Patient Scheduling Subsystem

Michael Than
CS 3560-01
Cal Poly Pomona
Pomona, CA
michaelthan@cpp.edu

Hendrick Nguyen
CS 3560-01
Cal Poly Pomona
Pomona, CA
hendrickn@cpp.edu

Andrew Gonzales
CS 3560-01
Cal Poly Pomona
Pomona, CA
amgonzales@cpp.edu

Paul Sarmiento
CS 3560-01
Cal Poly Pomona
Pomona, CA
pbsarmiento@cpp.edu

Nathaniel Chandler
CS 3560-01
Cal Poly Pomona
Pomona, CA
ncchandler@cpp.edu

Abstract

Subsystems can play a very essential role when it comes to how well the overall systems itself can run. Such an example includes a patient scheduling subsystem. The significance of this subsystem is the ease of use for all the stakeholders who will be directly involved with this system. For this particular subsystem, the stakeholders who will be closely involved include the patients, doctors, nurses, medical assistants and receptionists. Receptionists will be the main users of the system. They will be able to use most of the abilities such as create/delete/edit an appointment and add a patient. This system will be a subsystem in a larger healthcare system that includes other subsystems such as medical records, billing, and other subsystems. The goal of this system will be to create something for receptionists to easily schedule patients and create patients. Also creating an easy to view schedule for doctors and medical employees.

Keywords—subsystem, users, stakeholders

I. Introduction

In hospitals and other medical facilities, patients need to be assigned appointments at specific times with their preferred doctor to run smoothly. A subsystem has to be created in order to be able to handle a patient that may be new or returning. Any new patients must first have to provide all the necessary information to be processed first. This will take a few days since the billing company of each medical facility will need to verify that the patients' insurance will be able to cover their appointment. Any other additional payments that needed to be made must be provided by the patient themselves. Once the new patient is verified, they will be allowed to make their appointment. By this point the facility will now have any previous medical records of their new patient. For returning patients, they can immediately make their appointment without having to go through this process because they are already an established patient in the system. Making an appointment can be done through either the phone or in person.

Since sensitive information will be able to the users of this subsystem, anyone trying to gain access will have to provide their login information. Most users with access will be able to view the day, time, and which doctor each patient is assigned to. Receptionists are the only ones who will be allowed to create a new patient, and either make or reschedule an appointment. All the data and information in this subsystem will of course be stored in a database at the medical office.

II. Analysis and Design

A. Stakeholders

Stakeholders are persons that have interests in the successful implementation of the patient scheduling system. Persons that have interests in our system are ones that have direct contact with our system, financial interest with our system, or health interests. The stakeholders consists of:

A. Development Team(Internal)

- Developers
- Management

B. General Public(External)

- Regular Patients

C. Employees(Operational)

- Doctors
- Nurses
- Medical Assistants
- Receptionists

D. Outside Funding(Executive)

- Government
- Outside Investors
- Private Contractors

B. Interaction with other systems

As mentioned earlier the patient scheduling system is a subsystem of a whole healthcare medical system. The scheduling system would interact with the medical records

system and as well as the billing system. Patients would have their appointments and patient profile information managed by the scheduling system. The patient's medical records would be managed on the medical records system which stores the patients sensitive records of their labs, procedures, operations, diagnostics, imaging, and doctor notes. The medical records system would be the system to obtain medical history of new patients in the office. The patient's insurance and payment information would then be managed by the billing system. The billing system manages the patients' statuses on their insurance and any other activity related to payment. New patients would also have to get their insurance information approved with the billing subsystem in order to be approved for any appointments. All of these subsystems together create a whole healthcare system for healthcare workers to use.

C. Use Cases

Use cases are activities that the system performs in response to a user or a time. In our system, there are external events as well as temporal events. The use cases usually have an actor and the actors in our system will be the receptionists, medical employees, doctors, and system. The receptionists will be able to add a patient into the system as well as schedule an appointment for a patient. They would also be able to cancel appointments or change the appointment of a patient. Receptionists will also have the basic use of viewing the schedule. The ability to view the schedule is also available for doctors and medical employees since they do not have permission to create an appointment or add a new patient. All the use cases mentioned are all external events, which means they are events that occur outside the system by an external actor. The other type of event in the system is a temporal event. This event occurs when a point in time is reached. The temporal events in this system are alerts for the receptionists. These alerts tell the receptionists to call the patient because they have an upcoming appointment, or they have missed an appointment and they need to call the office back to reschedule. These use cases also show the accessibility of each actor of the system. Doctors and medical employees are only allowed to view the schedule to see the appointments. Medical employees consist of everybody else in a medical office who are not doctors or receptionists. They could be medical assistants, nurses, or medical records clerks viewing the schedule to prepare for upcoming appointments of patients. While the receptionists have the most access to this system, giving them the ability to create, cancel, and edit an appointment. The scheduling subsystem is basically the responsibility of the receptionists to maintain.

D: Different Patient states

Throughout the process of scheduling appointments a patient can be in many different states. This can be shown in a state machine diagram. The first step for the patient is to call the office. Once the patient has reached the office their state will change to inCall. Next, the state of the patient changes based on whether or not they have been to this medical office before. If they have been to this medical office before they would simply have to create an appointment with the scheduler and then the patient state would be hasAppointment. If the patient has never been here before they would have to have the receptionist create a patient account for them, moving them to the newPatient state. Once they have given the receptionist all of the info needed, the patient will wait a few days for their insurance information to be approved, changing the state of the patient to become a pending Patient. Once the patient information has been verified, they will become a verifiedPatient. The verifiedPatient now has the ability to create an appointment. The patient will receive a call from the receptionists after the billing department has notified the receptionists that the patient is verified. Then, finally move the patient to hasAppointment state. After the patients have their appointment setup and check in to their appointment then the state machine diagram would end.

E. A Fully Developed Use Case Description for Create An Appointment

To demonstrate a fully developed use case in this system, the create an appointment use case will be used as it is the most important use case in this subsystem. The scenario of this use case is when a patient calls or requests at the front desk for an appointment to be made. The two actors of this use case are the receptionists and patients. The stakeholders are the patients, receptionists and doctors. The postcondition of this use case would be that an appointment is made and the patient is expected to show up on time for the appointment with the doctor they are assigned or they requested. The flow of activities would start with the receptionist asking the patient if they are a new patient or existing patient. Then if they are new patients then the receptionist would add the patient information into the system. Then when the patient's insurance information is verified, the receptionist would create an appointment for the patient on the system. The final activity would be the system displaying all the appointment information and confirming with the patient that it is all correct.

D. UML Activity Diagram

The activity diagram shows specific actions taking place during the scheduling of patients. It is based on the flow of activities from the fully developed use case description. This diagram will continue to use the create an appointment use

case mentioned previously. The diagram starts off with the patient making the call to the medical office's receptionist desk. Once the receptionist receives the call they determine if they are a new patient or not. If they are a returning patient, they go straight to scheduling the appointment. If they are a new patient, they first go to create a new patient profile. After creating the patient profile, simultaneously, the system verifies the patients payment method/insurance and also obtains the patients past medical history. After the information is verified the receptionist calls back the patient to confirm their information is verified, and then they are allowed to schedule appointments for the patient.

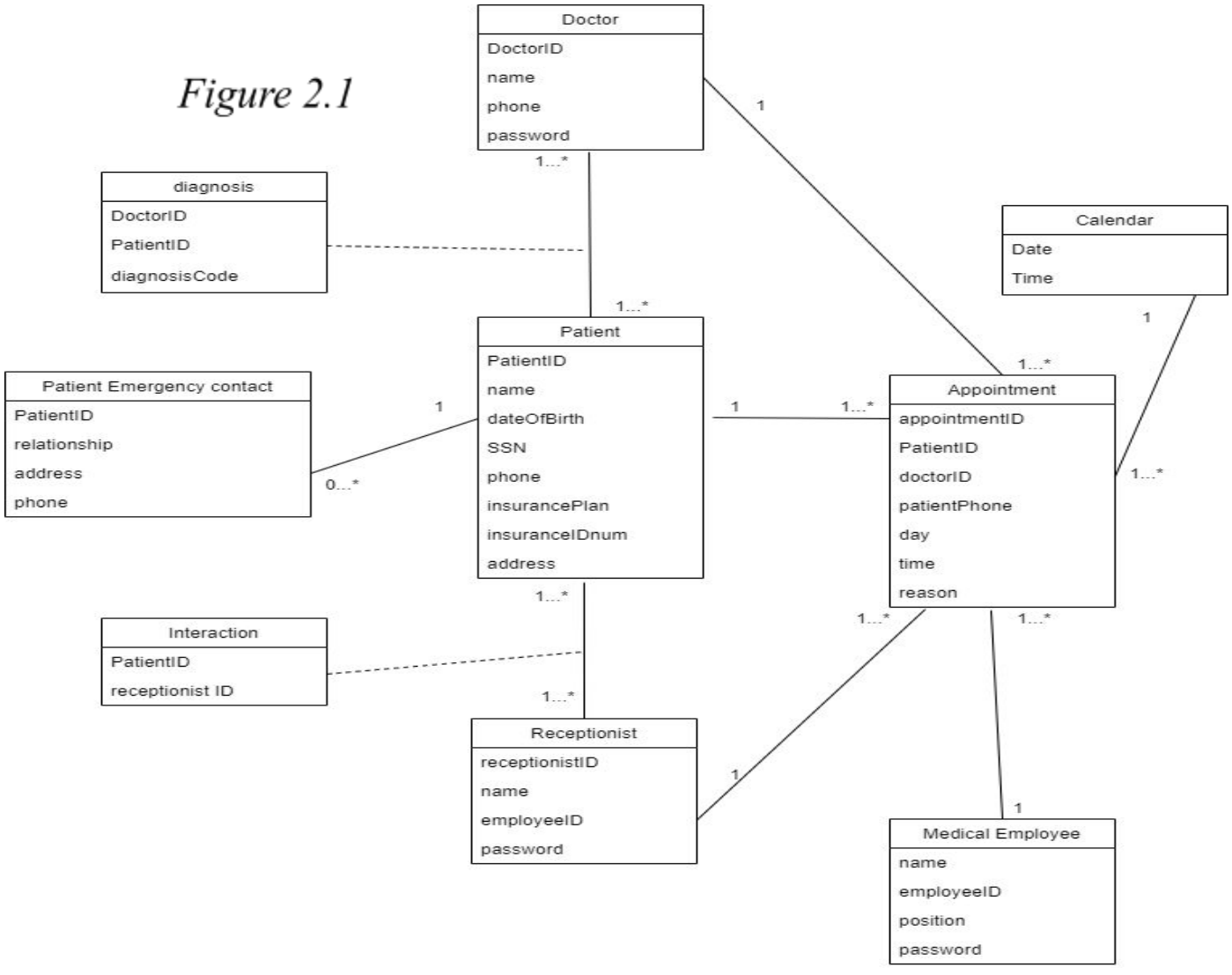
E. Sequence of Events

A system sequence diagram is also based on the fully developed use case description. We will continue to use the create an appointment use case in our example. In aSystem Sequence Diagram, we have the receptionist as our actor with the system. The receptionist first starts with determining if this is a new or returning patient. If they are new, they will go into the alternative section to create a new patient profile. The receptionist would use the createPatient function and input the patient details. After entering the information the system would return the details entered by the receptionist and a verification message once the patient's information is verified. After the alternative section the only option left is to use the createAppointment function. After entering the information for the appointment, the system verifies the appointment and returns the information on the screen with the appointment details. Then the information of the appointment will stay in the schedule until the appointment of the patient is over.

F. Domain Class Diagram

The domain class diagram is a class diagram that includes classes from the problem domain, so it will not include methods. They describe objects in the problem domain and their attributes. They will also have connections to other objects labeled with multiplicities. Figure 2.2 will show the domain class diagram of our subsystem. The patient will be in the middle of this diagram. The patient will have their personal information stored in the database and have a patient ID created for them. The doctors will have a doctor ID for

Figure 2.1



themselves and have their information stored. The appointments will have its own ID as well as having patient ID and doctor ID as foreign keys. The appointment information will also be stored. The patient emergency contact will have the patient ID as a foriegn key and later a primary key will be generated for it in the database. The medical employees and receptionists will all also have their own ID and their information stored.

The interesting part of the domain class diagram is the relationships between the objects. The patient will have a one to many relationship with the patient emergency contact as they will have the ability to add many emergency contacts. The patient will also have a one to many relationship with appointments because only one patient can schedule a specific appointment, but a patient can schedule many different other appointments. The patient has a many to many relationship with the doctor, which causes for there to be an association class. This association class is the diagnosis class which contains the diagnosis of the patient from the doctor. The patient can have many doctors with many diagnoses. The other many to many relationship is with the patient and receptionist. The association class is the interaction class. This shows there was an interaction between patient and receptionists as there can be many patients interacting with many receptionists. The appointment class will also have a many to one connection with a calendar class which just is just a time slot class. The medical employee has a one to many relationship with appointments as they can view many appointments. All these interactions between classes illustrate the interconnections between objects in our subsystem.

G. Database Schema (figure 2.2)

As mentioned earlier, a database will be needed for this subsystem. For our database we will be using Microsoft Access Database. Necessary information such as appointment details, employee information, and login credentials are stored to keep the subsystem functioning properly. Our database schema is very similar to our domain class diagram. The difference is we added the primary keys and foreign keys to each of the classes that will have information stored in the database. A primary key is a unique ID for an instance of that object and every class in the database schema will have one. For example the patient class will have its unique PatientID and all the rest of the necessary information on the patient. The appointment class is also an interesting class in our database as it has multiple foreign keys. A foreign key is a primary key from another class that a class associates with. An appointment has its own unique ID as well as three foreign keys. A foreign key for patient, doctor, and calendar slot. Each of these foreign keys make a unique appointment object. Each appointment needs to have a patient connected to it, a doctor

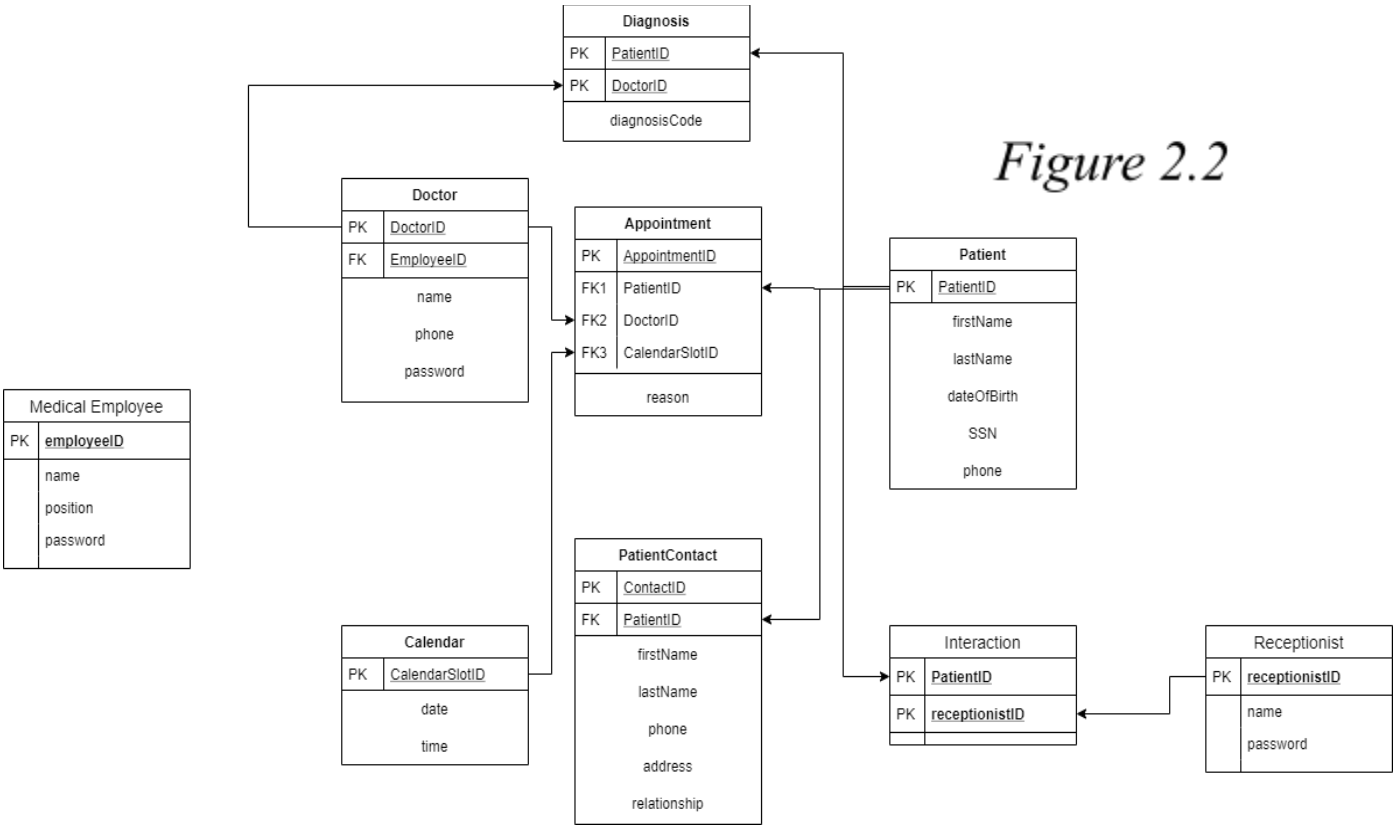


Figure 2.2

that the patient is having the appointment with, and the calendar spot the patient chose to schedule.

For the association classes, they will have two primary keys. The reason for that is because they will take the primary keys from the classes that they are associating and create a composite primary key from both of those primary keys. Both the interaction and diagnosis association class will have this characteristic.

III. Implementation

A. Resources/Technology used

To work on this project we used google drive to store all our relevant documents, and GitHub to store our code. We used draw.io to draw up our diagrams. To implement the patient scheduling subsystem, we chose to use Java as our programming language and used JavaFX to create the GUI for our system. As for our database, we opted to use Microsoft Access Database. We connected the database to our system by using JDBC access, which required adding extra jar files, to give the system access to the database file.

B. User Interface

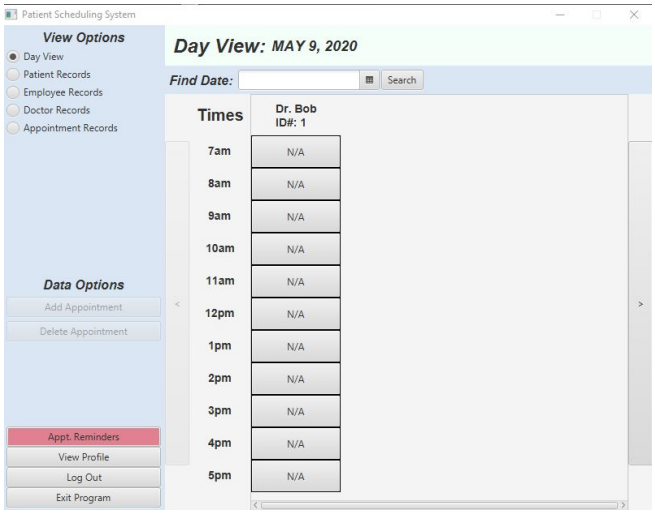


figure 3.1

The user interface (figure 2.1) starts with allowing the hospital employee and has a dropdown menu to state whether they are a receptionist, doctor, or medical employee.

Following logging in the user is shown a daily calendar with all the patients' appointments with their corresponding appointment time and doctor.

The top left of the window has a View Options section with 5 possible options that change the Data Options methods directly under it.:

- Patient Records
  -Allows receptionist to add appointment, edit appointment date/time, or delete appointment
- Employee Records
  -Allows adding, editing, or deleting employee records.
- Doctor Records
  -Allows adding, editing, or deleting Doctors records.
- Appointment Records and Day View
  -Allows receptionists to add, edit, or delete appointment

Then, at the bottom of the sidebar, there are options to either log in under a different account or exit the program.

C. Use Case Implementation

The only action available to those who are not logged in as receptionists, besides logging out, viewing the user's own profile, and exiting the program, is the ability to view appointment schedules under the day view and navigate to new days on the scheduler. If a user views their own profile, they can update their password. All other actions are reserved for the receptionists.

Under the day view, the receptionist has the ability to click on an appointment slot, and under the data options, the receptionist will be able to interact with the add or delete appointment options, depending on whether or not an appointment is booked on that time slot. If they choose to add an appointment, and it is not within two hours of the current time, then a new window will open up and allow the receptionist to file an appointment. If it is within two hours of the current time, the system will display a popup, and inform the user that appointments cannot be made within two hours of the current time. Likewise, clicking on a booked time slot will allow the receptionist the ability to delete it.

Under the patient records, the receptionist is given a table displaying each patient's basic information, with a button on the left-hand side that says edit. Under the Data Options will be the ability to add, edit, or delete a patient. If a patient record is not selected by clicking its corresponding edit button, the only option available under Data Options will be to add a patient; otherwise, the available options will be edit and delete. The edit patient will allow the receptionist to update a patient's data via a new window, and the delete option will provide a confirmation window before executing. The employee, appointment, doctor record pages work almost

identically, except for an option to edit.

While the menu has functionality shown by type of object, the code can be classified by its functionality: Object Creation, Object Management, Object Retrieval, and Object Removal. The reason that the code blocks can be classified this broadly is because the way in which they achieve their goals are very similar.

All Object Creation methods, once they receive the object that needs to be created, uses that data to complete a sql statement that inserts a new object of that type. All Object Management methods fill in an update sql statement with information received from the parameters. All Object Removal Methods fill in a delete statement based on the supplied id of the object in the parameters All Object Retrieval methods return an arraylist of the requested objects that is derived from a sql query. The only significant difference with the Retrieval methods are the ones for Appointments. There are two variants: one that returns all of the appointments for a given day, and one that returns all of the appointments for a given day tied to a specific doctor.

IV. Team findings/suggestions/Improvements

The system we have created is definitely not the final product we desire to have. There are many improvements I would add to the implementation of this system. The first improvement that we would like to implement is maybe change the database to mySQL instead of using Microsoft Access Database. Microsoft Database is very useful for a database that does not have constant changes to itself. When we have to add a new user to the system we would have to go into the database file and edit it, then save it to the database server. In mySQL, we could just edit it on the database GUI that is already created for us. It is much more accessible for us to maintain the database.

The other improvement we would like to make to the system would be better improving the aesthetics of the system. The GUI is a little bland and we feel that it could be improved, especially if we use XML files in JavaFX. The XML file would allow us to add more characteristics to the GUI to make it more pleasing to the user. Also reorganizing the way buttons and menu could also improve the accessibility of the system as a whole.

Another improvement we would change is maybe change the accessibility of each user, and add a patient records subsystem. The doctor users would be able to view only their own schedule as well as their own patients. Medical employees however would have full access to every patient and see every doctor's schedule in order to prepare for appointments. Receptionists would have access to only the patients' personal information but not their medical records. But the receptionists do have full access to the scheduling system to see every doctors' and every patients' schedule.

Obviously there are still more changes we can make but we do not know about until the system is actually put into use. Criticism from the users of the system would be essential in improving this system since they are at most contact with the system.

V. Conclusion

This project was definitely a challenging task. Many of the people working on this project did not have the knowledge to complete this project when we started, but along the way we learned extremely beneficial technologies that will help in the future. JavaFX, Microsoft Access, and ucanaccess will be resources we will be using in our future projects. This system is nowhere near where we want it to be as a final product. We wish to clean up the code and the GUI to make it even easier to navigate. We will probably even change technologies in the system, such as switching to mySQL for the database. The most important part of completing this project was definitely communication. Once everyone was doing what they were supposed to do and did them on time, we finally were onto something successful.

References

[1] Akionode, J. and Oloruntoba, S.,*Design And Implementation Of A Patient Appointment And Scheduling System*. [online] Iarjset.com. Available at: <https://iarjset.com/upload/2017/december-17/IARJSET%203.pdf> [Accessed 9 May 2020].