# Assignment - III

# Implementation of a Hospital Appointment System

Gönül AYCI 2016800003 Bogazici University May 17, 2017

### 1 Introduction

In this project, I want to implement a Hospital Appointment System with using PHP. In this project, we have two user roles who are

- Admin
- Patient

They have different rights. At the login step, depending on the their roles that they are directed to their respective home page which are *homepage\_user.php* for Patient and *admin.php* for Admin.

In this database (DB) system, I create five tables which are **Admins, Appointment, Branches, Doctor, and Patient**. I give more detailed information (e.g., structure) about their parameters in *DataBase* section.

After creation and implementation of the hospital appointment system (HAS), we need to write two stored procedure (SP). These should return all past and future appointments. So, we need to go the root of the DB, click **Routines**, and then add two routines. You can find the routines' details on my DB (exported sql file).

Finally, we should also write a trigger on doctor entity. You can also find the trigger's details on my DB (exported sql file).

### 1.1 Installation

I use metarials which is the youtube video that is shared by TA of this course. I download XAMPP from https://www.apachefriends.org/download.html

To run the server, I use the terminal with cd XAMPP/xamppfiles/ sudo ./xampp start

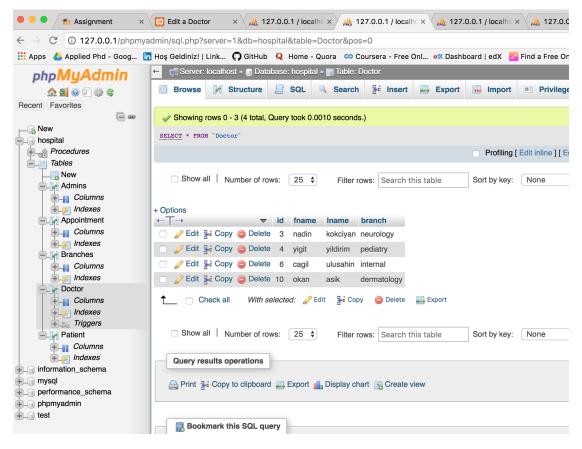


Figure 2.1: My database for a Hospital Appointment System (HAS)

# 2 DATABASE

To create the HAS, I think about the relations between Doctor and Patients. First of all, I create **Doctor, and Patient** types . Both of them need to save appointments, so I create the new type which is **Appointment**. Simultaneously, I need to create the **Admins** type. Finally, doctors have branches, so I create the **Branches** type which has only two parameters that are *id, and name* 

In the ER diagram, I show the relationship between types, and give information about their paramaeters. In addition tot his, I add the numbers which represent many-to-many, many-to-one relations. For instance, in generally, a branch has many doctor, but doctor has only one branch.

### <u>E-R Diagram of</u> Hospital Appointment System

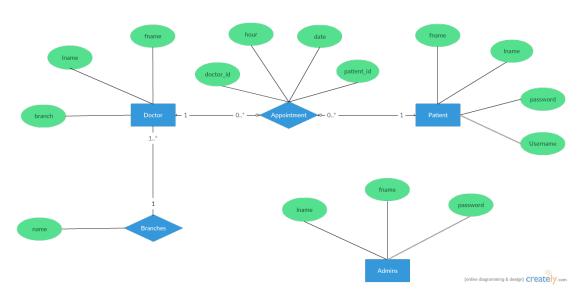


Figure 2.2: Entity Relationship (ER) diagram of the database

# 3 INTERFACE

In this section, I share screenshots of the system which runs on localhost.

At the *registration, login, and logout* steps, I implement these functionlities with three php files which are **index.php, registration.php, login.php, and logout.php**.



(a) Login page: index.php

(b) Registration page: registration.php

Figure 3.1: Home page of both registered and unregistered users



Figure 3.2: Enter invalid username or password



Figure 3.3: Home page for both a patient and an admin

(b) admin.php

#### 3.1 REGISTRATION AND LOGIN FUNCTIONALITIES

In this DB system, there are two users. They can do different actions on HAS. From this point of view, I can implement two files which are **admin.php and homepage\_user.php**.

#### 3.2 PATIENT FUNCTIONALITIES

In this HAS, Patient can

- Make an appointment
- Edit an appointment
- · Cancel an appointment

For these functionalities of Patient, I implement some files which are

- makeAppointment.php, and makeApp.php
- editAppointment.php, and editApp.php
- cancelAppointment.php and cancelApp.php

In the implementation part, I use the 96 character string for saving the hours of appointments. Its all characters are initially zeros. Each of it's characters represent the time interval (e.g., 8:00-0, 8:05-1, 8:10-2, ...). I also use an array to save the days.



Figure 3.4: Make an appointment page for patients



Figure 3.5: Edit an appointment page for patients



Figure 3.6: Cancel an appointment page for patients

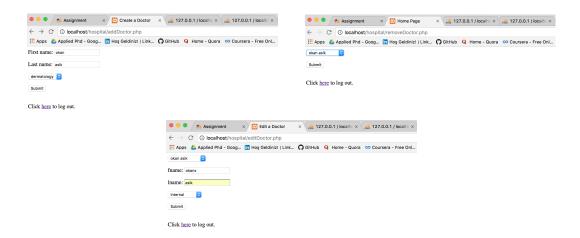


Figure 3.7: Admin can create, delete, and edit a doctor

# 3.3 Admin functionalities

### In this HAS, Admin can

- Add / Remove / Edit a doctor
- Add / Remove / Edit branches

For these functionalities of Admin, I implement some files which are

- addDoctor.php, createDoctor.php
- removeDoctor.php, deleteDoctor.php
- editDoctor.php, modifyDoctor.php
- addBranch.php, createBranch.php
- removeBranch.php, deleteBranch.php
- editBranch.php, modifyBranch.php

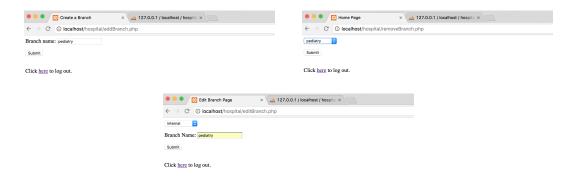


Figure 3.8: Admin can create, delete, and edit a branch

# 4 CONCLUSIONS

In conclusion, we can use PHP, HTML, and SQL for this project. I simply implement this project which does not include any CSS tag, and so on. In addition to implementation, I learn *php-MyAdmin* and creation of SP, and Trigger.

There are some future works for this project. For instance, I can use CSS to show the system in a more user-friendly, and elegant way. Moreover, the doctor and patient data can be retrieved from one of the Hospital on the Internet. In this version of the project, I use names of my lab members as a doctor or patient.