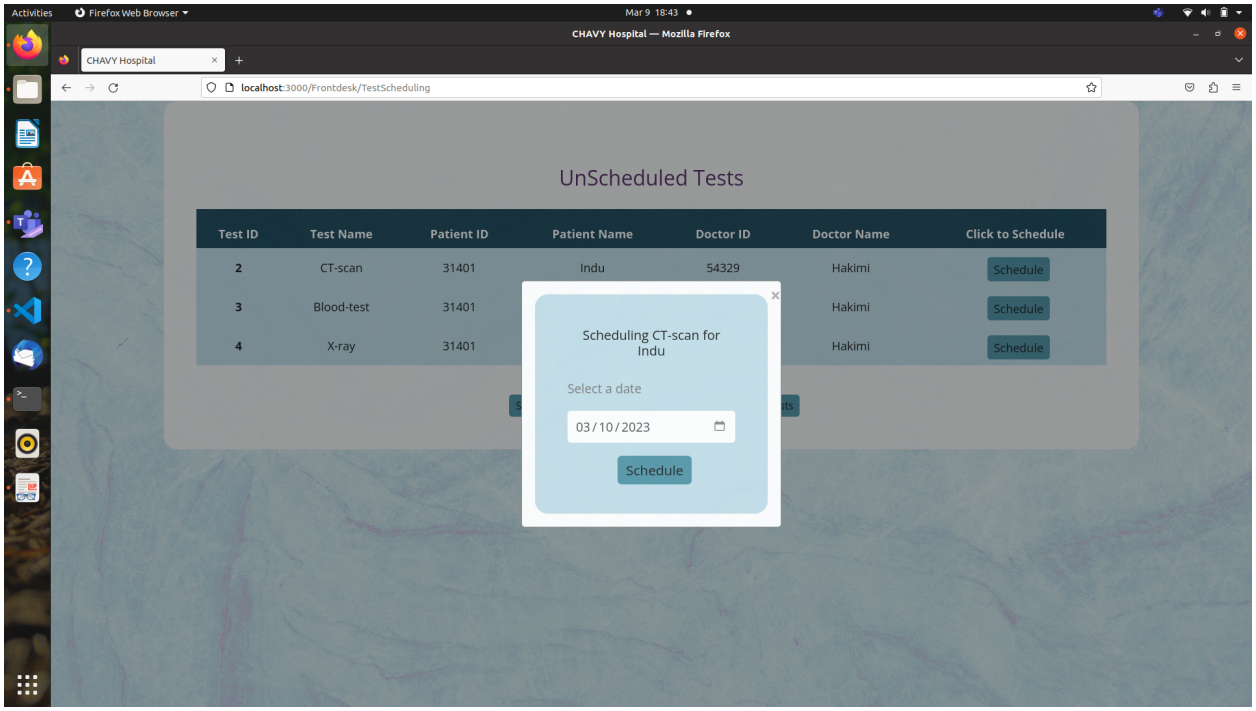
A new Patient is to be enrolled by their mentioned details by the operator prior to scheduling any activities or booking an appointment.

Appointment Booking:



Appointment to be booked by the operator with the doctor listed for the selected patient and slot. This will also direct an email to the doctor as well as the patient regarding the appointment and its schedule.

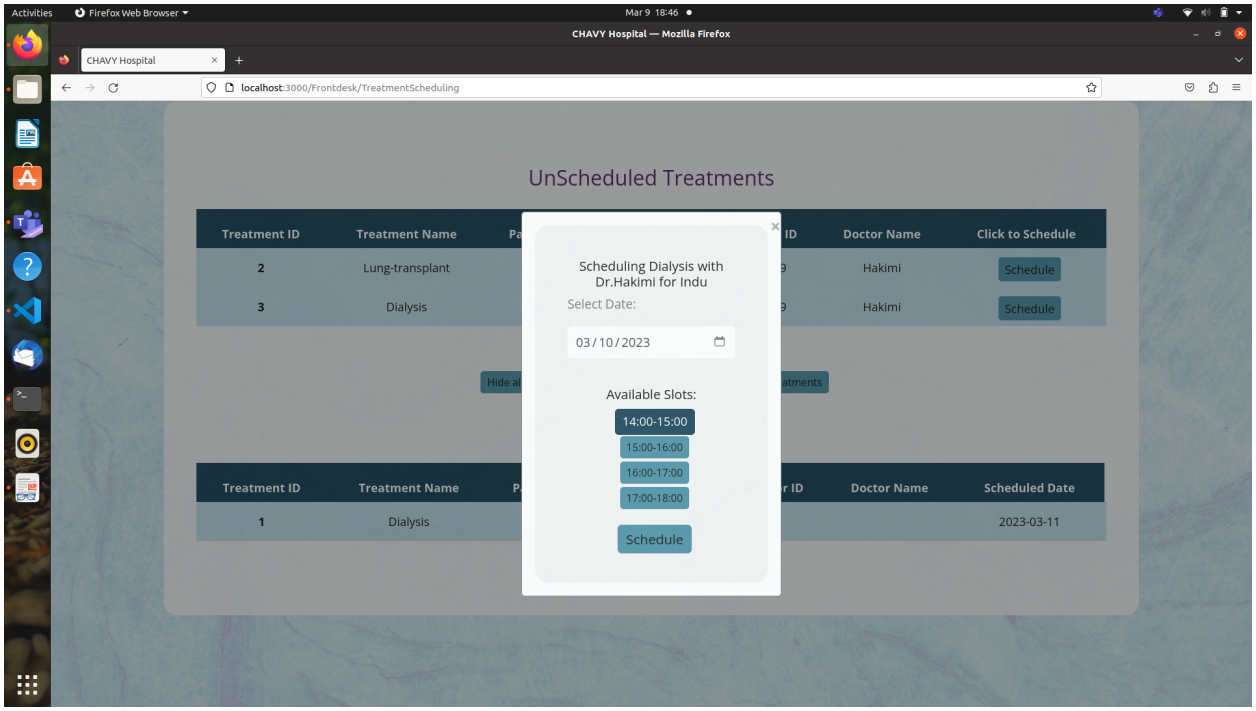
Test Scheduling:



When a doctor prescribes a test for a particular patient , it is the front desk operator’s duty to schedule the test and has to select the date to schedule the test.

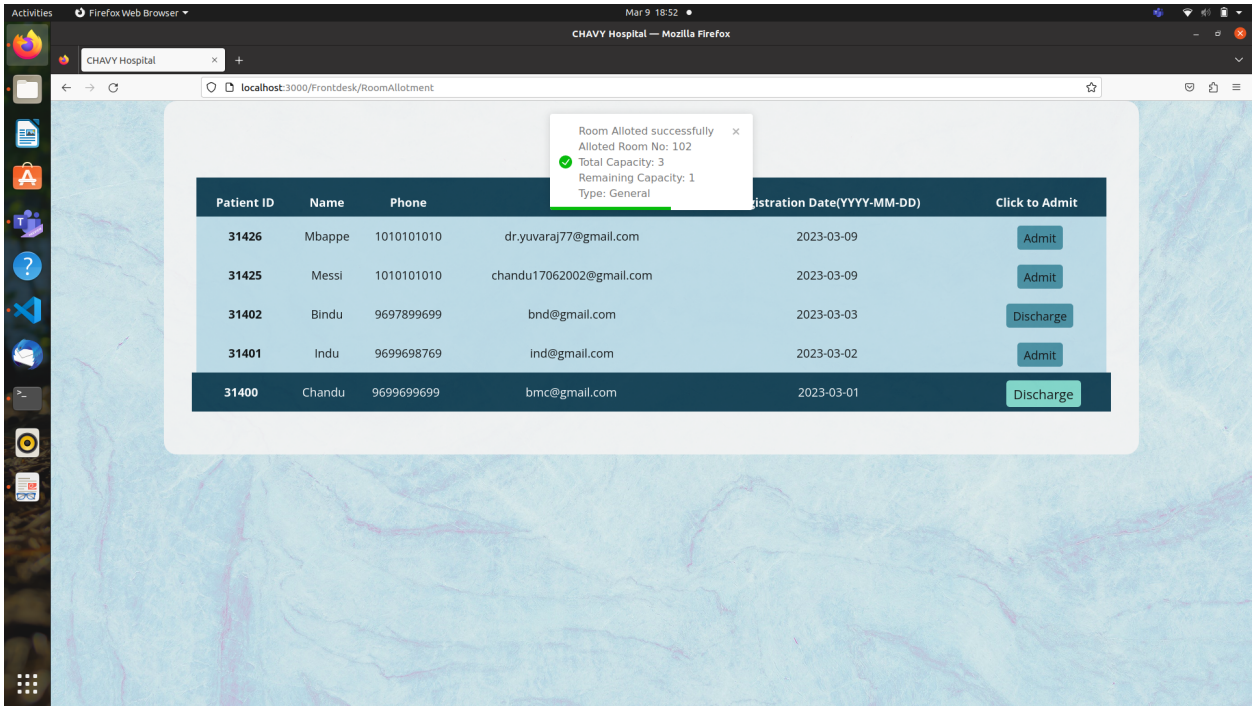
Treatments Scheduled:

When a doctor prescribes a treatment for a particular patient , it is the front desk operator’s duty to schedule the treatment and has to select the schedule for the treatment.



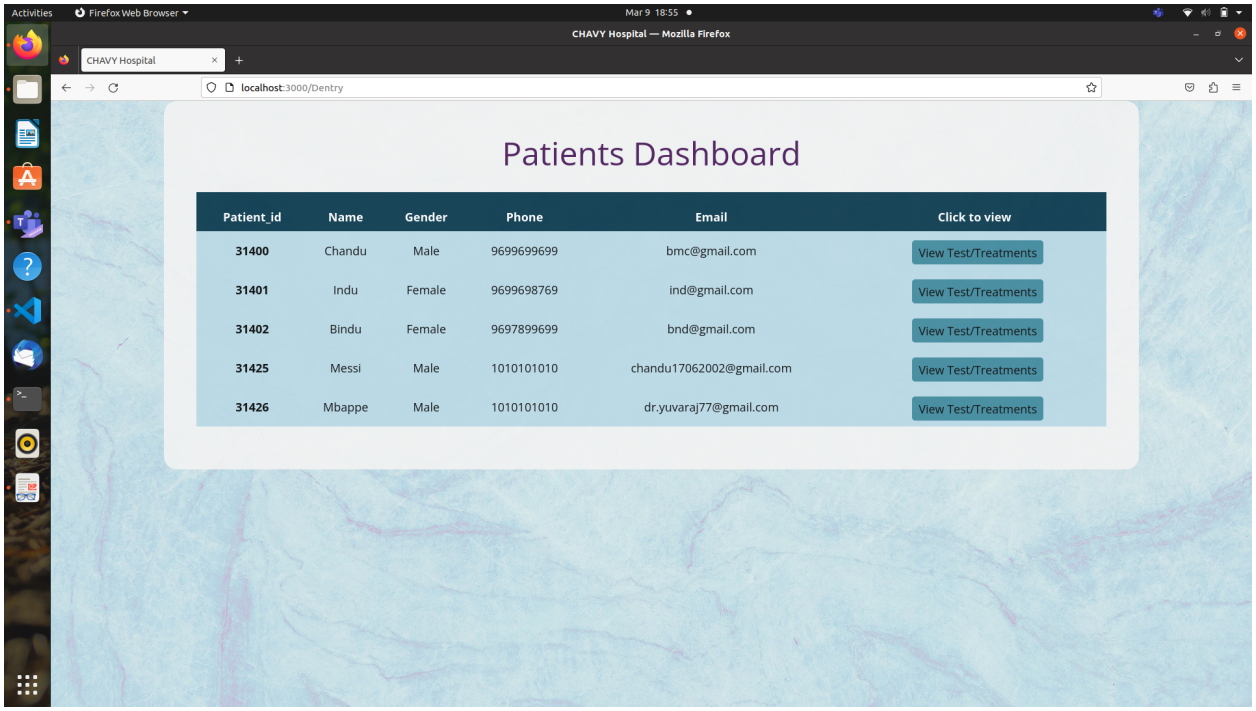
Room Allotment:

The rooms allottment to the patients are done here.



(ii) Data Entry Operator:

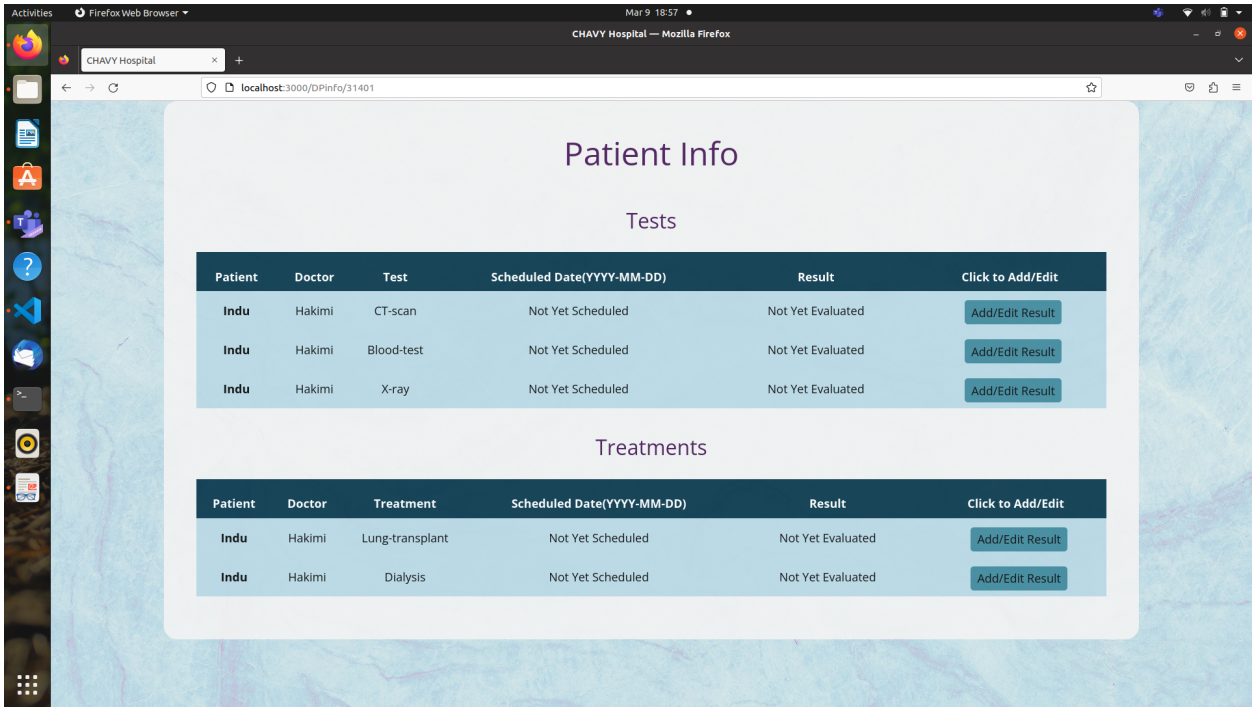
Home page:



All the Patients will be listed in the home page of the data entry operator. He/She can then update/edit the treatments and test results of the listed Patients.

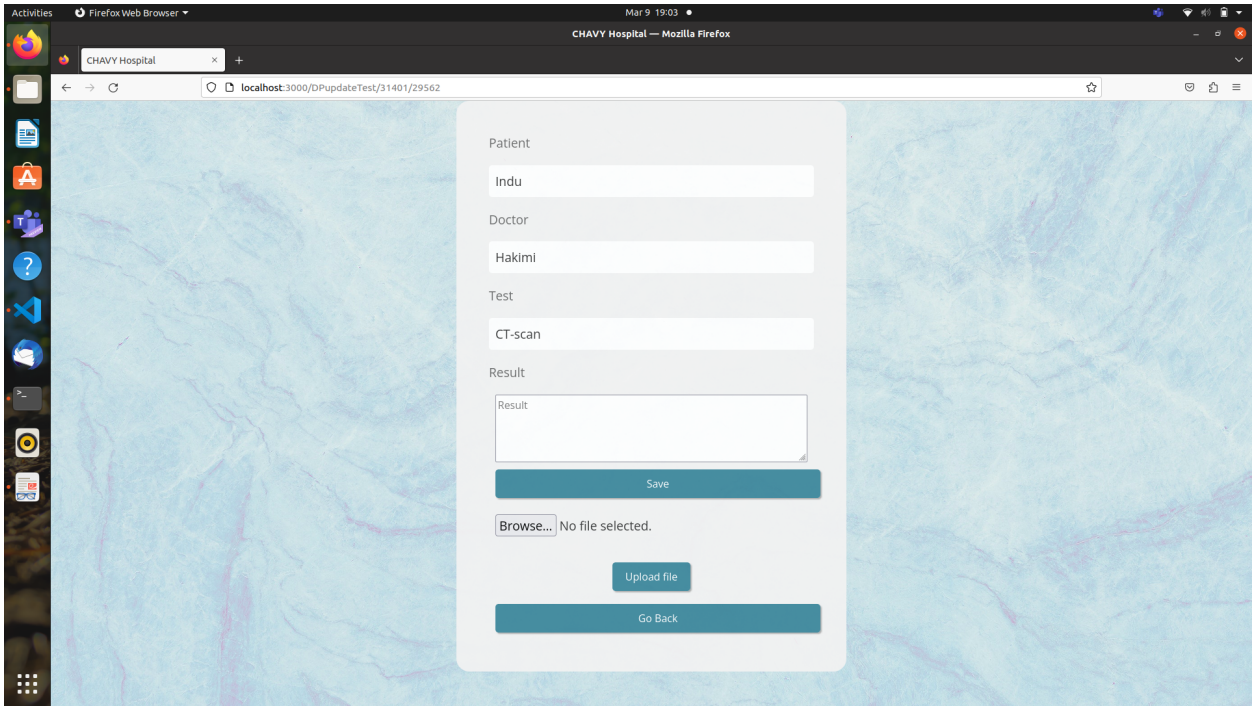
They are also entitled to view any previous records of the patients'.

View Patient:



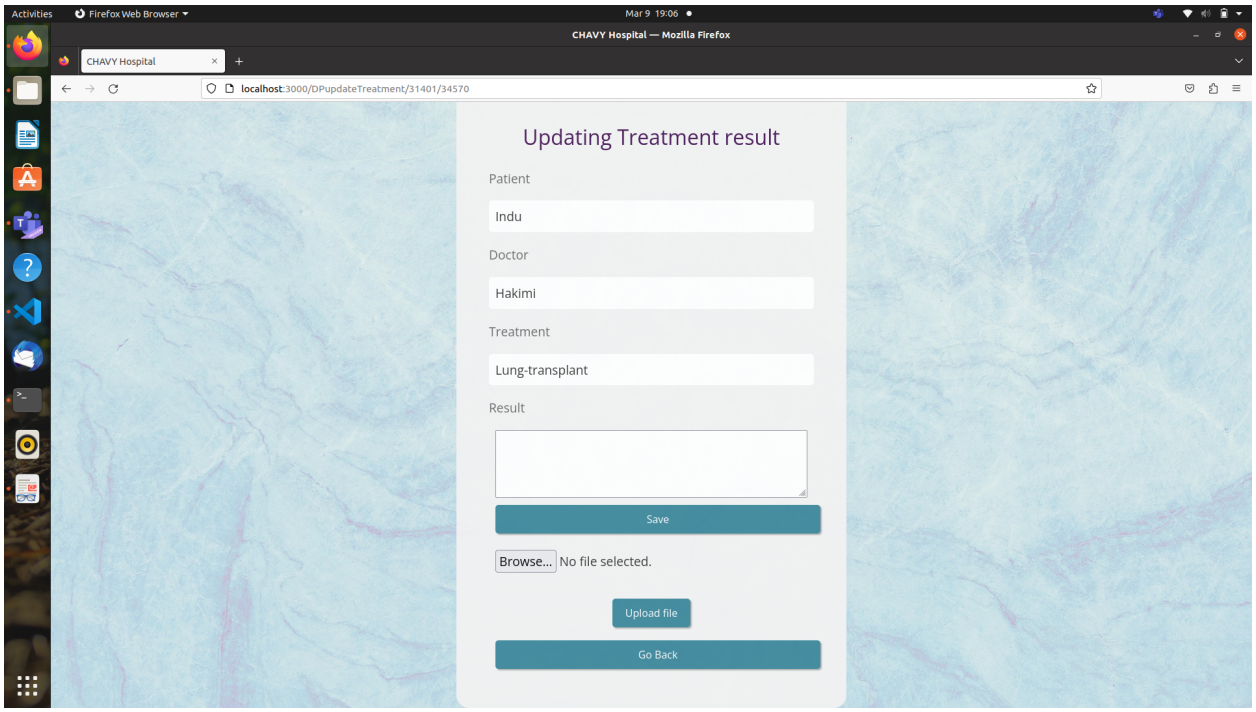
We can view the tests and treatments of a patient and can update the results of them accordingly.

Add/Edit Test:



For a particular patient we can provide the result of the tests they were prescribed here.

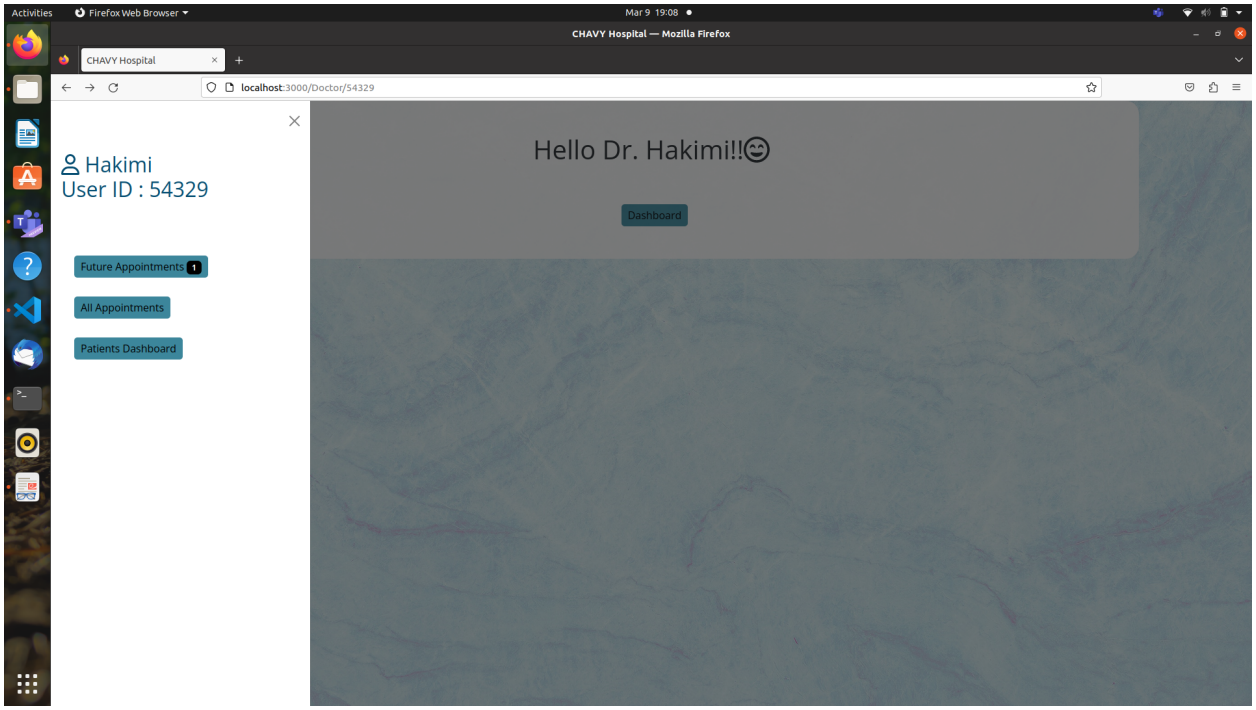
Add/Edit Treatment:



For a particular patient we can provide the result of the treatments they were prescribed here.

(iii) Doctor:

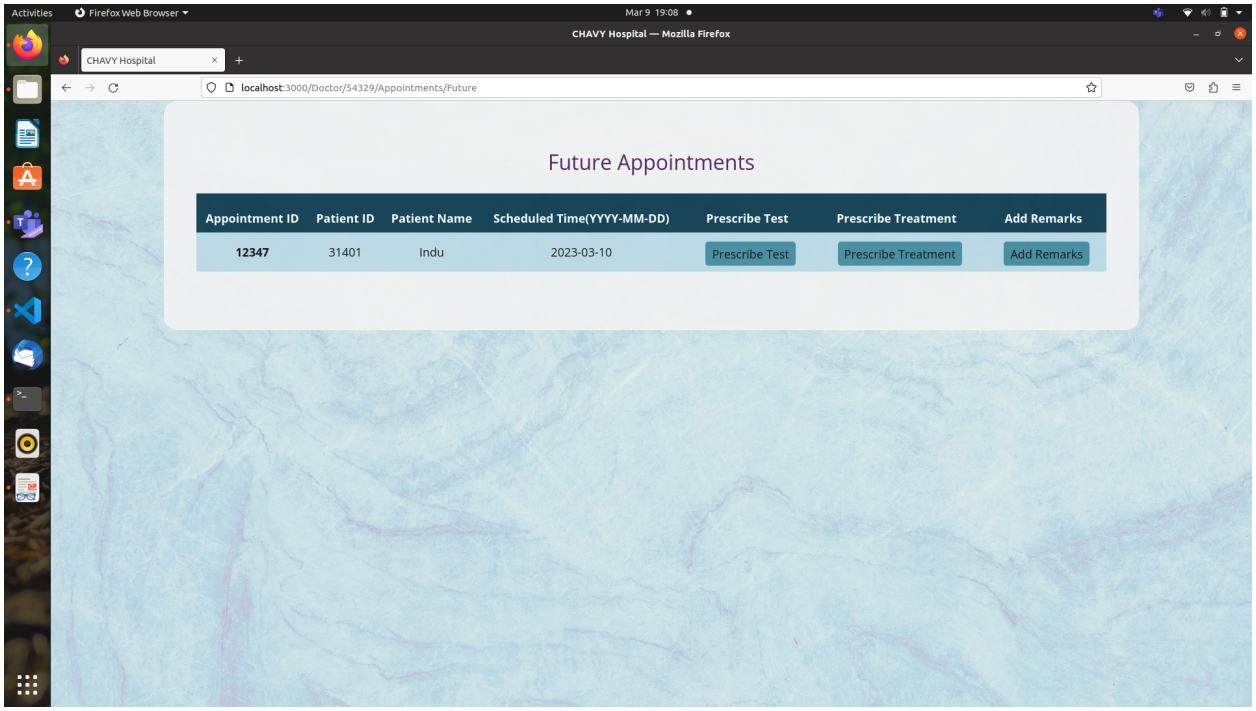
Home Page:



This page shows the credentials of the doctor along with his/her future appointments. All past appointments along with those scheduled for later and all the patients' dashboard.

The count of the Future appointments is also being mentioned here.

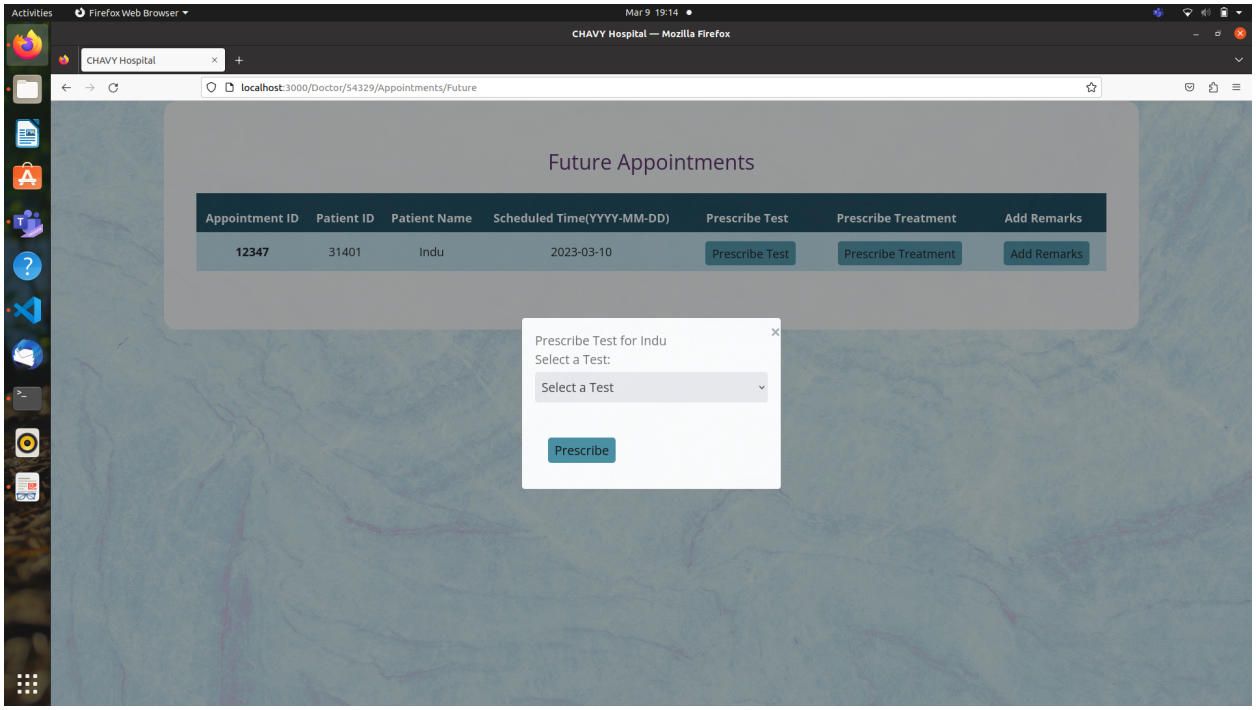
Future Appointments:



The list of all the future scheduled appointments are listed here. The doctor can prescribe a test or treatment prior or post the appointment.

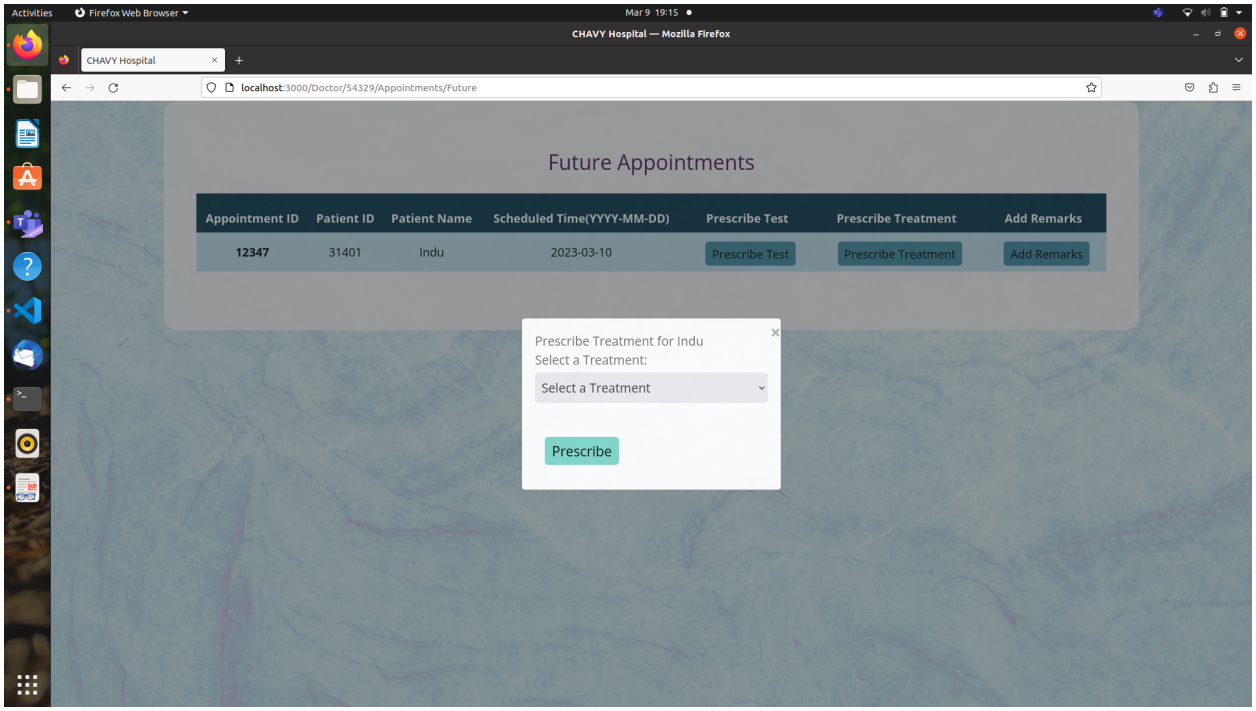
Prescribe Test:

The doctor can prescribe any test to his/her patients' from the list of tests available.



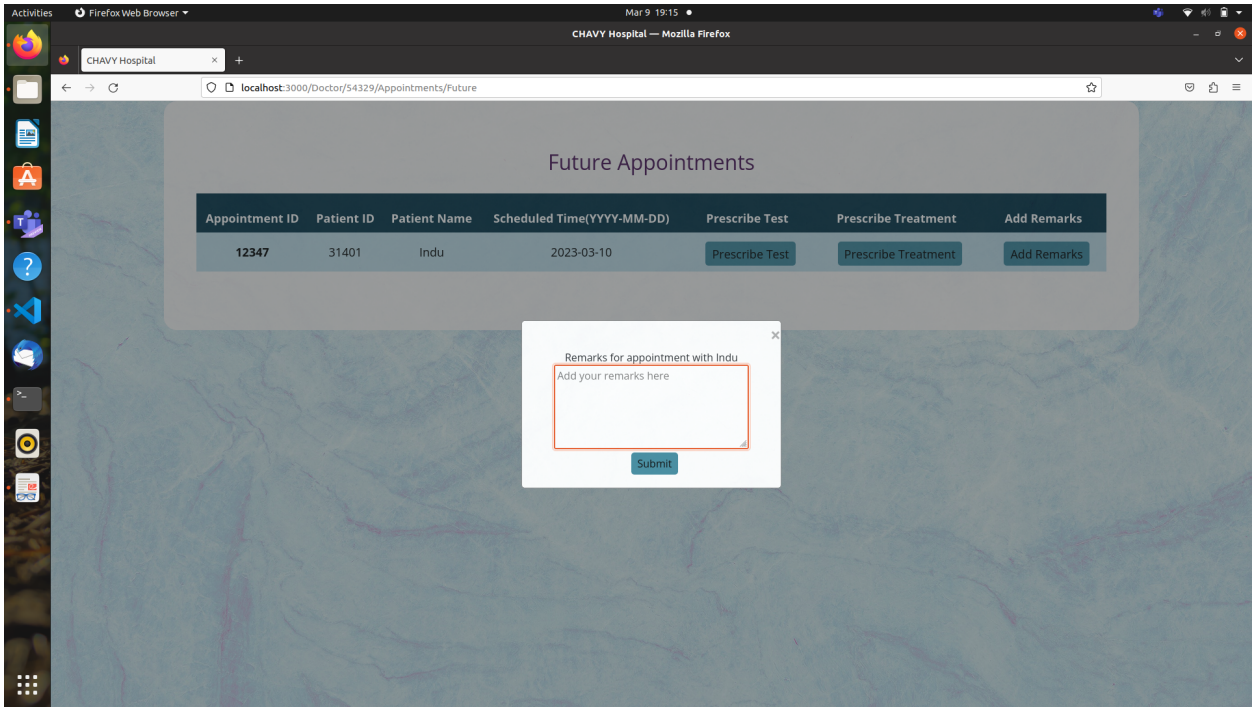
Prescribe Treatment:

The doctor can prescribe any treatment to his/her patients' from the list of treatments available.

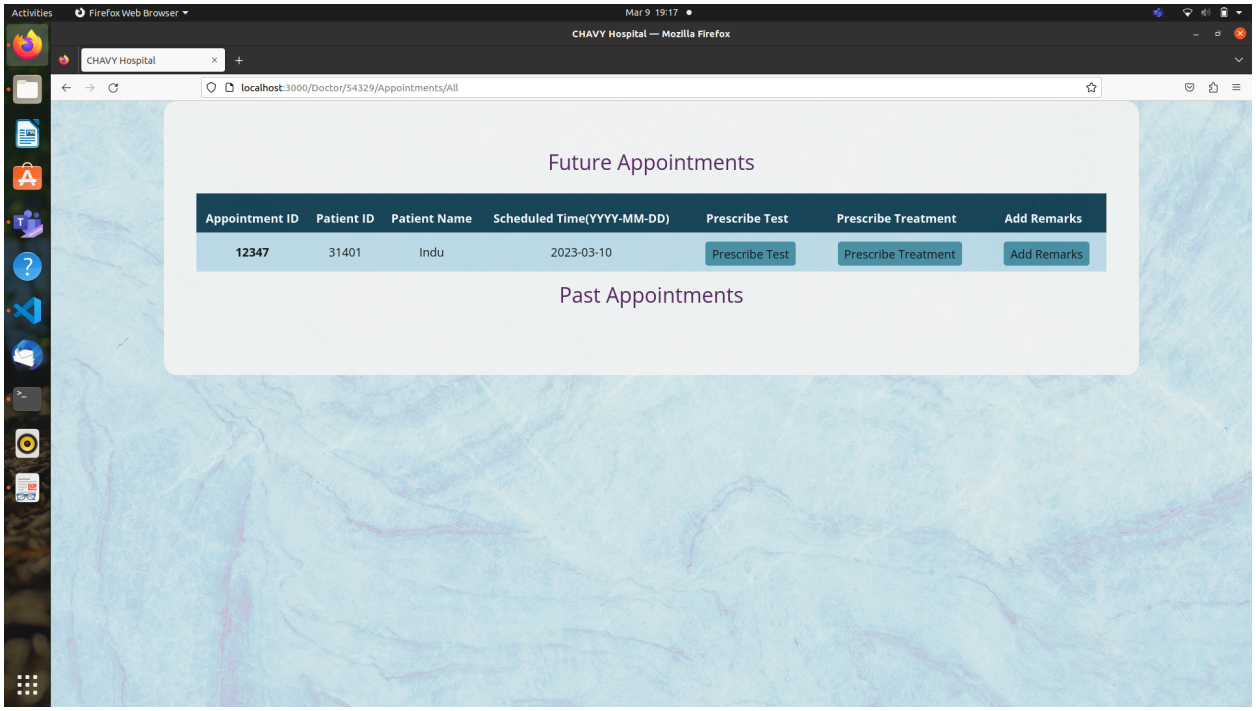


Add Remarks:

The doctor can shed some remarks post his appointment to the patient.

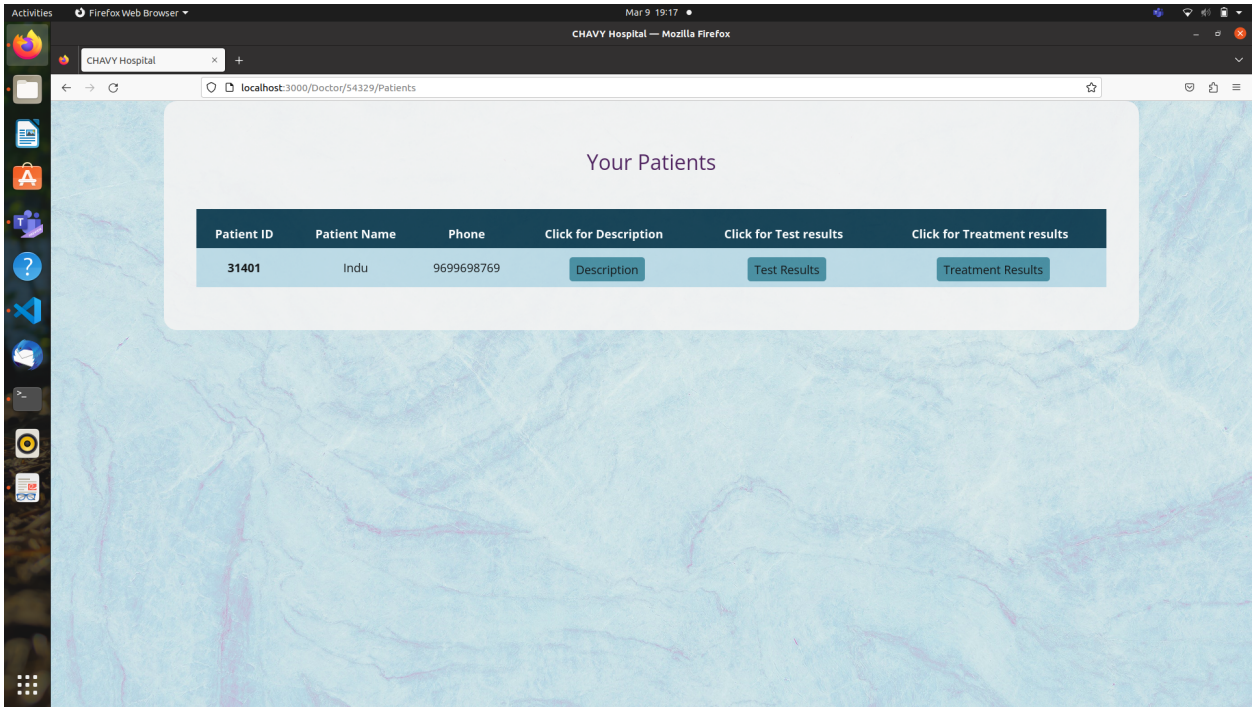


All Appointments Record:



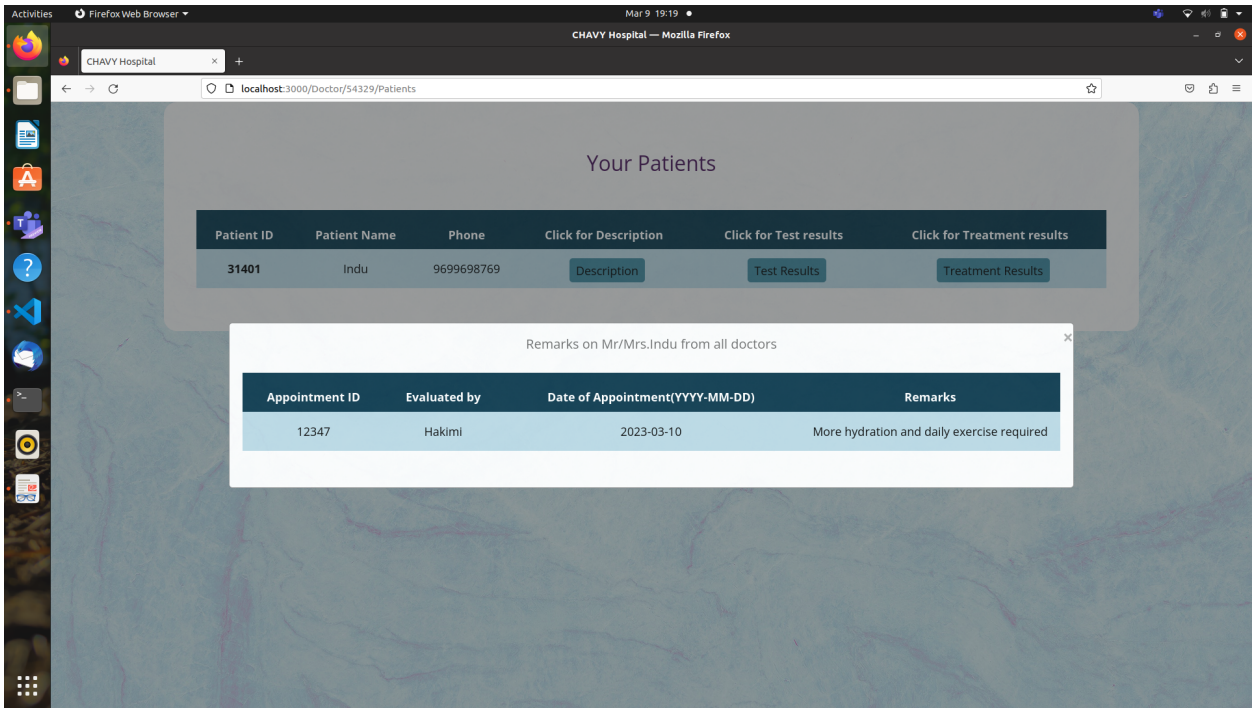
All the Appointments of the doctor are listed here partitioned as past and future appointments.

Patient’s Dashboard:



The doctor can maintain a brief description of the patient's condition in his/her patients’ dashboard. He can check the test and treatment results he/she prescribed.

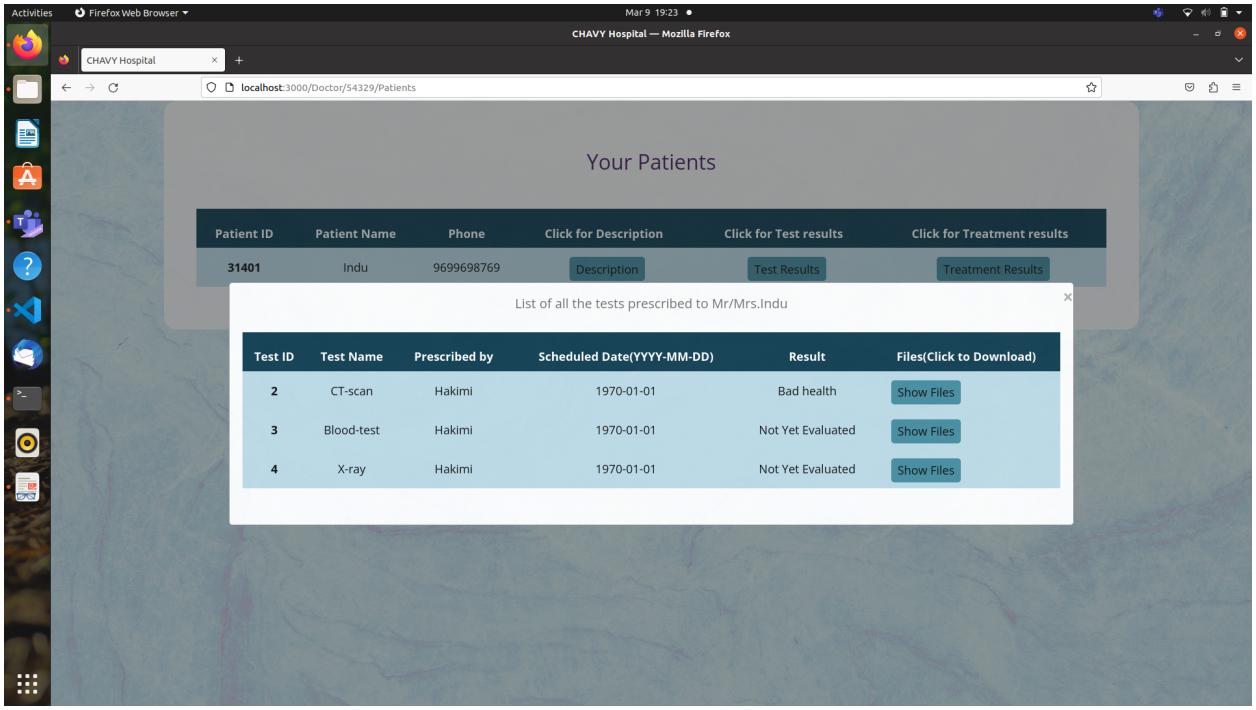
Patient Description:

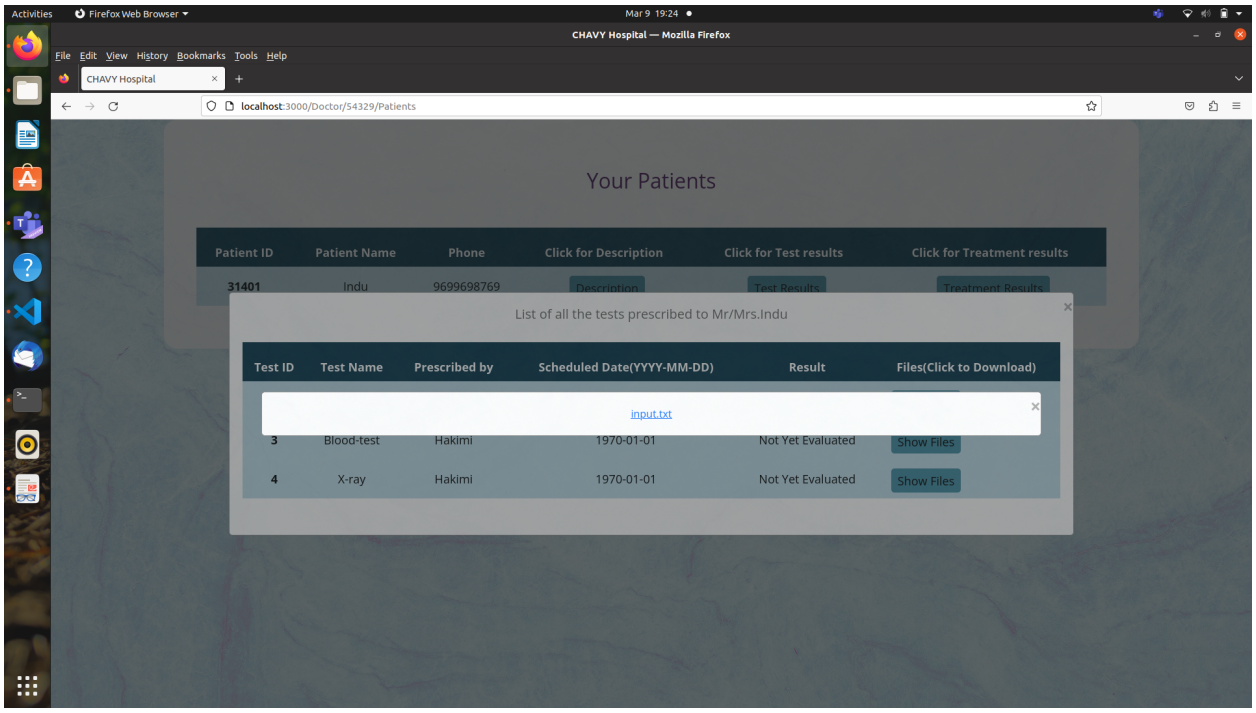


Here we allow doctor to add some remarks about the patient after his/her consultation for their future appointments and/or tests and treatments purpose. This also includes the appointments that are not by the same doctor as he/she needs to view the remarks given by any other doctor before continuing the treatment.

Patient's All Tests:

Here we are listing all the tests that particular patient was tested along with the date to keep track of the timeline of tests and the result of that test we also included a section to upload any files regarding the test (eg. x-ray copy , test report stats copy,etc.) .

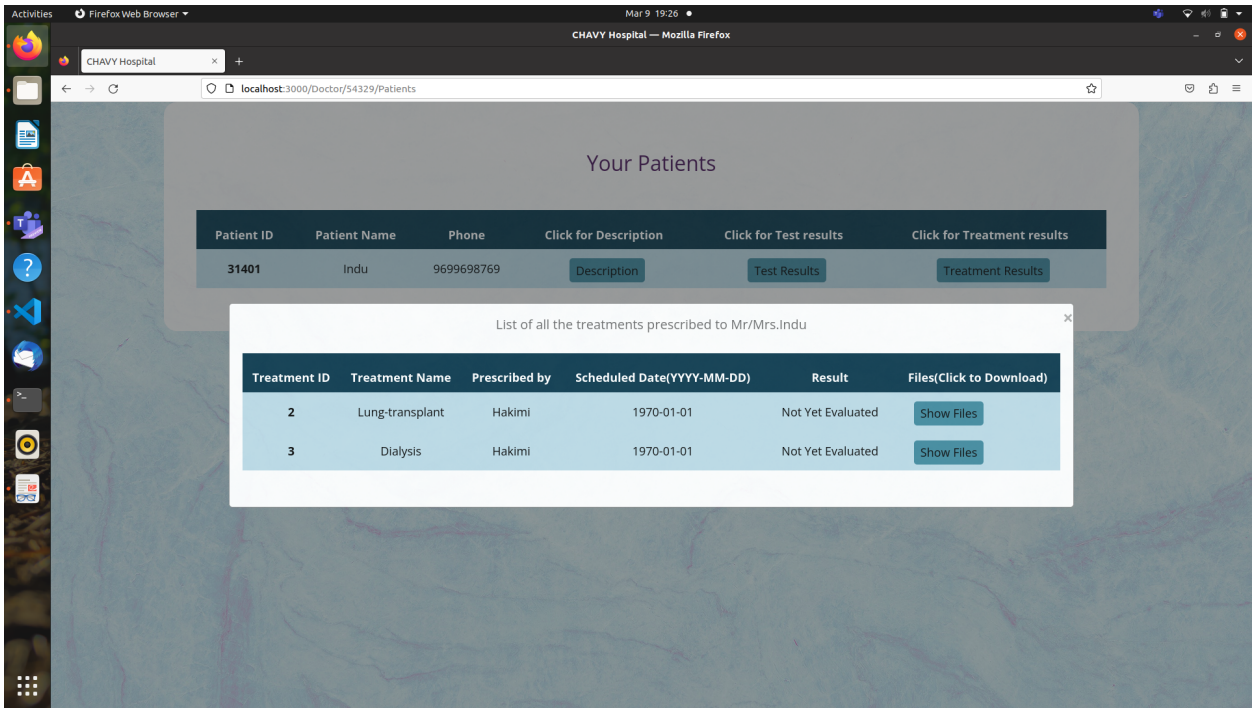

Activities
Firefox Web Browser
Mar 9 19:23
CHAVY Hospital — Mozilla Firefox
CHAVY Hospital
localhost:3000/Doctor/54329/Patients
Your Patients
Patient ID
Patient Name
Phone
Click for Description
Click for Test results
Click for Treatment results
31401
Indu
9699698769
Description
Test Results
Treatment Results
List of all the tests prescribed to Mr/Mrs.Indu
Test ID
Test Name
Prescribed by
Scheduled Date(YYYY-MM-DD)
Result
Files(Click to Download)
2
CT-scan
Hakimi
1970-01-01
Bad health
Show Files
3
Blood-test
Hakimi
1970-01-01
Not Yet Evaluated
Show Files
4
X-ray
Hakimi
1970-01-01
Not Yet Evaluated
Show Files


Activities
Firefox Web Browser
Mar 9 19:24
CHAVY Hospital — Mozilla Firefox
File Edit View History Bookmarks Tools Help
CHAVY Hospital
localhost:3000/Doctor/54329/Patients
Your Patients
Patient ID
Patient Name
Phone
Click for Description
Click for Test results
Click for Treatment results
31401
Indu
9699698769
List of all the tests prescribed to Mr/Mrs.Indu
Test ID
Test Name
Prescribed by
Scheduled Date(YYYY-MM-DD)
Result
Files(Click to Download)
input.txt
3
Blood-test
Hakimi
1970-01-01
Not Yet Evaluated
Show Files
4
X-ray
Hakimi
1970-01-01
Not Yet Evaluated
Show Files

Patient’s All Treatments:

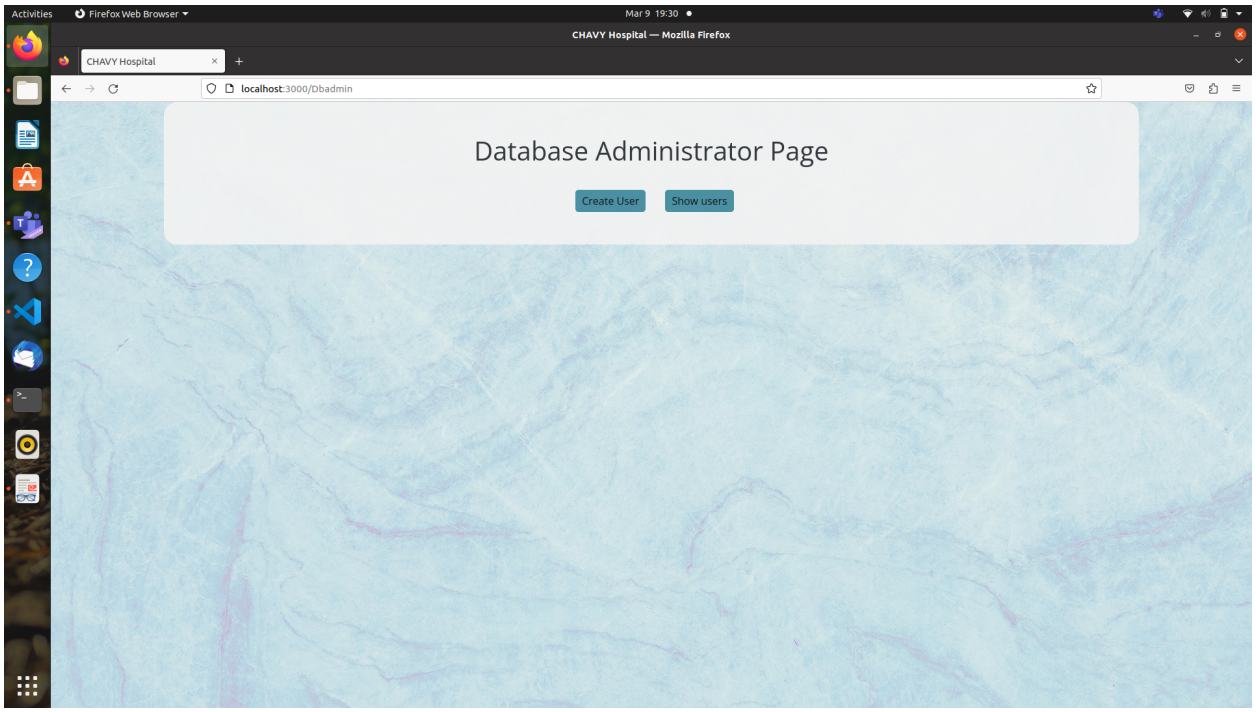
Here we are listing all the treatments that particular patient has undergone along with the date to keep track of the timeline of the treatment. We also provided the result section where we hove shown the result of the particular treatment after evaluation.

Similar to the tests section we have provided the feature to view the related files of the treatment . This helps doctor to get to know the stats of the patient prior and after the treatment



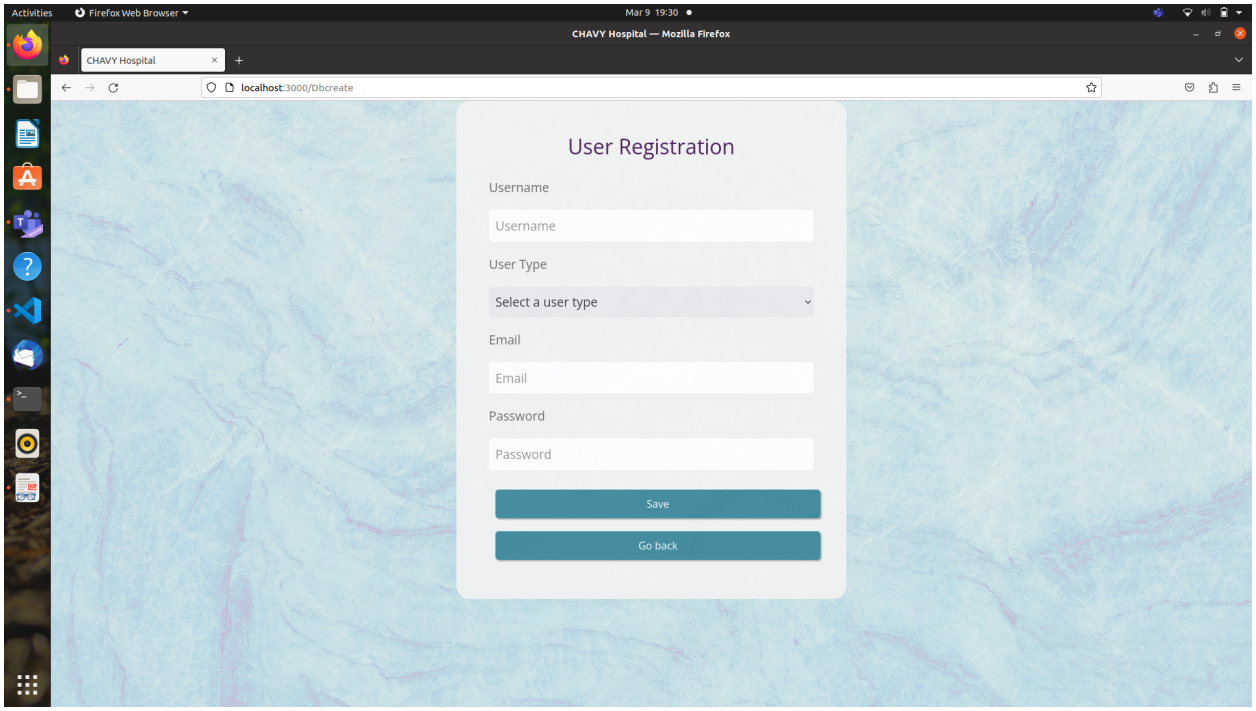
(iv) Database Administrator:

Home Page:



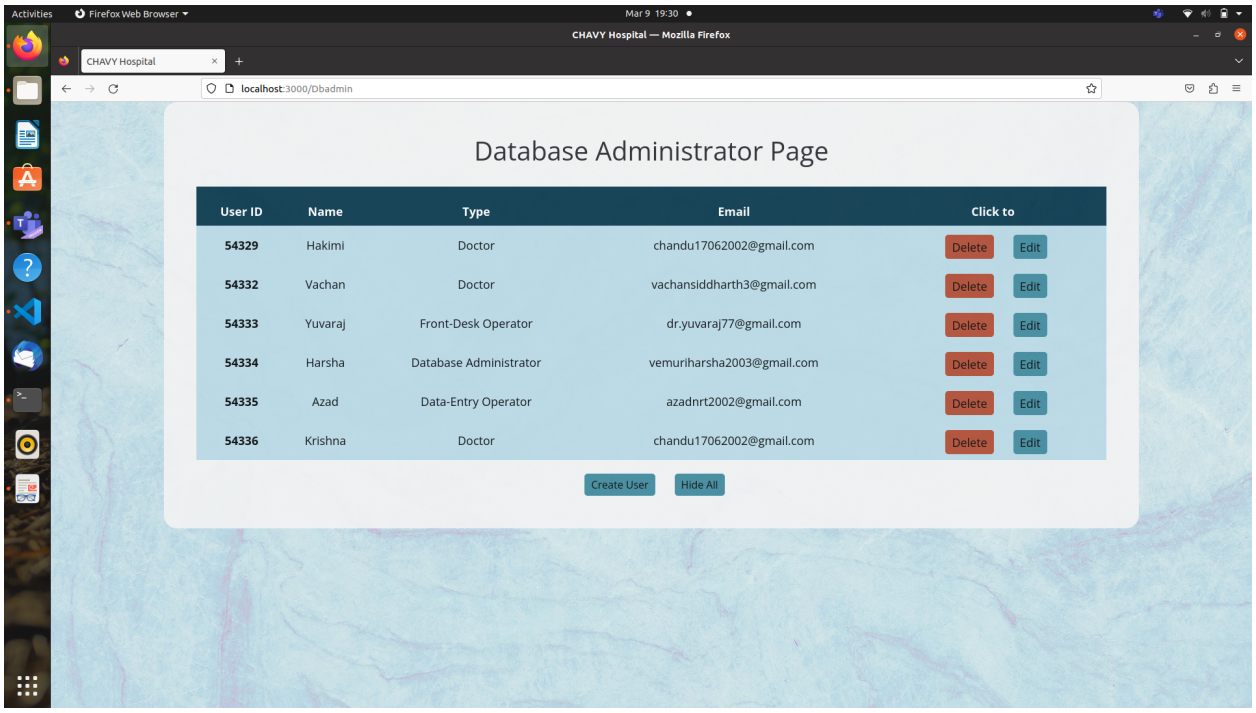
Home page has buttons to add new users and view the existing ones.

Create user:



Here the administrator new users with the above listed credentials of the permitted types.

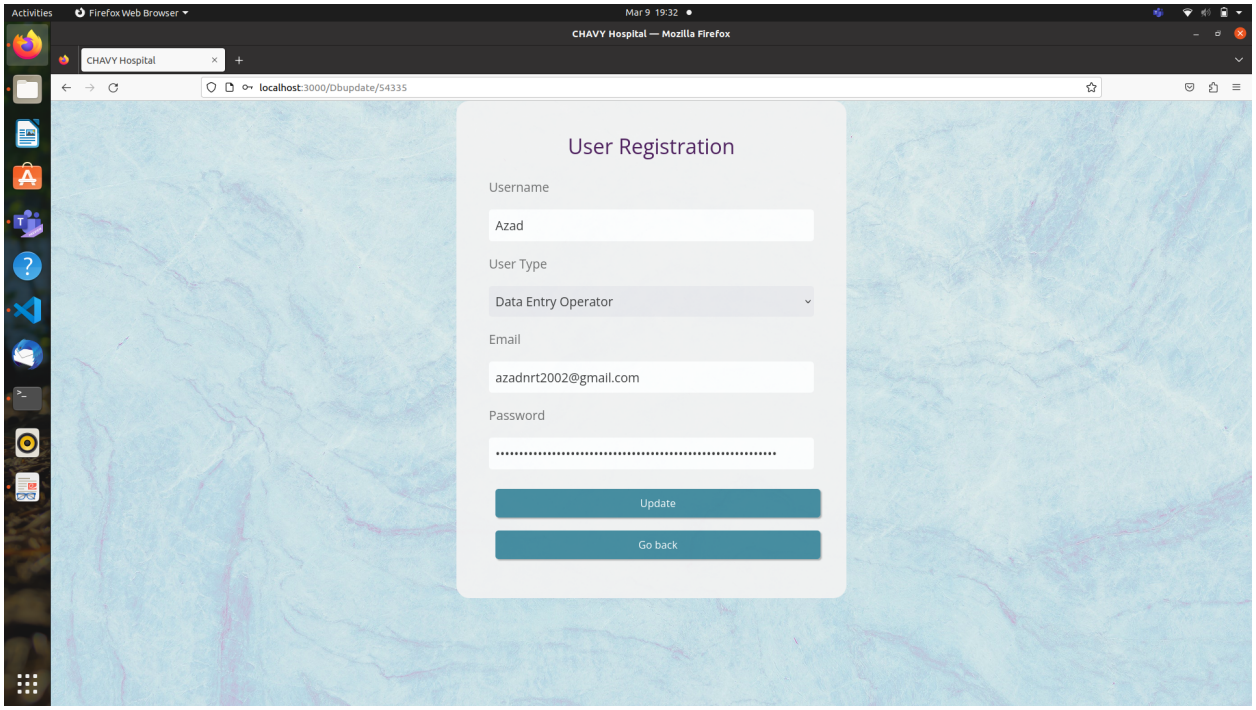
Show Users:



Here the administrator can look at all the users. He is given option to delete or edit the details of that user.

Update User:

When the edit button is triggered we allow the administrator view the current credentials of that user and allow to modify them as shown below. And on updating the credentials will be updated and all other information stays the same.


Activities
Firefox Web Browser
Mar 9 19:32
CHAVY Hospital — Mozilla Firefox
CHAVY Hospital
localhost:3000/Dbupdate/54335
User Registration
Username
Azad
User Type
Data Entry Operator
Email
azadnrt2002@gmail.com
Password
Update
Go back

All Queries list:

Database Administrator:

Add user:

```
db.query(
     "INSERT INTO Users_(Username, Type, Email, Password)
VALUES (?, ?, ?, ?)",
     [Username, Type, Email, hashedPassword],
     (err, result) => {
     if (err) {
          console.error(err);
          res.status(500).send("Server error");
     } else {
          console.log(result);
          res.status(200).send(result);
     }
     }
     );
```

If User is a doctor:

```
db.query(
     "INSERT INTO Doctor (Name ,Doctor_id ,Specialization)
values (?,?,?)",
     [Name, ID, spec],
     (err, result) => {
     console.log(result);
     res.status(200).send(result);
     }
     );
```

Delete User:

```
db.query("DELETE from Users_ where UserID = ? AND Type!='Doctor';",
UserID, (err1, result1) => {
      if (err1) {
            console.log(err1);
            res.status(300).send(result)
      } else {
            console.log(result1);
            db.query("DELETE from Appointment where Doctor_id=? AND
Scheduled_Date > ?",[UserID,date],(err2,result2)=>{
            if(err2){
            console.log(err2)
            res.status(300).send(result)
            }
            else{
            console.log(result2);
            res.status(200).send(result)
            }
            })
            // res.status(200).send(result1)
      }
      });
```

Updating users:

```
db.query("UPDATE Users_ set  Username=?, Type=?, Email=?, Password=?
where UserID = ?",
      [Username, Type, Email, Password, id],
      (err, result) => {
      if (err) {
      console.log(err);
      }
      console.log(result);
      res.status(200).send(result);
      }
  );
```

Data Entry Operator:

Selection of Patient:

```
db.query(
     "SELECT Patient_id,Name,Gender,Phone,Mail  FROM
Patient;",
     (err, result) => {
     console.log(result);
     res.status(200).send(result);
     }
  );
```

Update Prescribed Tests:

```
db.query(
     "UPDATE PrescribedTests set Result=? where Patient_id
= ? and Test_id=?",
     [Result, Patient_id, Test_id],
     (err, result) => {
     if (err) {
     console.log(err);
     }
     console.log(result);
     res.status(200).send(result);
     }
  );
```

Listing all Tests :

```
db.query(
      "SELECT Pa.Name as Patient, D.Name as Doctor, T.Name as Test,
P.Result as Result ,T.Test_id as Test_id,Pa.Patient_id as Patient_id
,P.Scheduled_Date as Date FROM PrescribedTests as P,Test as T, Patient as
Pa, Doctor as D WHERE P.Test_id=T.Test_id and P.Patient_id=? and
Pa.Patient_id=P.Patient_id and D.Doctor_id=P.Doctor_id;",
      [Patient_id],
      (err, result) => {
      if(!err){
      console.log(result);
      let final=[];
      db.query("SELECT Pa.Name as Patient,T.Name as Test, P.Result as
Result ,T.Test_id as Test_id,Pa.Patient_id as Patient_id ,P.Scheduled_Date
as Date FROM PrescribedTests as P,Test as T, Patient as Pa WHERE
P.Test_id=T.Test_id and P.Patient_id=? and Pa.Patient_id=P.Patient_id AND
P.Doctor_id is NULL",[Patient_id],(err1,result1)=>{

      if(!err1)
      {
            final=result.concat(result1);
            formattedResult = final.map((row) => {
                  const formattedScheduledDate =
update_Current_Date(row.Date);
                  return {
                  ...row,
                  Date: formattedScheduledDate,
                  };
            });
            res.status(200).send(formattedResult);
            console.log(formattedResult);
      }
      else console.log(err1)
      })

      }
      }
  );
```

Listing Treatments:

```
db.query(
      "SELECT Pa.Name as Patient, D.Name as Doctor, T.Name as Treatment,
U.Result as Result ,T.Treatment_id as Treatment_id, Pa.Patient_id as
Patient_id,U.Scheduled_Date as Date FROM Undergoes as U,Treatment as T,
Patient as Pa, Doctor as D WHERE U.Treatment_id=T.Treatment_id and
U.Patient_id=? and Pa.Patient_id=U.Patient_id and D.Doctor_id=U.Doctor;",
      [Patient_id],
      (err, result) => {
      if(!err){
      console.log(result);
      let final=[];
      db.query("SELECT Pa.Name as Patient,T.Name as Treatment, U.Result as
Result ,T.Treatment_id as Treatment_id, Pa.Patient_id as
Patient_id,U.Scheduled_Date as Date FROM Undergoes as U,Treatment as T,
Patient as Pa WHERE U.Treatment_id=T.Treatment_id and
Pa.Patient_id=U.Patient_id and U.Doctor is NULL and U.Patient_id=?
;",[Patient_id],(err1,result1)=>{

      if(!err1)
      {
            final=result.concat(result1);
            formattedResult = final.map((row) => {
            const formattedScheduledDate = update_Current_Date(row.Date);
            return {
                  ...row,
                  Date: formattedScheduledDate,
            };
            });
            res.status(200).send(formattedResult);
            console.log(formattedResult);
      }
      else console.log(err1)
      })

      }
      }
  );
```

Update Undergoes table:

```
db.query(
     "UPDATE Undergoes set Result=? where Patient_id = ?
and Treatment_id=?",
     [Result, Patient_id, Treatment_id],
     (err, result) => {
     if (err) {
     console.log(err);
     }
     console.log(result);
     res.status(200).send(result);
     }
  );
```

Adding new Test record:

```
db.query(
     "INSERT INTO PrescribedTests (Test_id ,Patient_id
,Doctor_id,Scheduled_Date) values (?,?,?,NOW())",
     [Test, patientID, Doctor],
     (err, result) => {
     console.log(result);
     res.status(200).send(result);
     }
     );
```

Doctor:

Past Appointments:

```
const query="SELECT * FROM Appointment as A,Patient as P
where A.Doctor_id =? AND P.Patient_id=A.Patient_id AND
A.Scheduled_Date < ? ORDER BY A.Scheduled_Date"
  db.query(query,[id,Current_Date],(err,result)=>{
     const formattedResult = result.map((row) => {
     const formattedScheduledDate =
update_Current_Date(row.Scheduled_Date);
     return {
     ...row,
     Scheduled_Date: formattedScheduledDate,
     };
     });
     res.send(formattedResult)
     console.log(formattedResult)
     console.log(err)
  });
```

Future Appointments:

```
const query="SELECT * FROM Appointment as A,Patient as P where A.Doctor_id =?
AND P.Patient_id=A.Patient_id AND A.Scheduled_Date >= ? ORDER BY
A.Scheduled_Date"
  db.query(query,[id,Current_Date],(err,result)=>{
     const formattedResult = result.map((row) => {
     const formattedScheduledDate = update_Current_Date(row.Scheduled_Date);
     return {
     ...row,
     Scheduled_Date: formattedScheduledDate,
     };
     });
     res.send(formattedResult)
     console.log(formattedResult)
     console.log(err)
  });
```

Patients' Dashboard:

```
const query="SELECT DISTINCT P.Patient_id,P.* FROM
Appointment as A,Patient as P where A.Doctor_id =? AND
P.Patient_id=A.Patient_id"
  db.query(query,[id],(err,result)=>{
     res.send(result)
     console.log(result)
     console.log(err)
  });
```

Prescribe tests:

```
query1 = "SELECT Count(*) as count_limit From PrescribedTests
WHERE Patient_id = ? AND Test_id = ? AND Doctor_id = ? AND (
DATE(Scheduled_Date) >= ? OR Scheduled_Date is NULL)"
  let query2="Insert into PrescribedTests
(Patient_id,Test_id,Doctor_id) values(?, ?, ?)"

db.query(query1,[Patient_id,selectedTest,Doctor_id,dDate],(err,r
esult)=>{
     if(result[0].count_limit == 0)
     {

db.query(query2,[Patient_id,selectedTest,Doctor_id],(err1,result
1)=>{
          res.status(200).send("Inserted")
          //console.log(err)
          console.log(result)
     });
     }
     else
     {
     res.status(200).send("Present");
     }
  })
```

Prescribe Treatment:

```
  let query1 = "SELECT Count(*) as count_limit From
Undergoes WHERE Patient_id = ? AND Treatment_id = ? AND
Doctor = ? AND ( DATE(Scheduled_Date) >= ? OR
Scheduled_Date is NULL)"
  let query2="Insert into Undergoes
(Patient_id,Treatment_id,Doctor) values(?, ?, ?)"

db.query(query1,[Patient_id,selectedTest,Doctor_id,dDate],(
err,result)=>{
     console.log(err)
     if(!err){
     if(result.length>0 && result[0].count_limit == 0)
     {

db.query(query2,[Patient_id,selectedTest,Doctor_id],(err1,r
esult1)=>{
                res.status(200).send("Inserted")
                //console.log(err)
                //console.log(result)
     });
     }
     else
     {
     res.status(200).send("Present");
     }
  }
  })
```

Test results:

```
const query="SELECT PT.*,T.Name as T_Name,D.Name as
D_Name,P.Name as P_Name FROM PrescribedTests as PT,Test as
T, Patient as P, Doctor as D where PT.Patient_id =
P.Patient_id AND PT.Doctor_id=D.Doctor_id AND PT.Test_id =
T.Test_id AND PT.Patient_id =  ? "
  db.query(query,[id],(err,result)=>{
     console.log(err)
     if(!err){
     if(result.length>0){
     const formattedResult = result.map((row) => {
     const formattedScheduledDate =
update_Current_Date(row.Scheduled_Date);
     return {
          ...row,
          Scheduled_Date: formattedScheduledDate,
     };
     });
     console.log(formattedResult)
     res.send(formattedResult)
     }
     else res.send(result)
  }

  });
```

Treatment Results:

```
const query="SELECT U.*,T.Name as T_Name,D.Name as
D_Name,P.Name as P_Name FROM Undergoes as U, Treatment as
T,Patient as P, Doctor as D where U.Patient_id =
P.Patient_id AND U.Doctor=D.Doctor_id AND U.Treatment_id =
T.Treatment_id AND U.Treatment_id = T.Treatment_id AND
```

```
U.Patient_id =  ? "
  db.query(query,[id],(err,result)=>{
     console.log(err)
     if(!err){
     if(result.length>0){
     const formattedResult = result.map((row) => {
     const formattedScheduledDate =
update_Current_Date(row.Scheduled_Date);
     return {
          ...row,
          Scheduled_Date: formattedScheduledDate,
     };
     });
     console.log(formattedResult)
     res.send(formattedResult)
     }
     else res.send(result)
  }
  });
```

Results Description:

```
const query="SELECT A.*,D.Name as D_Name,P.Name as P_Name
FROM Appointment as A,Patient as P, Doctor as D where
A.Patient_id = P.Patient_id AND A.Doctor_id=D.Doctor_id AND
A.Patient_id =  ? AND A.Description is NOT NULL"
  db.query(query,[id],(err,result)=>{
     console.log(err)
     if(!err){
     if(result.length>0){
     const formattedResult = result.map((row) => {
     const formattedScheduledDate =
update_Current_Date(row.Scheduled_Date);
     return {
```

```
            ...row,
            Scheduled_Date: formattedScheduledDate,
        };
        });
        console.log(formattedResult)
        res.send(formattedResult)
        }
        else res.send(result)
    }
    });
```

Post-Appointment remarks:

```
const query="UPDATE Appointment SET `Description` = ? Where
Appointment_id=?";
        db.query(query,[remarks,id],(err,result)=>{
        if(!err)
        {
        res.status(200).send("Remarks added successfully");
        }
        });
```

Frontdesk Operator:

Treatment Scheduling:

```
const query="SELECT U.U_id, P.Name as P_Name, D.Name as D_Name,
T.Name as T_Name, P.Patient_id, D.Doctor_id FROM Patient as P, Doctor
as D, Treatment as T, Undergoes as U WHERE
(U.Patient_id,U.Doctor,U.Treatment_id) =
(P.Patient_id,D.Doctor_id,T.Treatment_id) AND U.Scheduled_Date is
NULL"
  const query1="SELECT U.U_id, P.Name as P_Name, T.Name as T_Name,
P.Patient_id FROM Patient as P, Treatment as T, Undergoes as U WHERE
(U.Patient_id,U.Treatment_id) = (P.Patient_id,T.Treatment_id) AND
U.Scheduled_Date is NULL AND U.Doctor is NULL"
  db.query(query,(err,result)=>{
      console.log(result)
      console.log(err)
      if(err)
      {
      res.status(300).send(result);
      }
      else
      {
      let final=[];

      db.query(query1,(err1,result1)=>{
      if(!err1)
      {
            final=result.concat(result1);
            res.status(200).send(final);
            console.log(final);
      }
      else
      {
            console.log(err1);
      }
      });
      }

  });
```

Patient Registration:

```
db.query("INSERT INTO
Patient(Name,Address,Phone,Mail,Gender,DOB,Registration_dat
e) VALUES
(?,?,?,?,?,?,?)",[Name,Address,Phone,Mail,Gender,DOB,Regist
ration_date],  (err, result)=>{
          if(err) {
          res.status(300).send(result);
          console.log(err)
          }
          else
          {
          res.status(200).send(result);
          console.log(result.insertId);
          }
     }
     );
```

Book Appointment:

```
const query="SELECT * FROM Doctor,Users_ Where
Doctor.Doctor_id=Users_.UserID AND Doctor.Status = 1"
  db.query(query,(err,result)=>{
     console.log(result)
     console.log(err)
     if(err)
     {
     res.status(300).send(result);
     }
     else
     {
     res.status(200).send(result)
```

```
    }

  });
```

Schedule Tests:

```
const query="SELECT PT.PT_id, P.Name as P_Name, D.Name as
D_Name, T.Name as T_Name, P.Patient_id, D.Doctor_id FROM
Patient as P, Doctor as D, Test as T, PrescribedTests as PT
WHERE (PT.Patient_id,PT.Doctor_id,PT.Test_id) =
(P.Patient_id,D.Doctor_id,T.Test_id) AND PT.Scheduled_Date
is NULL"

  db.query(query,(err,result)=>{
    console.log(result)
    console.log(err)
    if(err)
    {
    res.status(300).send(result);
    }
    else
    {

    res.status(200).send(result);
    }

  });
```

Booking Slot:

```
const date=req.params.date;
     const docid=req.params.Docid
     console.log(date);
     console.log(docid)
     const data = [
     { Slot: "9:00-9:30" },
     { Slot: "9:30-10:00" },
     { Slot: "10:00-10:30" },
     { Slot: "10:30-11:00" },
     { Slot: "11:00-11:30" },
     { Slot: "11:30-12:00" }
     ];
     let Current_Date = get_Current_Date();
     console.log(Current_Date)
     const query="SELECT DISTINCT Slot FROM Appointment
where DATE(Scheduled_Date) = ? AND Doctor_id = ?"
     db.query(query,[date,docid],(err,result)=>{

     console.log(result)
     if(Current_Date>date+" 9:00:00") result.push({ Slot:
"9:00-9:30"});
     if(Current_Date>date+" 9:30:00") result.push({ Slot:
"9:30-10:00"});
     if(Current_Date>date+" 10:00:00") result.push({ Slot:
"10:00-10:30"});
     if(Current_Date>date+" 10:30:00") result.push({ Slot:
"10:30-11:00"});
     if(Current_Date>date+" 11:00:00") result.push({ Slot:
"11:00-11:30"});
     if(Current_Date>date+" 11:30:00") result.push({ Slot:
"11:30-12:00"});
```

```
    const difference = data.filter(item => {
        return !result.some(otherItem => item.Slot === 
otherItem.Slot);
    });
    console.log(difference);
    console.log(err)
    res.send(difference)

    });
```

Admit Patient:

```
db.query("SELECT * FROM Room WHERE Rem_capacity > 0 ORDER 
BY Rem_capacity DESC LIMIT 1",  (err, result)=>{
        if(result.length === 0) {
        res.status(200).send([{Room_no: "-1"}]);
        console.log(err)
        }
        else
        {
        const room = result[0].Room_no;
        let rem_capacity = result[0].Rem_capacity;
        rem_capacity = rem_capacity - 1;
        res.status(200).send(result);
        db.query("INSERT INTO Stay VALUES (?,?)", 
[id,room], (err1, result1)=>{
            if(err1) res.status(401).send(result1);
        })
        db.query("UPDATE Room SET Rem_capacity = ? WHERE 
Room_no = ?", [rem_capacity,room], (err2, result2)=>{
            if(err2) res.status(401).send(result2);
        })
        }
    }
    );
```

Discharge:

```
db.query("SELECT * FROM Stay,Room WHERE Patient_id = ? AND
Room.Room_no=Stay.Room_no", [id], (err, result)=>{

          let room = result[0].Room_no;
          db.query("DELETE FROM Stay WHERE Patient_id = ?
AND Room_no = ?", [id,room], (err1, result1)=>{
               if(err1) res.status(401).send(result1);
          })
          db.query("SELECT Rem_capacity FROM Room WHERE
Room_no = ?", [room], (err3, result3)=>{
               if(err3) res.status(401).send(result3);
               else
               {
                    let rem_capacity =
result3[0].Rem_capacity;
                    rem_capacity = rem_capacity + 1;
                    db.query("UPDATE Room SET Rem_capacity =
? WHERE Room_no = ?", [rem_capacity,room], (err2,
result2)=>{
                         if(err2)
res.status(401).send(result2);
                    })
               }
               })
               console.log(result);
               res.status(200).send(result);
     });
```

Conclusion:

The application was developed in order to adjust for any future updates made to the data later. The data redundancy is reduced to the minimum possible level and the data is securely stored by protecting it from any alterations from any users other than those who are supposed to have access to edit.