
**Department of Computer Engineering**

**CS 353 - Database Systems**

# Hospital Database Management System

**Design Report**

**Instructor: Uğur Güdükbay**

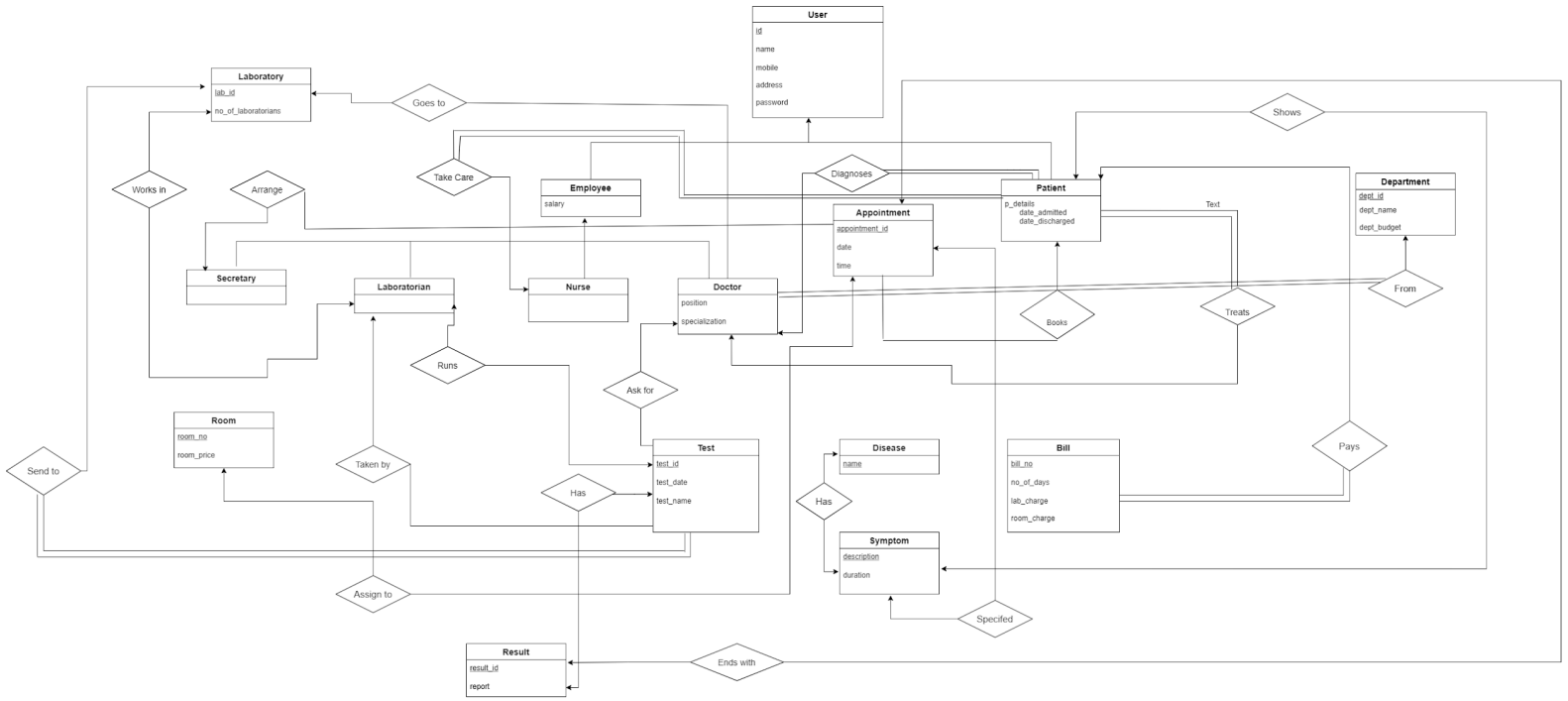
**Group 28**

**Aziz Ozan Azizoğlu - 21401701**

**Mastan Abdulkhaligli - 21403007**

**Emna Maghrebi - 22004007**

# 1. Revised E/R Model




After consulting our TA's feedback , and while designing the User Interface , we have decided to make the following changes in our E/R model to improve the structure of our database system:

- Primary keys removed from children entities
- Result/component part reviewed
- Symptom is made as an entity not an attribute
- Appointment is made as an entity not a relationship , and is connected to other important entities
- We kept room and its relations because every room has a different price , and also every patient needs to be assigned to a room in order to get his treatment and for the nurses to take care of  him .
- Removed foreign keys from entities in the ER diagram

# 2. Relation Schemas

## 2.1. User

**Relational Model:**

User(id, password, name, phone_number, email, address)

**Candidate Keys:** { (id), (phone_number), (email) }

**Table Definition:**

```
create table User(
        id                  int not null,
        password            varchar(15) not null,
        name                varchar(30) not null,
        phone_number        int not null,
        email               varchar(30) not null,
        primary key (id),
        unique (phone_number, email)
);
```

## 2.2. Employee

**Relational Model:**

Employee(e_id, salary)

**Candidate Keys:** { (e_id) }

**Table Definition:**

```
create table Employee(
        e_id                int not null,
        salary              numeric(8, 2),
        primary key (e_id),
        foreign key (e_id) references User(id)
);
```

## 2.3. Doctor

**Relational Model:**

Doctor(d_id, position, specialization)

**Candidate Keys:** { (d_id) }

**Table Definition:**

```
create table Doctor(
        d_id                int not null,
        d_salary            numeric(8, 2),
        position            varchar(20),
        specialization      varchar(20),
        primary key (d_id),
        foreign key (d_id) references User(id),
        foreign key (d_salary) references Employee(salary)
);
```

## 2.4. Laboratiorian

**Relational Model:**

Laboratiorian(l_id, lab_no)

**Candidate Keys:** { (l_id) }

**Table Definition:**

```
create table Laboratorian(
        l_id                int not null,
        lab_no              int not null,
        l_salary            numeric(8, 2),
        primary key (l_id),
        foreign key (l_id) references User(id),
        foreign key (l_salary) references Employee(salary)
        foreign key (lab_no) references Laboratory(lab_id)
```

);

## 2.5. Nurse

**Relational Model:**

Nurse(n_id, n_salary)

**Candidate Keys:** { (n_id) }

**Table Definition:**

```
create table Nurse(
        n_id                    int not null,
        n_salary                numeric(8, 2),
        primary key (n_id),
        foreign key (n_id) references User(id),
        foreign key (n_salary) references Employee(salary)
);
```

## 2.6. Secretary

**Relational Model:**

Secretary(s_id, s_salary)

**Candidate Keys:** { (s_id) }

**Table Definition:**

```
create table Secretary(
        s_id                    int not null,
        s_salary                numeric(8, 2),
        primary key (s_id)
        foreign key (s_id) references User(id),
        foreign key (s_salary) references Employee(salary)
);
```

## 2.7. Result

**Relational Model:**

Result(<u>result_id</u>, test_id, test_name, report, test_date, lab_no)

**Candidate Keys:** { (result_id), (test_id) }

**Table Definition:**

```
create table Result(
        test_id             int not null,
        test_name           varchar(20),
        result_id           int not null,
        report              varchar(500),
        test_date           date,
        lab_id              int not null,
        primary key (result_id),
        foreign key test_name references Test(test_name),
        foreign key test_id references Test(test_id),
        foreign key test_date references Test(test_date),
        foreign key lab_id references Laboratory(lab_id),
        unique (test_id)
);
```

## 2.8. Test

**Relational Model:**

Test(test_id, date, category, lab_no)

**Candidate Keys:** { (test_id), (test_name) }

**Table Definition:**

```
create table Test(
        test_id             int not null,
        test_date           date,
        test_name           varchar(20),
        lab_no              int not null,
        primary key (test_id),
        unique (test_name)
);
```

## 2.9. Disease

**Relational Model:**

Disease(name)

**Candidate Keys:** { (name) }

**Table Definition:**

```
create table Disease(
        name                varchar(20),
        primary key (name)
);
```

## 2.10. Symptom

**Relational Model:**
Symptom(description, duration)

**Candidate Keys:** { (description) }

**Table Definition:**
**create table** symptom(
description **varchar(400) not null**,
duration **time**,
**primary key** (duration)
);

## 2.11. Patient

**Relational Model:**
Patient(p_id, room_no, date_admitted, date_discharged, complaint)

**Candidate Keys:** { (p_id), (room_no) }

**Table Definition:**
**create table** Patient(
p_id **int not null**,
room_no **int not null**
date_admitted **date not null**,
date discharged **date not null**,
complaint **varchar(20)**,
**primary key** (p_id),
**foreign key** (p_id) **references** User(id)
**foreign key** room_no **references** Room(room_no)
);

## 2.12. Bill

**Relational Model:**
Bill(bill_no, no_of_days, room_charge)

**Candidate Keys:** { (bill_no) }

**Table Definition:**

```
create table Bill(
        bill_no             int not null,
        no_of_days          int,
        room_charge         numeric(8,2),
        primary key (bill_no)
);
```

## 2.13. Department

**Relational Model:**
Department(dept_id, dept_name, dept_budget)

**Candidate Keys:** { (dept_id), (dept_name) }

**Table Definition:**

```
create table Department(
        dept_id         int not null,
        dept_name           varchar(20) not null,
        dept_budget         numeric(12,2),
        primary key (dept_id),
        unique (dept_name)
);
```

## 2.14. Appointment

**Relational Model:**
Appointment(appointment_id, d_id, p_id, date, time)

**Candidate Keys:** { (appointment_id) }

**Table Definition:**
```
create table Appointment(
        appointment_id      int not null,
        d_id                int not null,
        p_id                int not null,
        date                date not null,
        time                time not null,
        primary key (appointment_id),
        foreign key (d_id) references Doctor(d_id),
        foreign key (p_id) references Patient(p_id)
);
```

## 2.15. Laboratory

**Relational Model:**
Laboratory(lab_id, no_of_laboratorians)

**Candidate Keys:** { (lab_id) }

**Table Definition:**
```
create table Laboratory(
        lab_id              int not null,
        no_of_laboratorians int,
        primary key (lab_id)
);
```

## 2.16. Room

**Relational Model:**
Room(room_no, unit_price, availability)

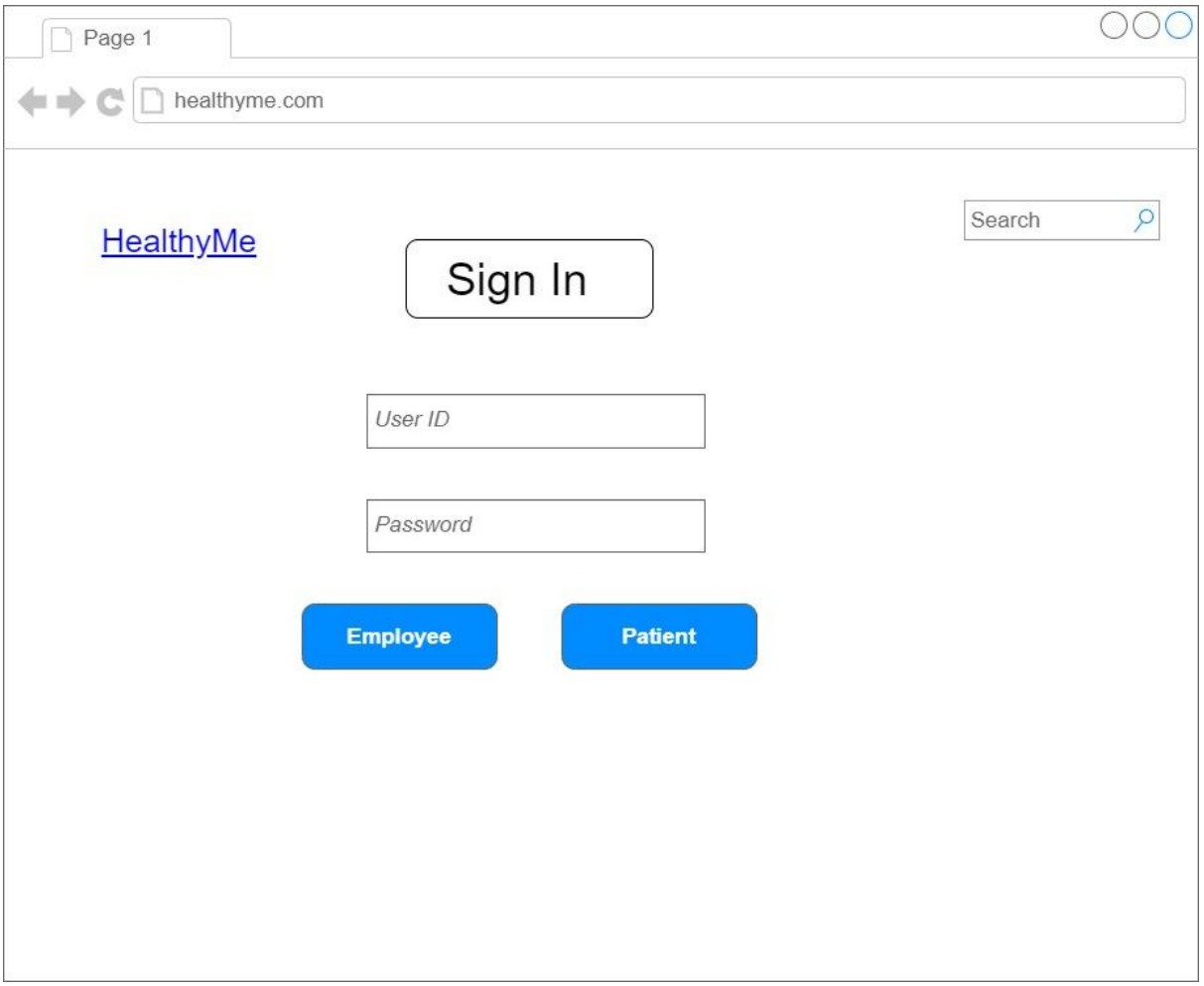
**Candidate Keys:** { (room_no) }

**Table Definition:**

```
create table Room(
        room_no         int not null,
        unit_price      numeric(4, 2) not null,
        availability    boolean not null,
        primary key (room_no)
);
```

# 3. User Interface Design & Corresponding SQL Statements

## 3.1. Sign In Page




**Inputs:** entered_id, entered_password

To check if user with given ID and password exists in database:

select * from user

where id = entered_id and password = entered_password

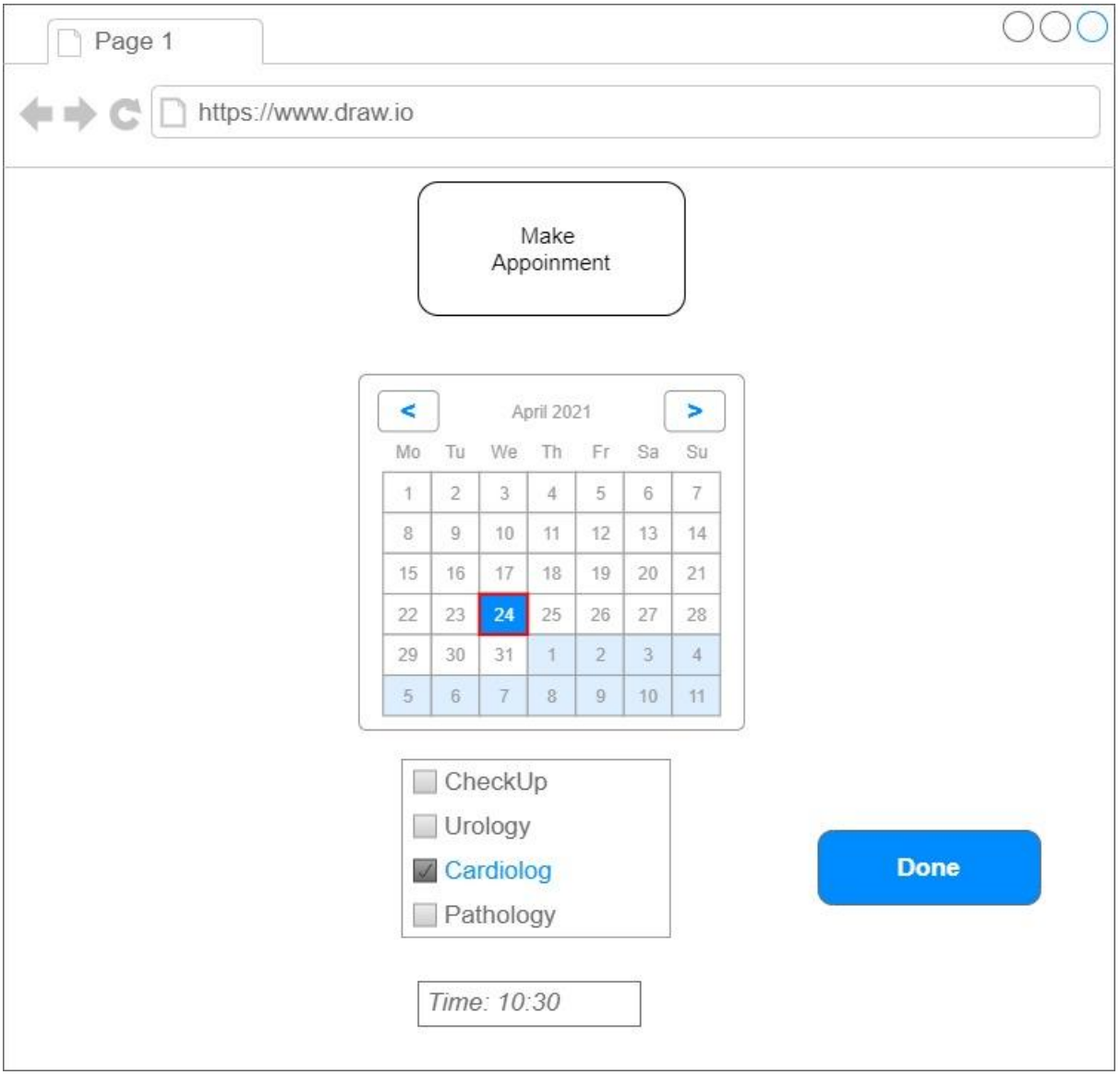
To check if employee with given ID and password exists in database:

select * from employee natural join user

where id = 'mastanabdulkhaligli@gmail.com'

and password = md5("123456789!")

## 3.2. Make Appointment Page




**Inputs:** entered_id, entered_pass

Before assigning an appointment first check this time slot available or not. Then assign appointments with a doctor. If the time slot taken appointment fails.

select * from user
where id = entered_id

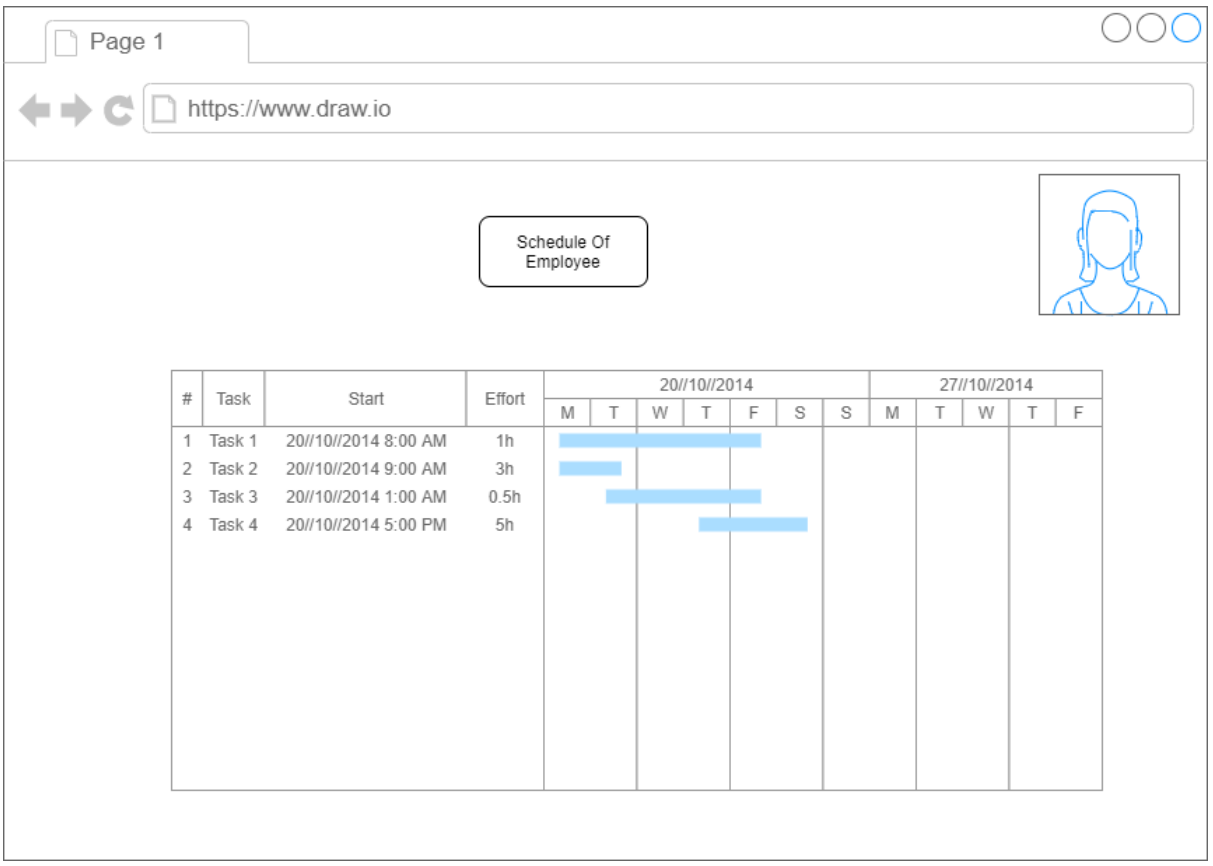
and password = entered_pass

**Assigning appointment:**

Insert into Appointment

values (0132546, 21403007, 1236547, “”16.03.2021, “15:30”)

## 3.3. Schedule of Employee Page

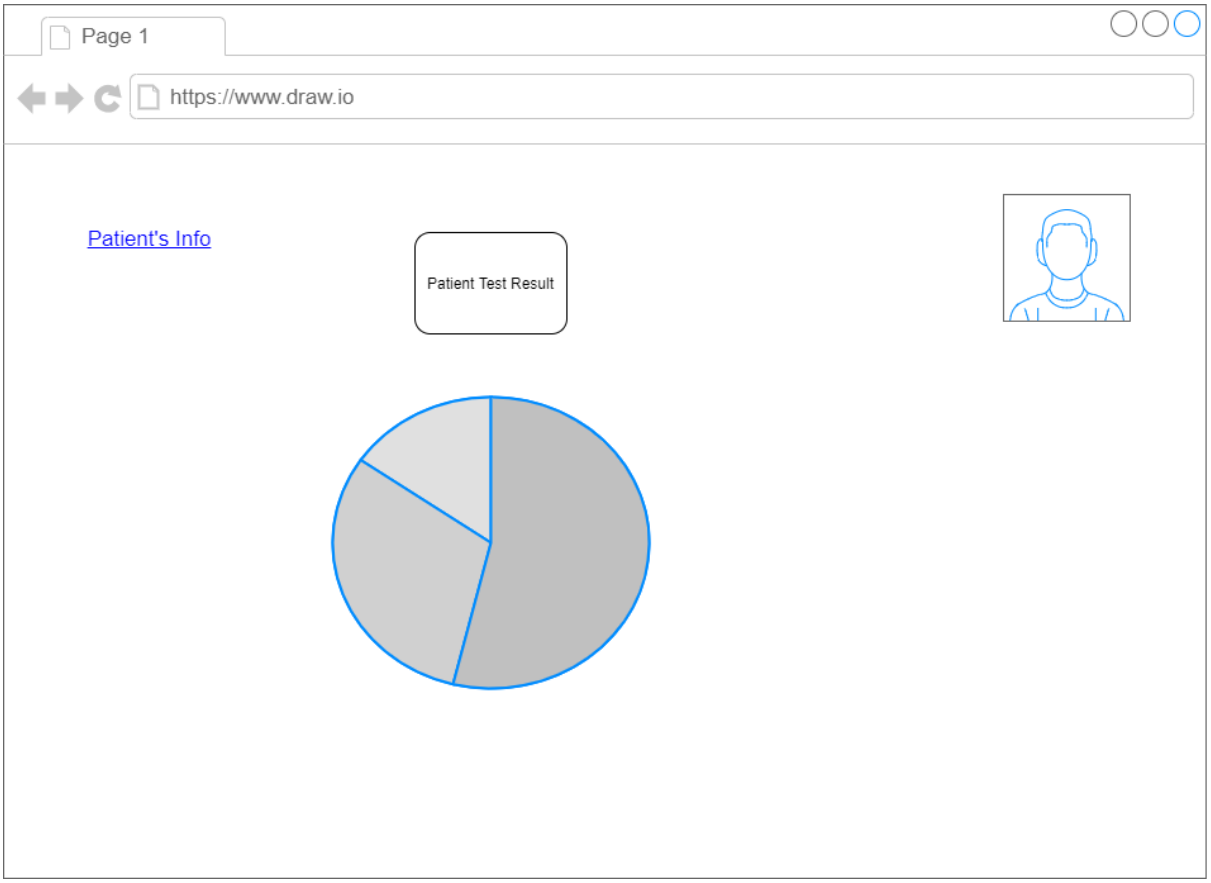



**Inputs**: doctor_id

Employees such as doctors can see appointments which are assigned to him/her.

select *
from appointment
where d_id = doctor_id

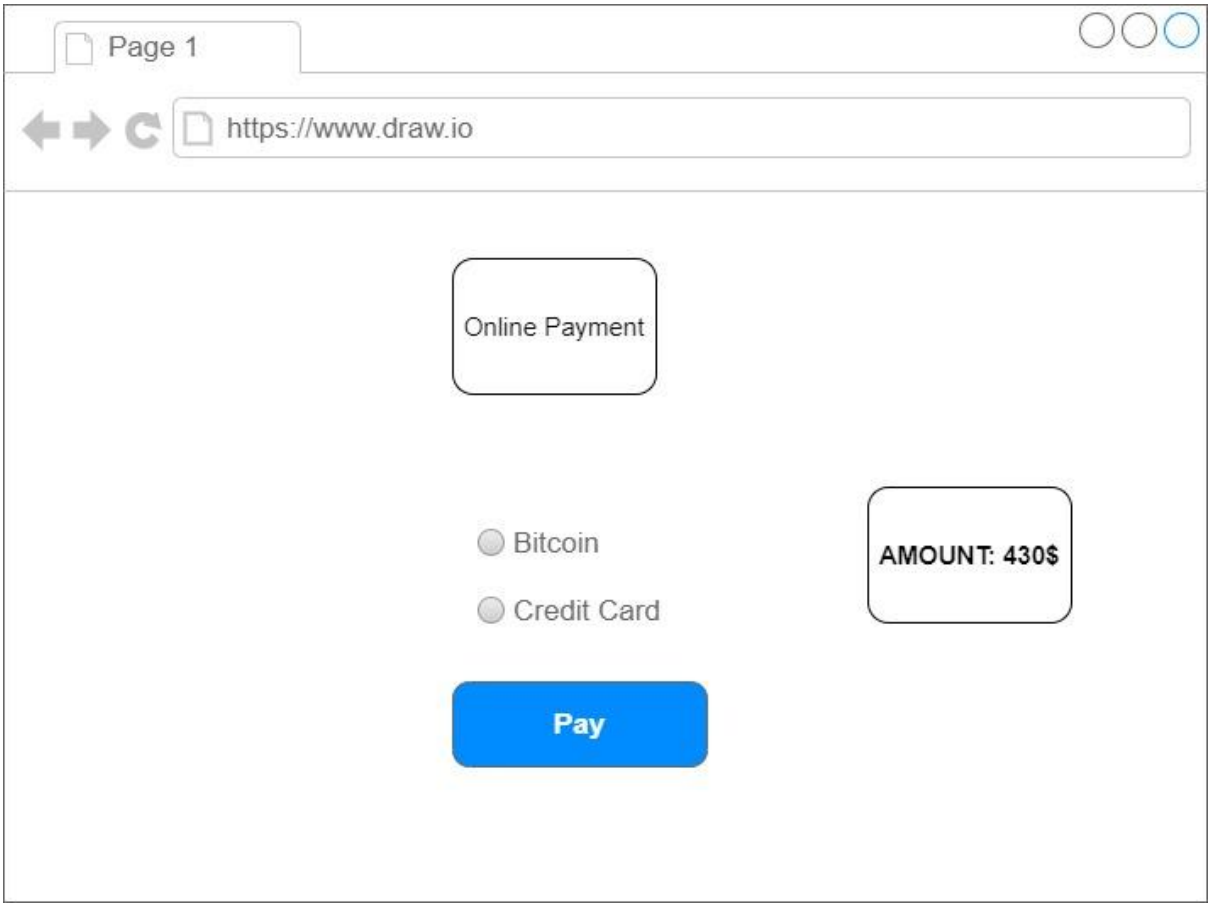
## 3.4. Test Result Page




**Inputs:** entered_result_id

select result_id, report
from Result natural join Test
where result_id = entered_result_id

## 3.5. Online Payment Page




**Inputs:** entered_bill_no

To pay online:

select bill_no, no_of_days, room_charge
from Bill
where bill_no = entered_bill_no

## 3.6. Download Result


**Inputs:** patient_id

To see test results:

```
select  *
from Test natural join Patient
where patient_id = Patient.p_id
```

# 4. Implementation Plan

For our software design, we are planning to use PHP for the backend, Bootstrap for the front-end. The reason behind using PHP is that we are more familiar with the development environment of it. Bootstrap is used because it has various sources and libraries, also we think that Bootstrap is more user-friendly. In order to construct and maintain the database, we are planning to use MySQL server to put our data.

# 5. Website

https://mastanabdulkhaligli.github.io/HelathMe.github.io/