Design and Analysis of Algorithms

WECARE4U: ONLINE BOOKING SYSTEM FOR VIRTUAL APPOINTMENTS WITH DOCTORS

BSCS 2-2

Group 7

Members:

Hidden for privacy

1. Introduction and its Background

The world is evolving and we the inhabitants are as well. From our ancestors who created tools from stones to recent times where people can travel through the vacuum of space, from communicating through wordless sounds onto this time with the help of the internet, from using herbs and other methods to cure illness onto today with help from medicinal experts. These progressions help us live healthier and longer and through time the study of the human body keeps on progressing, where each time that passed by we learn more and more from the complexity of the human body, and this knowledge, helped our medical personnel find cures for the worst epidemics that happened before, from Antonine Plague to Black Death and now COVID-19 Pandemic; and with this ongoing pandemic, it is a risk to go out for a consultation or check-up with your doctors not just risking your life but also your relatives and others around you. Luckily with our progression, there are other ways to communicate with someone wherever you are or they are; with the help of the internet.

All over the world, one of the fastest-growing industries is health care. Traditionally, medical appointments are usually taken on the phone or personally visiting the hospital. This traditional way of making appointments requires both the scheduler and the one making the appointment. The availability of the scheduler, phone lines, and the doctor needed is not guaranteed which can be inconvenient to the individual making the appointment personally or through the phone. In addition, patients also have to wait in the queue while making the appointment and if ever the doctor cancels the appointment for some emergency reasons, the patient would not be notified unless they visit the hospital. On the other hand, if a patient sets an appointment, there's a probability that the patient will not show up for their appointment and the doctor would resort to overbooking time slots which causes a long line of patient waiting.

Timeless and efficient medical care delivery is demanded by many as time passes by and as the population continues to grow. An efficient health care delivery is based solely on the effectiveness of the scheduling system. A good scheduling system is an important component of healthcare and also it can encourage satisfaction to both patient and physician. Considering the downside of the traditional way of making appointments, a lot of companies nowadays came up with the idea of an online appointment booking platform. Patients can check the doctor's availability and also(which can) help doctors to save a remarkable amount of time. Also, the online appointment system works for 24 hours which can be accessed anytime by the users, and also can save(saving) their time as they would not need to call the hospital for booking(to book) an appointment in the middle of their busy schedule. Not only the users can save their time but also the staff working at the hospital as they would not have to spend all of their time booking and managing appointments through calls but instead they can spend their free time attending to more important and vital tasks.

Out of all these advantages, the users also encounter problems. Patients may not be able to identify what service or doctor they need and it can cause confusion and uncertainty which can lead to booking the wrong appointment type. The easiness and efficiency of an online appointment system can attract a lot of new customers and it can lead to a large volume of bookings being handled. Also, other existing systems are not organized (uno) which can be difficult for the new users and also patients that will be signing up especially if they are not computer people.

To address all of these drawbacks of the online appointment system, this study will focus on implementing the best algorithm to create an online booking and scheduling application for hospital services that provides an efficient method in making virtual appointments with doctors. The researchers will use the Oyelami Algorithm to sort the doctors according to their department which can help the user access the doctor they need and also help the patients to easily decide and select time slots. On the other hand, to detect the availability of the doctors needed and their time slots, the researchers will use the Binary Search Algorithm.

2. Related Works

2.1 Algorithms

2.1.1 Binary Search Algorithm

According to the study conducted by Asagba P., et al. (2010), Binary search is an algorithm that is based on the Divide-and-Conquer strategy. It requires the list to be sorted in either ascending or descending order. (Parmar, 2015) In this algorithm, the target data can be searched by dividing the list into two sections and disregarding the half where the data cannot be found. For each iteration, the list will be divided into half and the middle element will be compared to the target. This process will continue until the target data matches the middle element.

Using Binary Search will significantly reduce the time consumption and the comparisons needed as it eliminates half of the elements of the list from further checking. (Asagba P., et al., 2010) This will be primarily used for searching doctors, schedules, and available time slots for our appointment system. Figure 1 shows the representation of the binary search process.

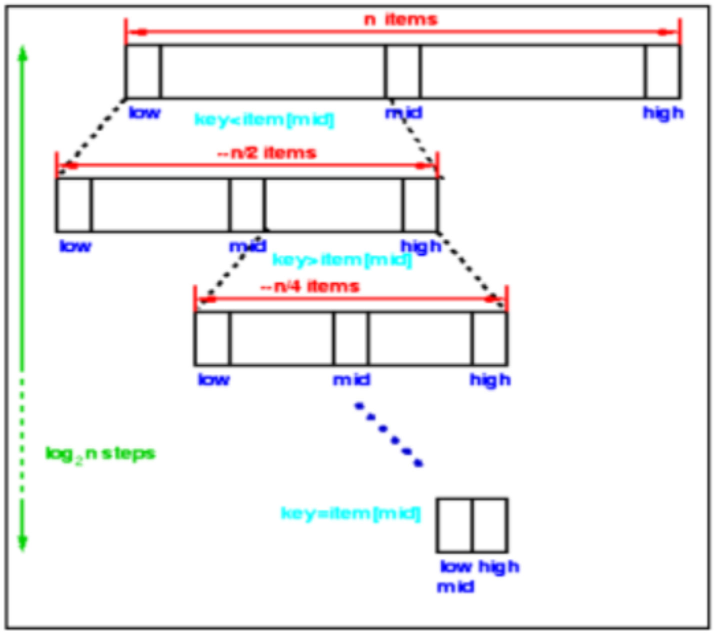
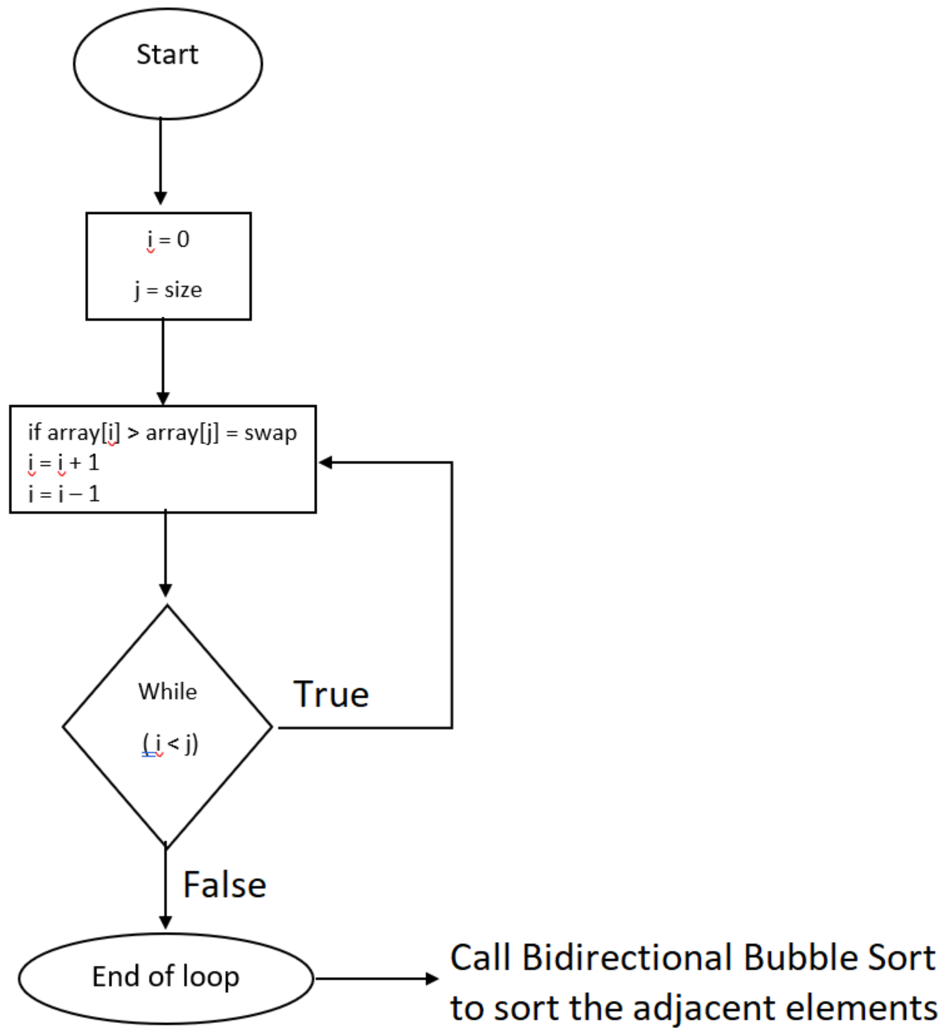


Figure 1: Binary Search Algorithm

2.1.2 Oyelami Sort Algorithm

According to Oyelami (2020), There are several sorting algorithms in existence. Some are well known while others are not so well known, but important. However, more and more are still being developed to take care of the weaknesses of the existing ones and to make sorting simpler to implement. In his study, he proposed a sorting algorithm called Oyelami's Sorting Algorithm, it is an improved sorting algorithm of a bidirectional bubble sort or cocktail sort based on Modified Diminishing Increment Sorting (MDIS). According to his study, Modified Diminishing Increment Sorting (MDIS) also partitions the items to be sorted into subsequences. It, however,(However, it) differs from Shell Sort in that it compares the first item on the list with the last and swaps them if they are not in order. If they are in order, they retain their positions. Next, the algorithm compares the second item on the list with the second to the last item. If they are not in order, they are swapped, otherwise, they maintain their respective positions.

This will help the researchers to sort out doctors, departments and available time slots for patient appointments in a timely manner since it is a more efficient sorting algorithm for multiple inputs. The diagram below shows the Oyelami Sorting Algorithm.


Start
i = 0
j = size
if array[i] > array[j] = swap
i = i + 1
i = i – 1
While
(i < j)
True
False
End of loop
Call Bidirectional Bubble Sort to sort the adjacent elements

2.2 Existing Systems

2.2.1 Clinic Appointment Scheduling System (CASS)

The Clinic Appointment Scheduling System (CASS) was developed by Atera Bintin Saifuddin. It is a mobile application which aims to manage the doctor and patient’s schedules without having the need to manually appoint a consultation. It also aims to lessen the labor of the clinic workers and decrease the waiting for appointment time. CASS consists of three entities: the patient, doctor, and admin. These entities are required to register before scheduling. Their information and their past consultations will be recorded to provide convenience for returning patients. (Saifuddin, 2018)

2.2.2 Smart Booking

The *Smart Booking*; *a concept for an automated booking solution* aims to deliver a proof of concept for a booking-tool which aids in the booking of patients into practitioners schedules, in a time- and resource efficient manner while still respecting patient preferences (Larsson, 2019). The solution presented in this paper provides the users/(and)patients the best possible time of available appointments for them with regards to their preferences whether they book one or more appointments in multiple schedules. The Smart Booking system uses a scoring function that is dependent on the needs and preferences of the users to find the best possible time of appointment for them. The system was tested through generating random schedules and appointments while the system processed the appointments using different configurations.

2.3 Telemedicine

Telemedicine is a broad term that is commonly defined as any medical activity that uses telecommunication devices to provide medical services. This includes various consultations such as diagnosis, education, and monitoring that use video conferences as their form of communication. Other forms include live monitoring of patients and telemonitoring, which involves a less trained person to conduct the needed procedures for the patient (Gogia, 2019). Telemedicine has been practiced since the 1950s in the United States. Although it was deemed impractical due to the high cost of telecommunication equipment and health workers are also not acquainted with these technologies, it also led to the improvement of telemedicine as a whole.(Jerant, 1997) As the earlier computer generations have been improved, telemedicine equipment around the 1990s has been considered as practical and can be applied in medical services. In 1991, an interactive-programmed consultations was established by The Medical College of Georgia. (Crump MD & Pfeil, MD, 1995) Over the years, telemedicine applications have now been extended and can be applied in various fields such as Public Health care, rural health care, nursing home care, and disaster medicine. (Berek & Canna, 1994)

Studies also show that the use of telemedicine consultations have a higher satisfaction rate compared to traditional on-site appointments. In a survey conducted in Xijing Hospital in China, 72% of their participants used their web-based appointment system (WAS) instead of the usual queueing method. There is also a significant decrease in waiting times for those who chose WAS. The average waiting time of the queueing method and WAS are 86 minutes and 12 minutes, respectively (Cao W., et al., 2011). According to the study by Pheng Zhao, et al. (2017), using WAS instead of traditional queueing methods has lowered the no-show rate of the patients and the waiting time for their consultations. The system is also “patient-centered” which improved the convenience of their access.

3. Objectives of the Project

The main objective of this project is to create an online booking and scheduling application for hospital services to provide an efficient way in making virtual appointments with doctors from the convenience of your home and to improve an existing system (Hospital Management System by Kishan Lal) through the use of algorithms.

This project aims to attain the following :

a. Develop a system that will maximize operation hours and reduce cost.

b. Develop a system for easy appointments and retrieval of data. Develop a system that provides easy and quick scheduling online appointments with doctors.

c. Develop a system that has an option of passive booking and real time booking for the users.

d. Develop a system that can detect the availability of the doctors and the time slots using binary search algorithm.

e. Develop a system that can sort the doctors by departments and their time slots for the patients to choose and pick easily. Develop a system that can sort the nearest doctor in the map for the real time booking. Both will be using the Oyelami sort algorithm.

f. Develop a system that can search in a client side process using a binary search algorithm instead of a database.

g. Develop a system that has a map feature where the distance tracking is implemented.

h. Develop a system that includes video conferencing for both passive and real time booking appointments.

Thus, this scheduling application will be created to automate and simplify the process for hospital management for all sizes of organizations.

4. Scope and Limitations of the Project

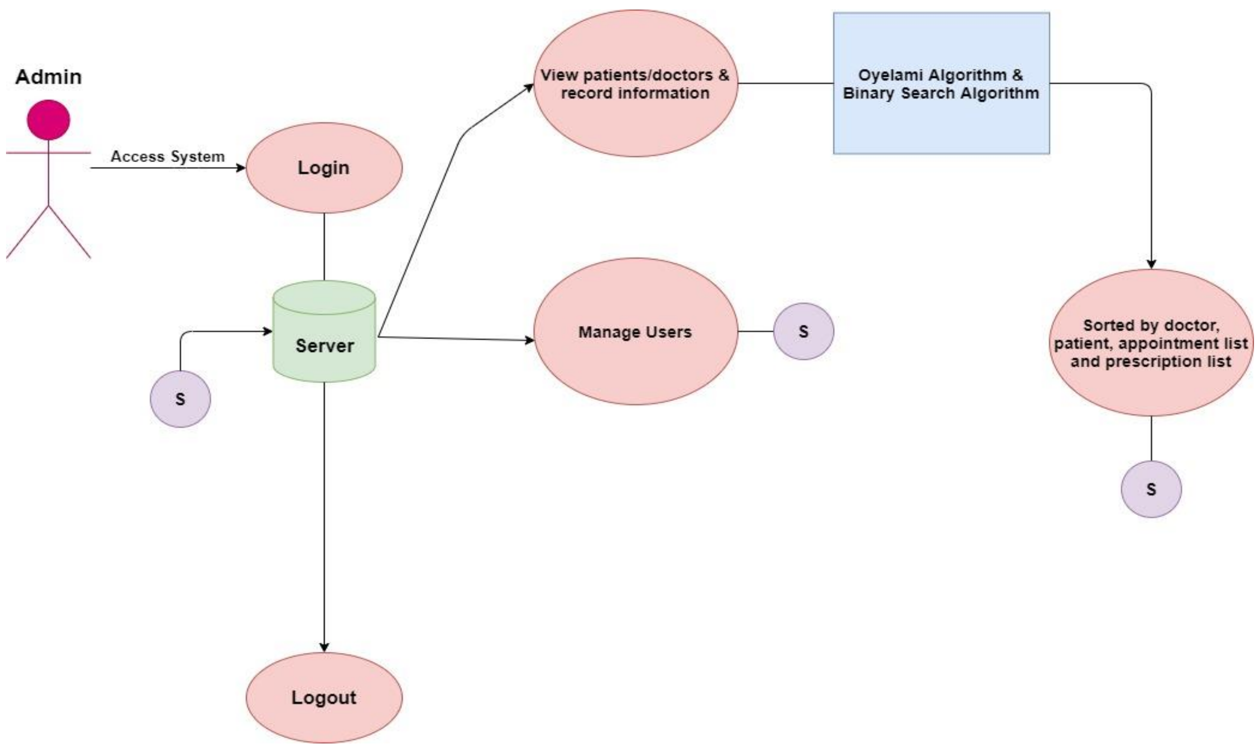
The entire project is only limited to the online booking of appointments with doctors. The project covers the process of online scheduling of appointments both passive and real time which involves implementation of searching and sorting algorithm.

The language that will be utilized for this project is Javascript and PHP to meet the requirements of the system.

The system is mainly for patients who will have online consultations and not for patients who will appoint a schedule for face-to-face consultation.

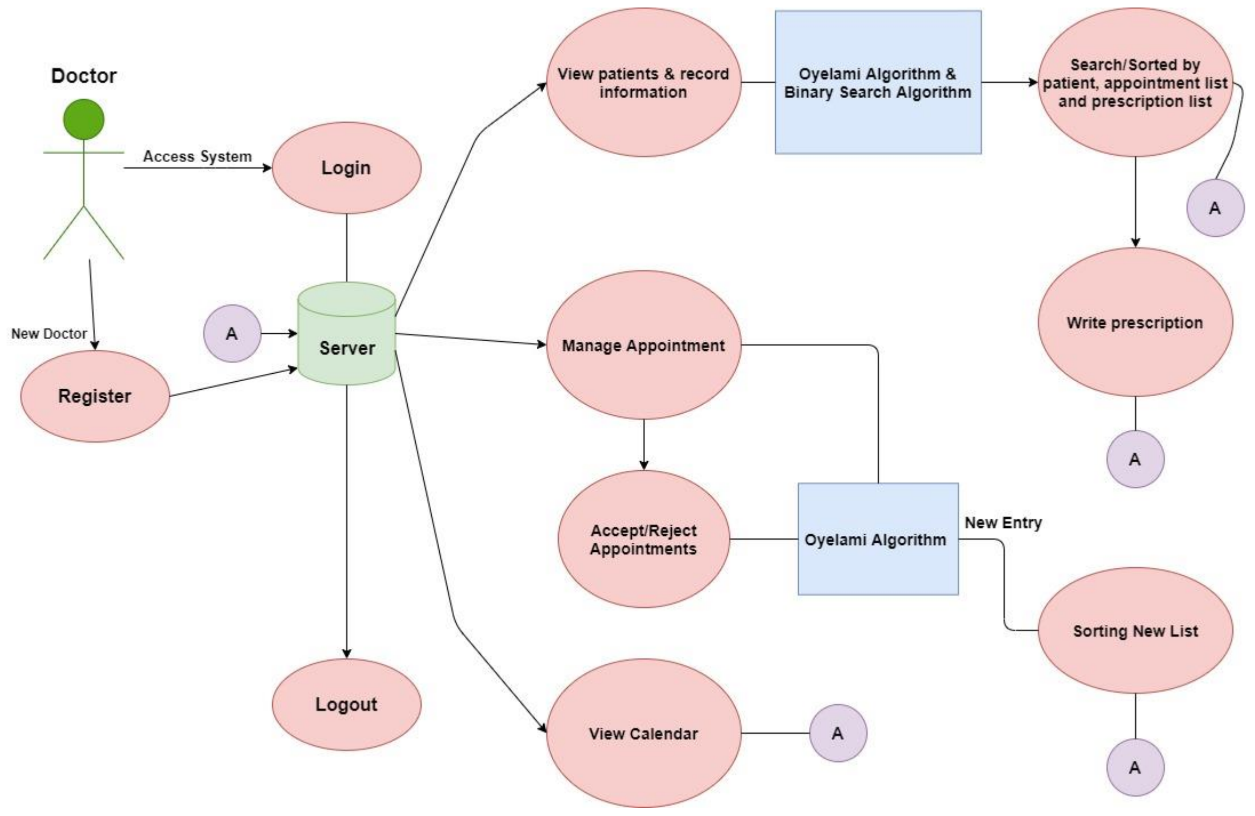
5. System Design and Methodology

Figure #1: System Architecture for the Admin as the User


Admin
Access System
Login
Server
S
View patients/doctors & record information
Oyelami Algorithm & Binary Search Algorithm
Manage Users
S
Sorted by doctor, patient, appointment list and prescription list
S
Logout

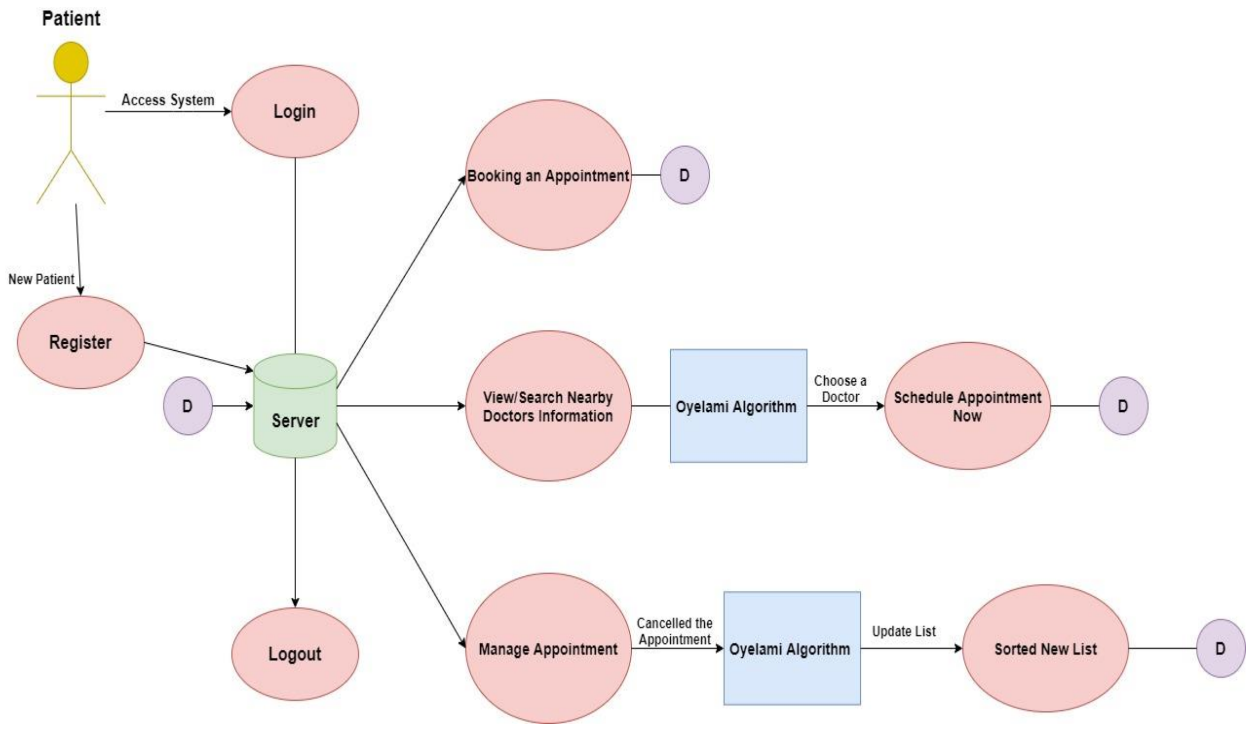
The figure above shows how the system works if the user is an admin. The admin must already have an account and they can access the system by logging in. After logging in, they have options to view the patients/doctors and record information, manage the users or log out. For viewing the information, they can search through the sorted list using the Binary Search algorithm. The list is sorted by doctors, patients, appointments and prescriptions by the use of Oyelami algorithm.

Figure #2: System Architecture for the Doctor as the User


Doctor
Access System
Login
New Doctor
Register
A
Server
Logout
View patients & record information
Oyelami Algorithm & Binary Search Algorithm
Search/Sorted by patient, appointment list and prescription list
A
Write prescription
A
Manage Appointment
Accept/Reject Appointments
Oyelami Algorithm
New Entry
Sorting New List
A
View Calendar
A

If the user is a doctor, the system operates as shown in the diagram above. If the doctor already has an account, they can access the system by logging in but if they don't have an existing account, they must first register to use the system. They can choose to browse the patients and record information, manage appointments, view calendar, or log out after logging in or registering. If the doctor chooses to view the information, they can search their patients and record information and they can also write prescriptions for the patients. Binary Search algorithm is used for searching the information and Oyelami algorithm is used for sorting the patients, appointments and prescriptions. The doctor can manage their appointments and accept or reject it if they want to. After managing their appointments, the Oyelami algorithm will work in the background and a new sorted list will be presented.

Figure #3: System Architecture for the Patient as the User



The system functions as illustrated in the diagram above if the user is a patient. If the patient already has an account, they can log in to use the system; but, if they don't, they must first create one. After logging in, the patient can choose to book an appointment passively or in real time, manage their appointments, and log out. For the passive booking process, a database will be used. The patients have doctors, their department, and appointments/schedules to choose from the database. With regard to the real time booking process, the patient can view and search nearby doctors' information, which is sorted by Oyelami algorithm. The patient can schedule an appointment as soon as they confirm the doctor they want to have a schedule with. The patient can manage their appointment by cancelling it. The Oyelami algorithm will update the list and show the new sorted list after the patient cancelled an appointment.

Figure #4: System Architecture for Oyelami Sorting Algorithm

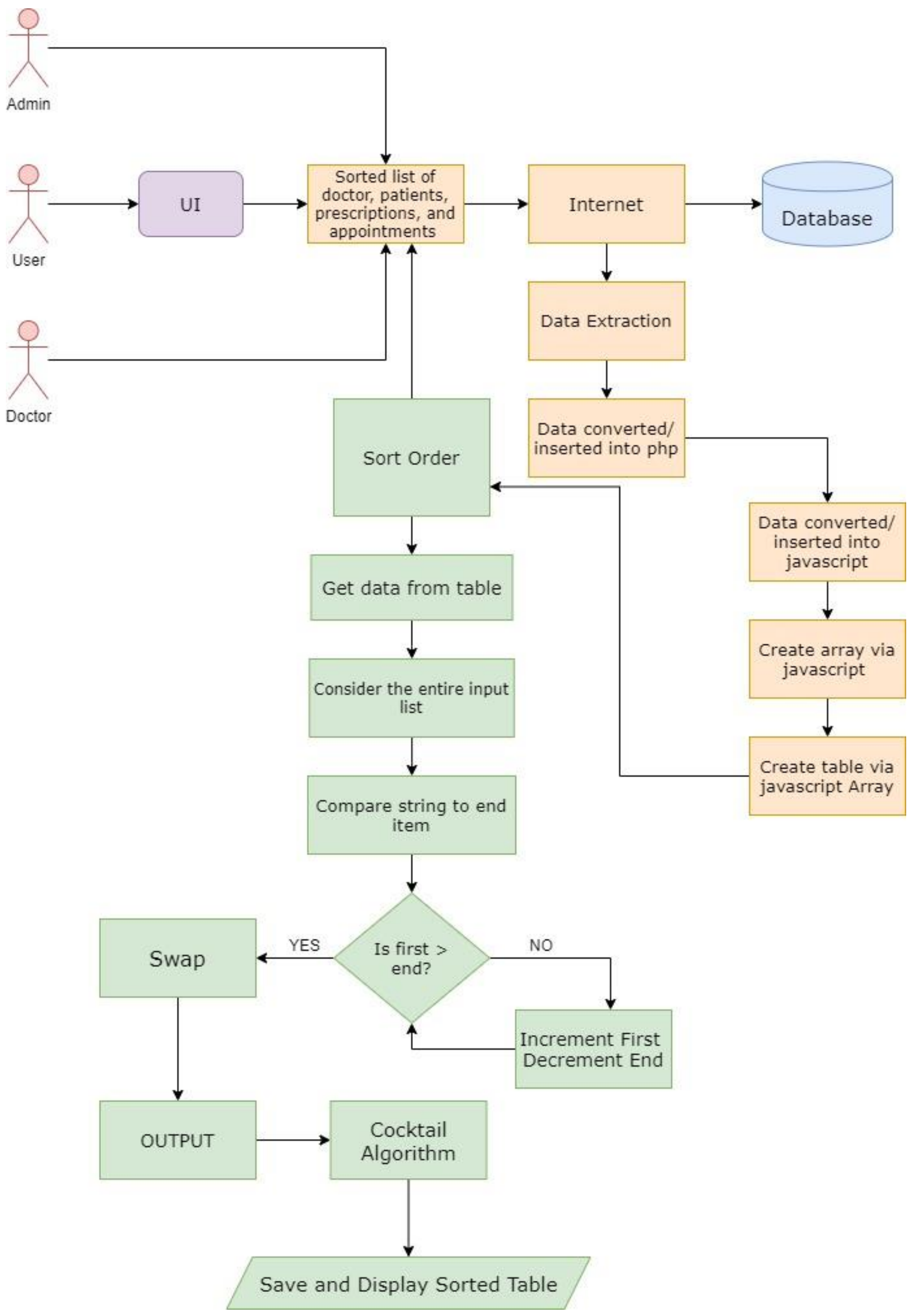

Admin
User
Doctor
UI
Sorted list of doctor, patients, prescriptions, and appointments
Internet
Database
Data Extraction
Data converted/ inserted into php
Data converted/ inserted into javascript
Create array via javascript
Create table via javascript Array
Sort Order
Get data from table
Consider the entire input list
Compare string to end item
Is first > end?
YES
NO
Swap
Increment First Decrement End
OUTPUT
Cocktail Algorithm
Save and Display Sorted Table

Figure above shows how the Oyelami algorithm sorts out the record of patients and doctors and their appointments. Every information that has been added or removed will be sorted out by the Oyelami algorithm in two processes. The unsorted list will be first retrieved from the database and will be transferred to an array in Javascript. The first and last elements of the array will then be compared. If the first element is greater than the last element, their positions will be swapped. Otherwise, they will remain in their positions and the next swap will occur with the element next to the first and the previous element of the last. This process will repeat until all elements are checked up to the two middle consecutive elements.

Once the first process has sorted out the information, the elements will proceed to the next process. The updated list will then be sorted out using the Cocktail Sort as part of the Oyelami Algorithm. The adjacent elements will be compared starting from left to right. The elements will be swapped if they are in the wrong order but will remain otherwise. Once it reaches the last element, the same process will occur but it will start comparing from right to left. This reduces the number of comparisons that the system must do to fully sort out the list. Every time the records are changed (e.g. Cancellation of appointments, Registration of a new patient/doctor) the Oyelami algorithm will continuously sort all this information. The updated list will then be displayed to the user's interface once it is sorted out.

Table 1. Comparison of Oyelami Sort, Bubble Sort, Insertion Sort, and Cocktail Sort's Performance Time in milliseconds (ms).

Number of Inputs	Oyelami Sort	Bubble Sort	Insertion Sort	Cocktail Sort
5000	154	665.10	308.10	248.5
10 000	600.70	2767.20	1024.60	983.90
25 000	5713.80	21708.20	7950.80	6300.40
30 000	8831.40	34176.40	12162.20	10151
50 000	30645.20	151 171	55770.20	33297.30

Table 1 shows that compared to other algorithms, the Oyelami Sort has shorter performance time. It shows that the Binary sort and Insertion sort is not ideal to be used for a larger number of inputs. The results also prove that the Oyelami sort which has the Cocktail sort as part of its process, is more efficient compared to using the Cocktail sort alone. This further implies that using the Oyelami Sort is faster, and therefore more efficient compared to the other algorithms.

```
//Oyelami Function

function Oyelami_sort(array, mapper){
        while(i<j-1) {                  ---------------------------------->n+1


        if (array[i] > array[j-1]){     ---------------------------------->n
        let temp = array[i];            ---------------------------------->n
         let tempMapper = mapper[i];            ---------------------------------->n
         array[i] = array[j-1];         ---------------------------------->n
         mapper[i] = mapper[j-1];       ---------------------------------->n
         array[j-1] = temp;             ---------------------------------->n
         mapper[j-1] = tempMapper;              ---------------------------------->n
         }//end of if statement
            i=i+1;                              ---------------------------------->n+1
                            j=j-1;      ---------------------------------->n+1
   }//end  of while loop
}//end of function Oyelami_sort         ---------------------------------->        10n+3
```

Time Complexity: O(n)

10n+3 ≤ 10n+3n, if n ≥ 1

10n+3 ≤ 13n

13 ≤ 13

```
//Cocktail function
function cocktailSort(arr,mapper) {
    //Start and end is used to keep track of where the beginning and the end of the
array is at
    //to determine where needs to be checked for sorting
    //Swapped is our conditional to check if everything is sorted
    let start = 0, end = arr.length, swapped = true;          -------------------------> 1

    while (swapped) {                                     -------------------------> n+1
        //Setting the flag to false, in case it is true from the previous iteration
        swapped = false;

        //Bubble sort from the left side of the array to the right, moving the largest.
        for (let i = start; i < end - 1; i++)             ------------------------->     n*
(n+1)
        {
            if (arr[i] > arr[i+1]) {                  -------------------------> n*n
                let temp = arr[i];                    -------------------------> n*n
      let tempMapper = mapper[i];                         -------------------------> n*n

                arr[i] = arr[i+1];                        -------------------------> n*n
      mapper[i] = mapper[i+1];                            -------------------------> n*n


                arr[i+1] = temp;                          -------------------------> n*n
      mapper[i+1] = tempMapper;                                    -------------------------> n*n
                swapped = true;                -------------------------> n*n
            }
        }
        //This is to update the end, so that next iteration, we don't have to check this
index.
        end--;                                     -------------------------> n
        //If everything is already sorted, we can break out of the loop early.
        if (!swapped)                              -------------------------> n
        {
            break;                                 -------------------------> 1
        }
        //Setting the flag to false, so it can be used for the next phase
        swapped = false;                           -------------------------> n
        //Reverse Bubble sort, moving the smallest to the front.
        for (let i = end - 1; i > start; i--)                    -----------------------> n*(n+1)
        {
            if (arr[i - 1] > arr[i])               -------------------------> n*n
            {
                let temp = arr[i];             -------------------------> n^2
      let tempMapper1 = mapper[i];                                 -------------------------> n^2
                arr[i] = arr[i - 1];           -------------------------> n^2
      mapper[i] = mapper[i-1];                                     -------------------------> n^2
```

```
                        arr[i - 1] = temp;                -------------------------> n^2
              mapper[i-1] = tempMapper1;                          -------------------------> n^2
                        swapped = true;                   -------------------------> n^2
                    }
                }

                //This is to update the beginning, so that next iteration, we don't have to
check this index.
                start++;                                          -------------------------> n
            }
        }                                                 -------------------------> 17n^2+7n+2
```

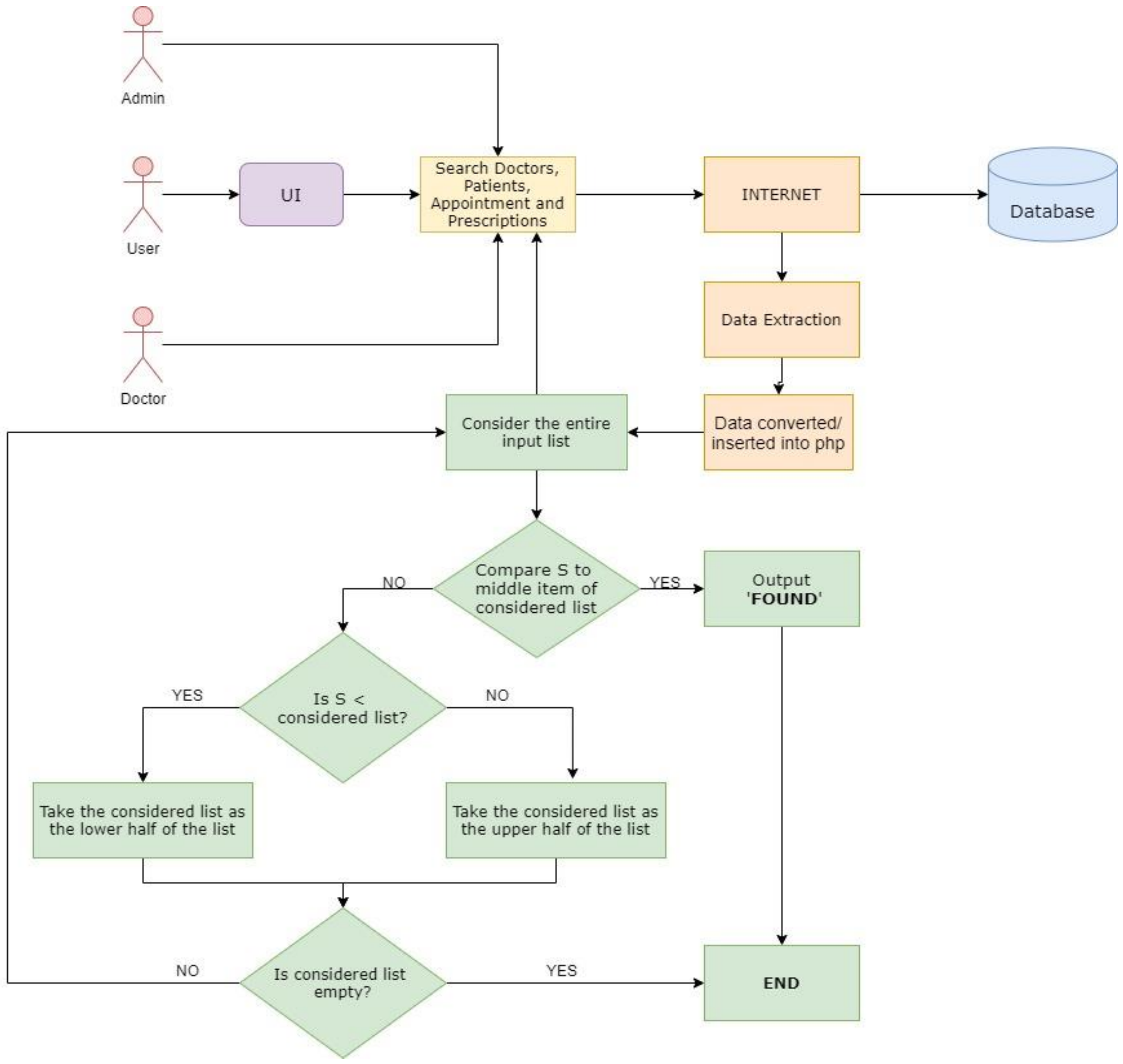
Time Complexity: O(n^2)

17n^2+7n+2 ≤ 17n^2+7n^2+2n^2, if ≥ 1

17n^2+7n+2 ≤ 26n^2

26 ≤ 26

Figure #5: System Architecture for Binary Search Algorithm


Admin
User
Doctor
UI
Search Doctors, Patients, Appointment and Prescriptions
INTERNET
Database
Data Extraction
Data converted/ inserted into php
Consider the entire input list
Compare S to middle item of considered list
NO
YES
Output 'FOUND'
Is S < considered list?
YES
NO
Take the considered list as the lower half of the list
Take the considered list as the upper half of the list
NO
Is considered list empty?
YES
END

The figure above illustrates the process of using the Binary Search algorithm. The Binary search is used to help the patients and doctors to search for their information when booking and managing an appointment. When searching, the system will retrieve the sorted list of information from the database. The value that the user has inserted will then be compared to the middle element of the list. If it matches, then the information will be displayed to the user. Otherwise, the system will compare if the input is greater or lesser than the middle element. If the input, for an instance, is lower than the element, then the system will consider the lower half of the list and disregard all the elements of the upper half. The process of comparing to the middle element and dividing the list will recur until the input is found or the list has been already emptied.

Table 2. Comparison of the performance times of Binary Search and Linear Search in milliseconds (ms).

Number of Inputs	Binary Search	Linear Search
3000	0.097	0.40
3500	0.10	0.50
4000	0.13	0.40
4500	0.13	0.47
5000	0.13	0.43

Table 2 shows the comparison of the searching algorithms, Binary Search and Linear search. This shows that both algorithms can work efficiently even with the increasing number of inputs. However, we can observe that Binary Search has a shorter performance time compared to the Linear Search. Using the Binary Search proves to be more efficient as it reduces the amount of elements to be compared in half for each of its iterations. Unlike Linear Search that requires the system to search for the input one by one starting from the first to the last element.

```
/*************Starting binary search ***********************/
  while (start<=end) {
          let middle = Math.floor((start + end) / 2); // defining the middle     ---------------> 1
          if ((txtValue[middle] == filter)) { //let filter = 3 || txtValue[2] = 1 and txtValue1[2] = 2
                                                                     -------------------------> n
          //propose solution this will add to the time complexity
           //but this is an displayer use case, not necessary to be
           //a binary search responsibility /start of block
           for (var z = 1; z<=tr.length-1 ; z++){                   -------------------------> n+1
                   if ( (txtValue[z] == txtValue[middle])  )        -------------------------> n
                    {    tr[middle].style.display = "";     }        -------------------------> n
                  else {                                             -------------------------> n
                         tr[z].style.display = "none";               -------------------------> n
                          tr[z].style.display = "not found";         -------------------------> n
                           } //correct close
        } //end for for loop z
```

```
      break;                                            -------------------------> 1

   } // end for if equivalent middle

   else if ((txtValue[middle] > filter) ) {             -------------------------> n

               end = middle -1;     } //end of less than      ----------------------> n+1

   else if   ((txtValue[middle] < filter) )  {          -------------------------> n

            start = middle + 1;                              -------------------------> n+1

     }//end of greater than

       else {                                           -------------------------> n

         break;       }                                      -------------------------> 1

  } //end of while loop for binary search               -------------------------> 13n+6
```

Time Complexity: O(n)

$13n+6 \le 13n+6n$, if $n \ge 1$

$13n+6 \le 19n$

$19 \le 19$

References

Asagba, P., & Osanghae, E. O. (2010). Is Binary Search technique faster than Linear Search technique? Scientia Africana, 9(2).

Berek, B., & Canna, M. (1996). Telemedicine on the move: health care heads down the information superhighway. Hospital Technology Series, 13(6), 1-65.

Cao, W., Wan, Y., & Tu, H. (2011). A web-based appointment system to reduce waiting for outpatients: A retrospective study. BMC Health Serv Res 11, 318. https://doi.org/10.1186/1472-6963-11-318

Crump, W. J., & Pfeil, T. (1995). A telemedicine primer. An introduction to the technology and an overview of the literature. Archives of Family Medicine. https://doi.org/10.1001/archfami.4.9.796

Gogia, S. (2019). Fundamentals of telemedicine and Telehealth. Google Books. https://books.google.com/books?hl=en&lr=&id=R165DwAAQBAJ&oi=fnd&pg=PP1&dq=gogia+2019&ots=HjiyWW7kXZ&sig=Ud4xHkpMUooAZaEm4-9TGRAvkPo

Jerant, A. F. (1997, April 1). Fundamentals of telemedicine. OUP Academic. https://academic.oup.com/milmed/article-abstract/162/4/304/4831711

Larsson, J. (2019). *Smart booking: Concept for an automated booking solution*. DIVA. https://www.diva-portal.org/smash/record.jsf?pid=diva2:1300728

Parmar, V. P., & Kumbharana, C. K. (2015). Comparing Linear Search and Binary Search Algorithms to Search an Element from a Linear List Implemented through Static Array, Dynamic Array and Linked List. International Journal of Computer Applications (0975 – 8887),121(3). https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.4377&rep=rep1&type=pdf

Roy, D., & Kundu, A. (2014). A Comparative Analysis of Three Different Types of Searching Algorithms in Data Structure. International Journal of Advanced Research in Computer and Communication Engineering, 3(5).

Saifuddin, A. B. (2018). CLINIC APPOINTMENT SCHEDULING SYSTEM (CASS). https://myfik.unisza.edu.my/www/fyp/fyp17sem2/report/041042.pdf

Zhao, P., Yoo, I., Lavoie, B. J., & Simoes, E. (2017, April 26). Web-based medical appointment systems: A systematic review. Journal of Medical Internet Research. https://www.jmir.org/2017/4/e134/