PROJECT DESIGN REPORT

for

HEALTH CENTER

Prepared by: Mustafa AKILLI 1818897

Burak GOYNUK 1819374

1. Introduction

In this part of the report, the definition of the problem, and the general structure of our solution will be stated. Our team is responsible for developing an efficient solution to manage a database related to a health center. The system should be available for use of doctors, patients, and different types of employees or anyone who is in a health system. The system will also provide an admin panel for managing health center. The problem does not only include implementing a database management system but also includes serving this solution as a web application. The web application will be a full stack system, which simulates a health center in many perspectives such as both front-end, server and client side, and an effective database management system.

During development process, the machines running on Windows Operating System will be used. Oracle SQL Developer IDE will be used as development environment. In addition, for DBMS, Oracle will be used. The stored procedures will be written in PL/SQL with using Oracle SQL Developer and they will be executed on Oracle DBMS. In addition to DBMS and database part of the application, as server languages, PHP is selected. For some client-side works, JavaScript and AJAX technologies will be used. Since the resulting application will be serving through as internet application, the application will be independent from operating system and hardware specialties of the end-users. The application shall run all web browsers which are popular.

2. Tables and Advance Database Operations

The tables which will be stored in database are stated in this part in detail. Their relationships, their fields and advance database operations on them are also described in this section of document. The E/R diagram of the application was converted into relations. Each relation's attribute domains, candidate/foreign keys, and referential integrity constraints will be expressed in detail. The relations of the application are as follows:

2.1 Tables

2.1.1 Employee

Fields: This relation keeps the fields employeeID(int), password(varchar), and employeeType(varchar).
Keys: The employeeID is the primary key of this relation.
Constraints: Password and employeeType columns cannot be empty.

Employee = (<u>EmployeeID</u>, Password, EmployeeType)

2.1.2 Doctor

Fields: This relation keeps the fields doctorID(int), name(varchar), surname(varchar), phone(varchar), salary(int), password(varchar) and qualification(varchar).
Keys: The doctorID is the primary key of this relation. The doctorID is also the foreign key which references to Employee table.
Constraints: All columns of this relation cannot be empty. Phone of the each doctor should be unique. Phone number length cannot be less than 10 or more than 13 (Ex. 312 323 50 50 or 505 666 55 55 or +90 505 666 55 55).
Trigger: If the phone number of the doctor is not set in correct form, (+90) will be added to this field.

Doctor = (<u>DoctorID</u>, Password, Name, Surname, Phone, Salary, Qualification)

2.1.3. Secretary

Fields: This relation keeps the fields secretaryID(int), name(varchar), surname(varchar), phone(varchar), salary(int), and password(varchar).
Keys: The secretaryID is the primary key of this relation. The secretaryID is also the foreign key which references to Employee table.
Constraints: All columns of this relation cannot be empty. Phone of the each secretary should be unique. Phone number length cannot be less than 10 or more than 13 (Ex. 312 323 50 50 or 505 666 55 55 or +90 505 666 55 55).
Trigger: If the phone number of the secretary is not set in correct form, (+90) will be added to this field.

Secretary = (<u>SecretaryID</u>, Password, Name, Surname, Phone, Salary)

2.1.4. Staff

Fields: This relation keeps the fields staffID(int), name(varchar), surname(varchar), phone(varchar), salary(int), and password(varchar).
Keys: The staffID is the primary key of this relation. The staffID is also the foreign key which refers to Employee table.
Constraints: All columns of this relation cannot be empty. Phone of the each staff should be unique. Phone number length cannot be less than 10 or more than 13 (Ex. 312 323 50 50 or 505 666 55 55 or +90 505 666 55 55).
Trigger: If the phone number of the secretary is not set in correct form, (+90) will be added to this field.

Staff = (<u>StaffID</u>, Password, Name, Surname, Phone, Salary)

2.1.5. Patient

Fields: This relation keeps the fields SSN(varchar), name(varchar), surname(varchar), phone(varchar), and address(varchar).
Keys: The SSN is the primary key of this relation.
Constraints: SSN, name, surname and phone number cannot be empty. Phone of the each patient should be unique. Phone number length cannot be less than 10 or more than 13 (Ex. 312 323 50 50 or 505 666 55 55 or +90 505 666 55 55).
Trigger: If the phone number of the secretary is not set in correct form, (+90) will be added to this field.

Patient = (SSN, Name, Surname, Phone, Address)

2.1.6. Record

Fields: This relation keeps the fields recordID(int), SSN(varchar), doctorID(int), status(varchar), treatment(varchar), roomID(int), startDate(date), and endDate(date).
Keys: The recordID is the primary key of this relation. The SSN field is the foreign key which refers to Patient table, doctorID is another foreign key referring to Doctor Table, and roomID is also foreign key which refers to Room table.
Constraints: StartDate cannot be empty.
Trigger: If the phone number of the secretary is not set in correct form, (+90) will be added to this field.

Record = (RecordID, SSN, DoctorID, Status, Treatment, RoomID)

2.1.7. Room

Fields: This relation keeps the fields roomID(int), name(varchar), numberOfPatient(int).
Keys: The roomID is the primary key of this relation.
Constraints: Name must be unique and cannot be empty.
Trigger: The number of patients cannot be more than 2.

Room = (RoomID, Name, NumberOfPatient)

2.1.8. ResponsibleFor

Fields: This relation keeps the fields roomID(int), StaffID(int).
Keys: The roomID, staffID tuple is the primary key of this relation. Both are foreign key that are referenced from Room Table and Staff Table respectively.
Constraints: -
Trigger: -

ResponsibleFor = (StaffID, RoomID)

2.9. StaysIn

Fields: This relation keeps the fields SSN(varchar) and roomID(int).
Keys: SSN is the primary key of this relation. Both field are foreign key that are referenced from Patient and Room Table respectively.
Constraints: -
Trigger: -

StaysIn = (SSN, RoomID)

2.2 Advanced Database Operations

2.2.1 Triggers

First trigger is for checking phone format.

```
CREATE OR REPLACE TRIGGER phoneNumberTrigger
BEFORE INSERT ON Doctor, Secretary, Staff, Patient
FOR EACH ROW

DECLARE
  newPhone varchar2(13);

BEGIN
  IF (LENGTH( :NEW.phone ) = 10) THEN
    newPhone := '+90' || :NEW.phone;
  END IF;

  IF (LENGTH( :NEW.phone ) = 11) THEN
    newPhone := '+9' || :NEW.phone;
  END IF;

  :NEW.phone := newPhone;

END;
```

The second trigger is for checking room capacity. The purpose of this trigger is to avoid that more than 2 patients stay in same room.

```
CREATE OR REPLACE TRIGGER roomCapacity
BEFORE INSERT ON StaysIn
FOR EACH ROW
DECLARE
  null;
BEGIN
  IF ( :NEW.numOfPatient < 2 )
            null;
  ELSE
            ABORT;
  ENDIF
END;
```

2.2.2 Important Procedures

We select login and add doctor procedures. Since login is common for all employees, it is selected. Adding doctor or secretary or staff is very similar to each other, so adding doctor is also selected.

```
PROCEDURE login(
      p_employee_id IN INTEGER,
      p_password   IN VARCHAR2,
      p_RESULT OUT CHAR)
  IS
    countU INTEGER := 0;

  BEGIN
  SELECT COUNT(*)
  INTO countU
  FROM Employee
  WHERE employeeId= p_employee_id and password = p_password;
```

```
IF countU > 0 THEN
  p_RESULT := 'T';
ELSE
  p_RESULT := 'F';
END IF;

END login;
```

/***/

```

PROCEDURE add_doctor(
      p_doctor_id IN INTEGER,
      p_name IN VARHAR2,
      p_surname IN VARCHAR2,
      p_phone IN VARHAR2,
      p_surname IN VARCHAR2,
      p_qualification IN VARCHAR2)
  IS
  BEGIN
    INSERT INTO Doctor
    (doctorId, name, surname, phone, salary, qualification, password)
    values (
              p_doctor_id,
              p_name,
              p_surname,
              p_phone,
              p_surname,
              p_qualification );
  END add_doctor;
```

Since there is no functional dependency in our system, it satisfies the both BCNF and 3NF condition.

3. GUI Components & Interfaces

The interfaces which will be shown on browsers are explained in this part in detail. The usage of application and how the users can act are detailed.

3.1. Main Page

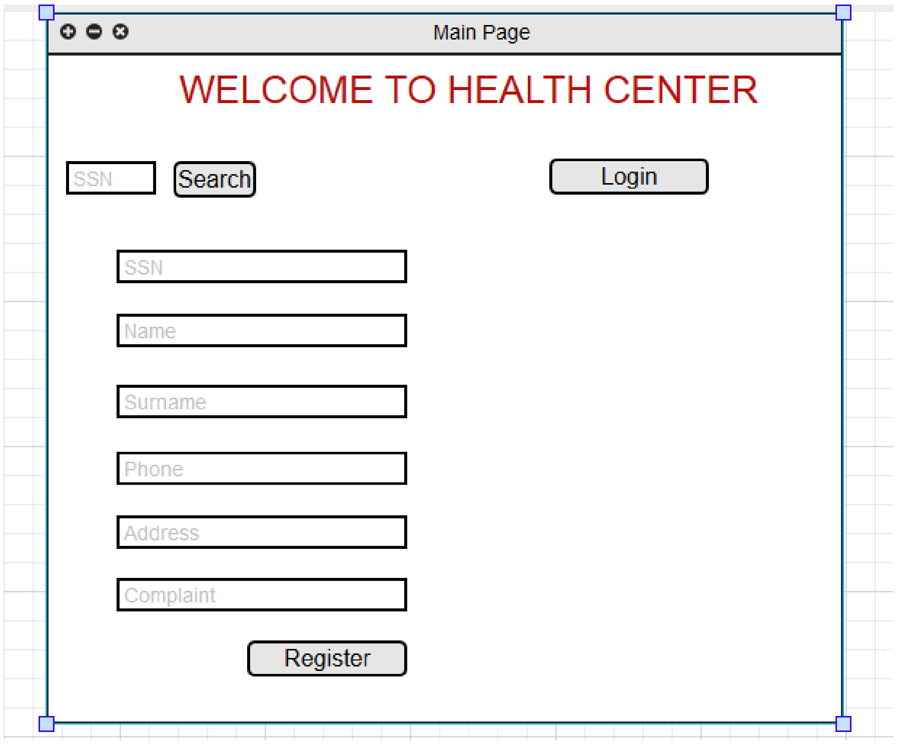


Figure 1 – Main Page

This is the index or home page of the application. Everyone who knows the URL of the application can see this page. In this page the patients can search records by entering their SSN on the field on top-left of the page. Also anyone can fill the patient form to register to hospital. After submission of form, if there is no patient with SSN number, automatically a row will be generated and added to Patient table. If there is such a person, there will be no action. Moreover there is login button on the top-right of the page. The employees of the health medical center can login the system.

The SQL queries that will be used for this screen are as follows;

Search Record: *SELECT * FROM Record WHERE ssn = givenSSN*
Submit: *INSERT INTO Record VALUES(.. , given values, ..)*
INSERT INTO Patient VALUES(.. , given values, ..)

3.2. Login Page

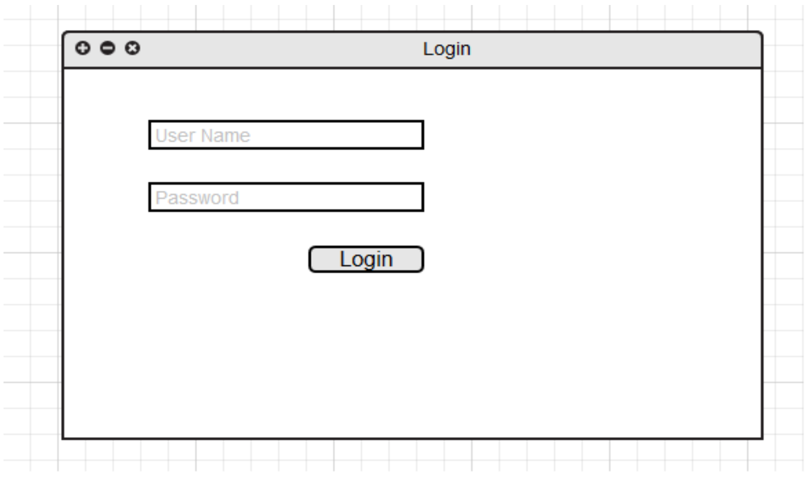


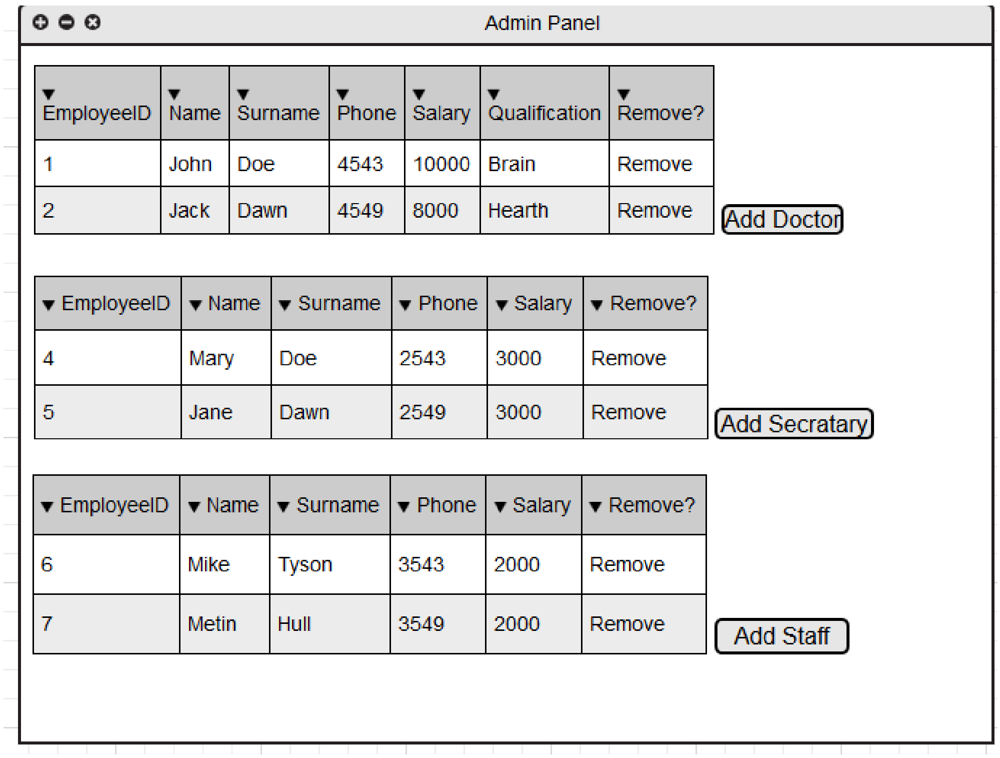
Figure 2 – Login Page

This is common login page for both admins and employees of the system. While the appearance is the same for all user types, the URLs will be different. Admins can reach this page by giving the URL with '…/admin.php'. On the other hand employees of the system can login after clicking login button on the main page. As stated above, the appearance of the login pages are the same and all user can login system with their username and password.

The SQL queries that will be used for this screen are as follows;

Login : *SELECT COUNT(*) FROM EMPLOYEE WHERE employeeID = givenID*
and
password = givenPassword

3.3. Admin Panel



EmployeeID	Name	Surname	Phone	Salary	Qualification	Remove?
1	John	Doe	4543	10000	Brain	Remove
2	Jack	Dawn	4549	8000	Hearth	Remove

EmployeeID	Name	Surname	Phone	Salary	Remove?
4	Mary	Doe	2543	3000	Remove
5	Jane	Dawn	2549	3000	Remove

EmployeeID	Name	Surname	Phone	Salary	Remove?
6	Mike	Tyson	3543	2000	Remove
7	Metin	Hull	3549	2000	Remove

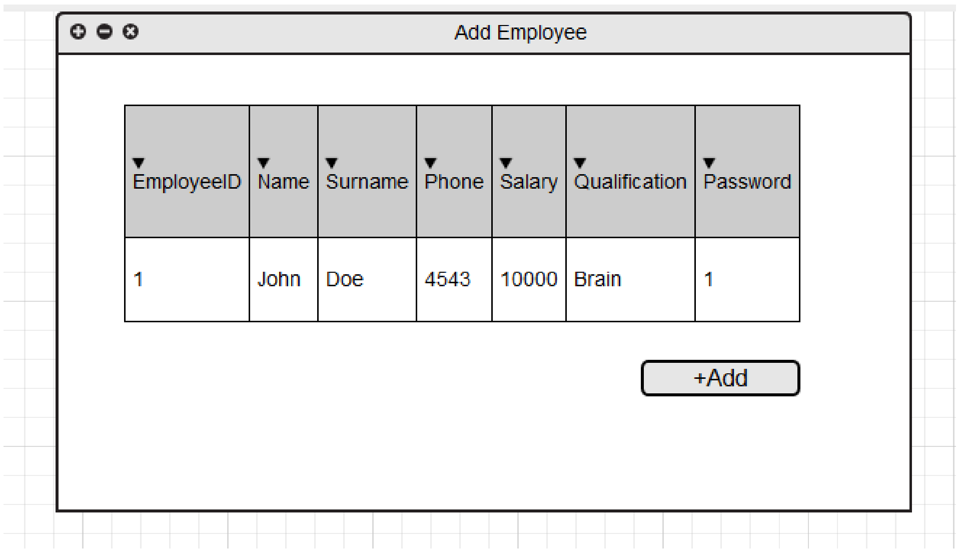
Figure 3 – Admin Panel

This is the page will be shown only admins of the system. The admins who login the system successfully can see this page. After login the system, they will be see current registered doctors, secretaries as well as staffs, namely all of the employees of the system. They can add or delete employees. There will be a column for each employee, which contains remove button that can admin click. After admin clicks remove button, the records are deleted from corresponding tables and page redirected to itself again. For example, if a doctor is deleted, it will be deleted from both doctors and employees tables. Moreover admin can add employee to tables. Admin can click add doctor, add secretary or add staff. They can also logout from system.

The SQL queries that will be used for this screen are as follows;

Remove: *DELETE FROM Employee WHERE employeeID = givenEmployeeID*
DELETE FROM Doctor WHERE employeeID = givenEmployeeID
DELETE FROM Secretary WHERE employeeID = givenEmployeeID
DELETE FROM Staff WHERE employeeID = givenEmployeeID

3.4. Add Employee Page



EmployeeID	Name	Surname	Phone	Salary	Qualification	Password
1	John	Doe	4543	10000	Brain	1

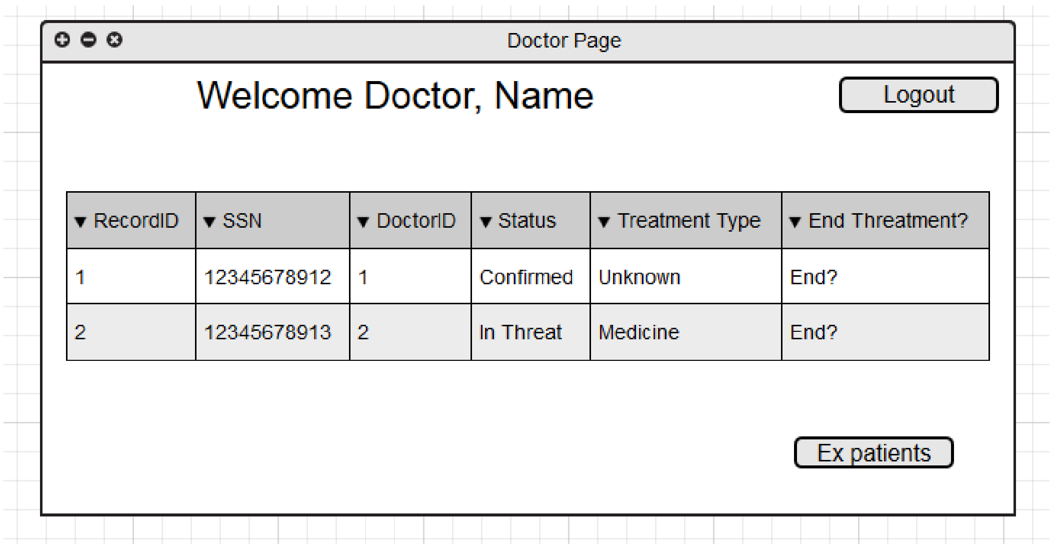
Figure 4 – Add Employee Page

This is the page where admins can add employee. The appearance for all employees is same with figure 4, however URLs will be different. The admins who login the system successfully can add employee after filling form. After add operation they will direct to admin panel.

The SQL queries that will be used for this screen are as follows;

ADD: *INSERT INTO Employee VALUES(.., given values, ..)*
INSERT INTO Doctor VALUES(.., given values, ..)
INSERT INTO Secretary VALUES(.., given values, ..)
INSERT INTO Staff VALUES(.., given values, ..)

3.5. Doctor Page



▾ RecordID	▾ SSN	▾ DoctorID	▾ Status	▾ Treatment Type	▾ End Threatment?
1	12345678912	1	Confirmed	Unknown	End?
2	12345678913	2	In Threat	Medicine	End?

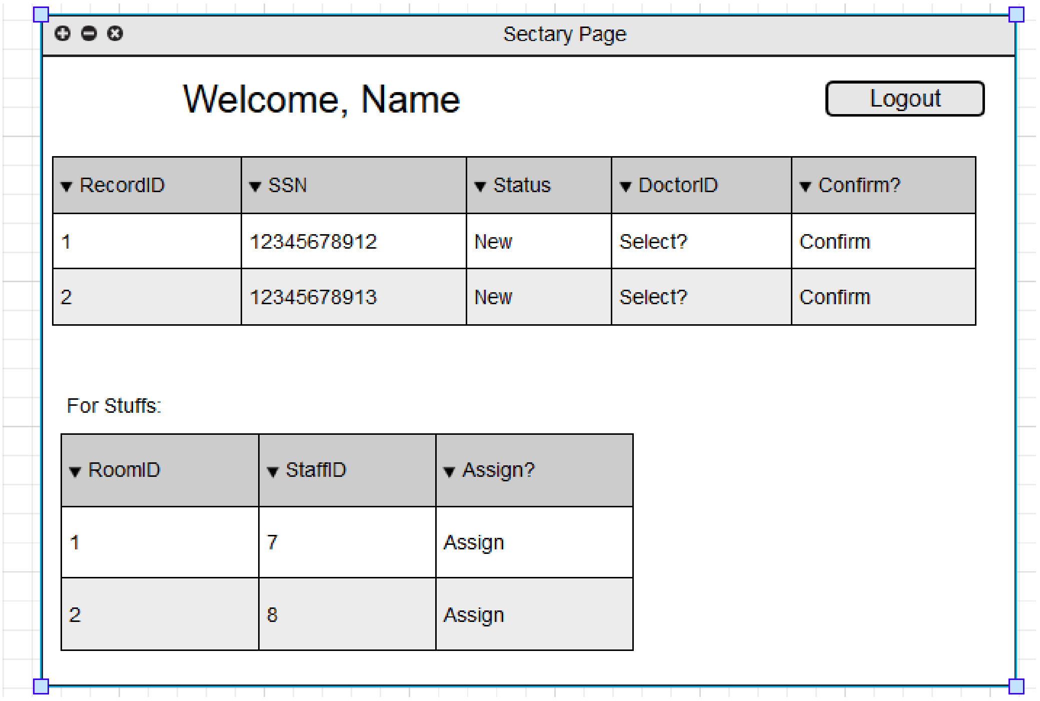
Figure 5 – Doctor Page

This is the page that will be shown to doctors who login to system successfully. After login, doctors can see their current patients together with their status. They can chance the status of the patient or they can end patients' treatment after selecting the treatment method. They can select room method so that patients can stay in hospital. They can select medicine or select no need for treatment. After selecting treatment method they should change the status of the patient to end treatment unless room is selected. If room is selected, then a random available room will be reserved for the patient. After end treatment is selected current date is written to endDate field of the corresponding row of the Record table. Moreover doctors can see their ex(old) patients' records. They can also logout from system.

The SQL queries that will be used for this screen are as follows;

Set Treatment : *UPDATE Record SET treatment= treatment, where recordID = givenID*
End Treatment: *UPDATE Record SET endDate = sys.date, where recordID = givenID*
Set Status : *UPDATE Record SET status = newStatus, where recordID = givenID*
Show Ex : *SELECT * FROM Record WHERE doctorID = givenID*
ArrangeRoom : *INSERT INTO StaysIn values (.., given values, ..)*

3.6. Secretary Page



RecordID	SSN	Status	DoctorID	Confirm?
1	12345678912	New	Select?	Confirm
2	12345678913	New	Select?	Confirm

RoomID	StaffID	Assign?
1	7	Assign
2	8	Assign

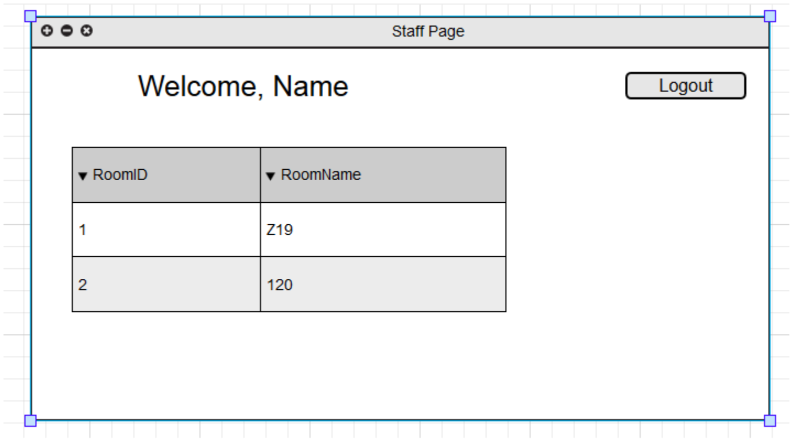
Figure 6 – Secretary Page

This is the page that will be shown to secretaries who login to system successfully. After login, they can see submitted patients forms. The status of the records will be new. The secretaries can assign doctor and change the status of the record to 'confirmed' from 'new'. There will a column for each record that contains confirm button which can be clicked by secretaries. Moreover they can assign/remove rooms to staffs. They can also logout from system.

The SQL queries that will be used for this screen are as follows;

Set Status : *UPDATE Record SET status = newStatus, where recordID = givenID*
Set Doctor : *UPDATE Record SET doctorID= givenID, where recordID = givenID*
AssignRoom : *INSERT INTO ResponsibleFor values (.., given values, ..)*
Remove Room : *DELETE FROM Employee where staffID = givenEmployeeID*
and
roomID = givenRoomID

3.7. Staff Page



Staff Page

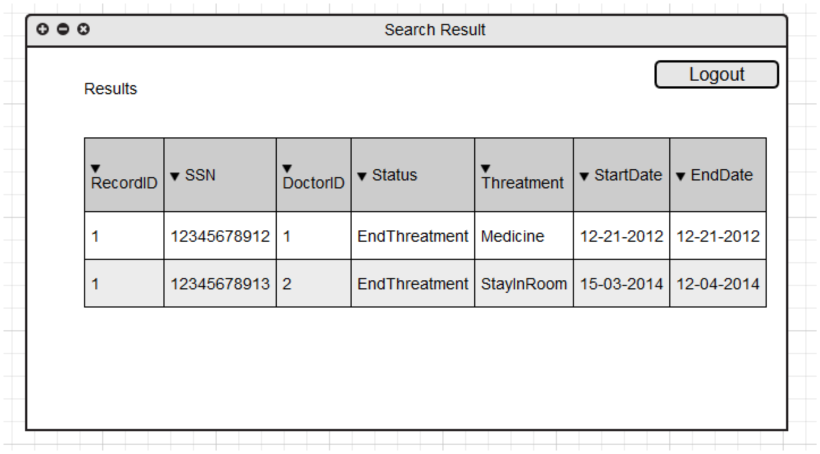
Welcome, Name

▾ RoomID	▾ RoomName
1	Z19
2	120

Figure 7 – Staff Page

This is the page that will be shown to staffs that login to system successfully. After login, they can see their assigned rooms. They can also logout from system.

3.8. Search Result Page



Search Result

Results

▾ RecordID	▾ SSN	▾ DoctorID	▾ Status	▾ Threatment	▾ StartDate	▾ EndDate
1	12345678912	1	EndThreatment	Medicine	12-21-2012	12-21-2012
1	12345678913	2	EndThreatment	StayInRoom	15-03-2014	12-04-2014

Figure 8 – Search Result Page

This is the page will be shown after search of record. The appearance will be same for both result of doctor searches and patients searches, however URLs will be different. The returned records will be shown.

4. Use Cases

In this part of the report, the use case and actions of the different type of users will be detailed.

4.1. Admin Usecase

The operations done by admins are shown in the figure 9.

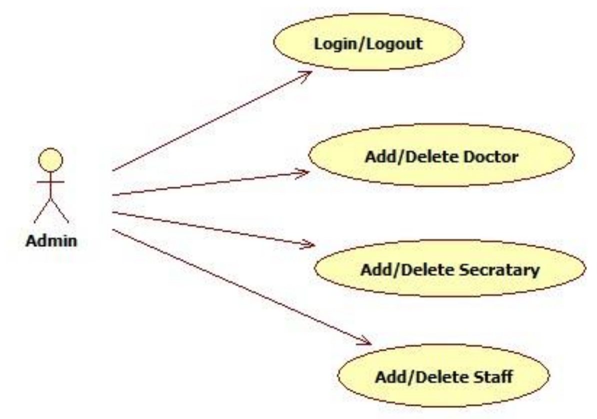


Figure 9 – Admin Usecase

According to figure 9, an admin can login, logout the system, add or delete employee to system. The added or removed employee can be doctor, secretary or staff.

4.2. Doctor Usecase

The operations done by doctors are shown in the figure 10.

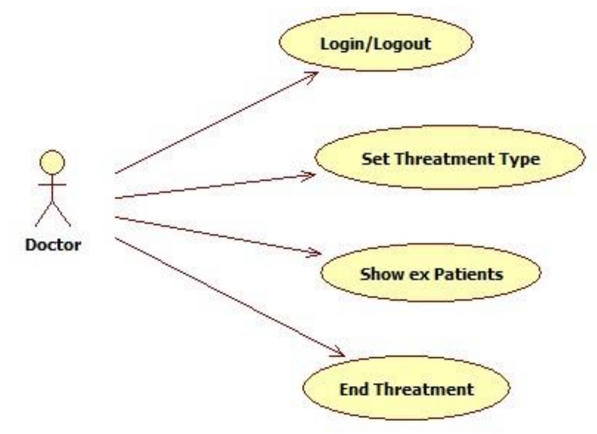


Figure 10 – Doctor Usecase

As can be seen in figure 10, a doctor can login, logout the system. In addition he can set treatment type or he can end treatment. Moreover he can monitor his old patients' records.

4.3. Secretary Usecase

The operations done by secretaries are shown in the figure 11.

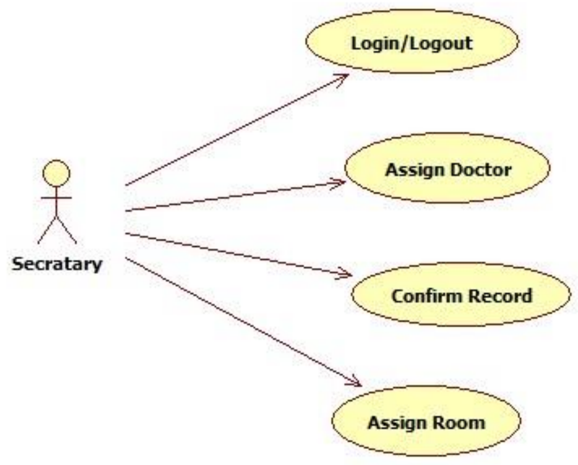


Figure 11 – Secretary Usecase

It can be observed from figure 11, a secretary can login, logout the system. In addition he can confirm records or he can assign doctor to patients. Moreover he can assign rooms to staffs where they are responsible about cleaning and food issues.

4.4. Staff Usecase

The operations done by staffs are shown in the figure 12.

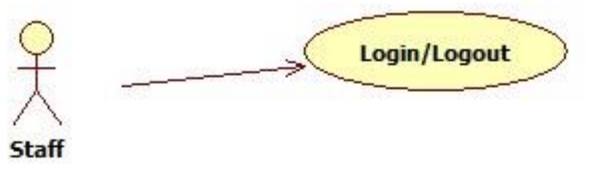


Figure 11 – Staff Usecase

It can be observed from figure 12, a staff can login, logout the system. When he logins to system, he automatically will see rooms which he responsible for.

4.5 Patient Usecase

The operations done by patients are shown in the figure 13.

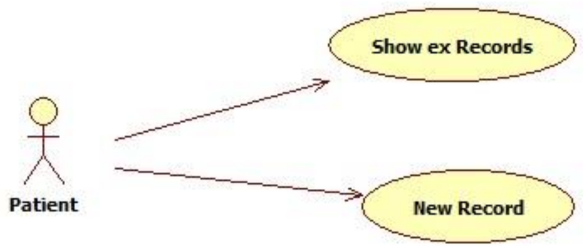


Figure 13 – Patient Usecase

As can be seen in figure 13, a patient can create new record without logging to system. After creating this record, he waits for record to be confirmed. Moreover he can search for their old records.