Clinic Check-in Queue System

In Partial Fulfillment of the requirements in
CS104P: Data Structures and Algorithms

Presented by:

Deus B. Cariño, BSCS 2nd
Joshua B. Cabahug, BSCS 2nd
Khing E. Laurente, BSCS 2nd

Presented to:
Prof. Martzel Baste

October 10, 2022

I. Introduction

A. Problem Scenario

Ever since the start of the pandemic, clinics have been constantly receiving more and more covid patients every day. There has been a surge of patients and it has been a challenge for clinics to keep up with the demand, especially with the lack of rooms and bed space. Because of that, the overpopulation of covid patients made it tough for nurses and staff to organize their queue system. The more patients coming in, the harder it is to accommodate all of them, making it a huge challenge. With the problem at hand, clinics have to compromise space just for covid patients, leaving the non-covid patients least prioritized. Papes (2021), a health journalist, states that many clinics in certain areas were already running out of space and personnel for non-COVID treatments such as fever symptoms which could possibly be a sign of COVID. That implies they have relatively few open beds to give to patients from tiny rural clinics lacking ICUs or medical centers located in viral hotspots.

Mismanagement of waiting lines in clinics adds to the stress of queuing. When you want medical assistance, the last thing you want is a huge line with no means of knowing where you stand (Lee, 2017). It has been a problem since before and it is an even bigger problem now that covid has been spread out throughout the entire world.

B. Objectives

To minimize patient trafficking in clinics by adding a better queue system. A more organized and efficient way of handling patient overpopulation is the goal. The proponents came up with a profound solution to the

a. **Scheduling.** It allows the patients to place a schedule for their check-up via their electronic devices. Once they have a schedule, they will be queued inside the system.
b. **Organizing.** The system will check the queues and organize the patients in order of who checked in earlier.
c. **Clearing.** When no more queues are detected, the system will proceed to the next day's queue

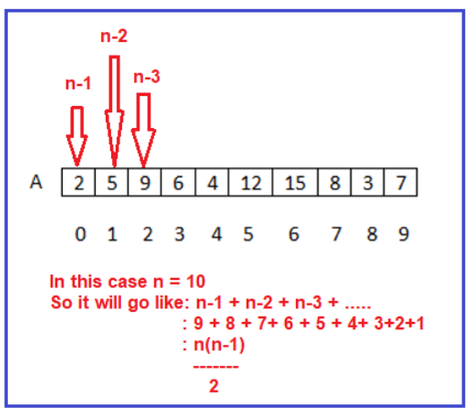
This method decreases the time it takes to organize the patient's check in time manually. Furthermore, this also decreases the workload for staffs and nurses making their job easier to do.

C. Project Definition

The Clinic Management Queue System uses the Queue linear data structure, with options such as enqueue, dequeue and update data. The first-in, first-out principle governs the Queue linear data structure. This strategy is also known as the first-come, first-served concept, because clients are served in the order in which they arrive. The program makes it so that data are added and withdrawn at the front and back ends of a queue linear data structure. Elements are added or enqueued at the front end, and they are deleted or dequeued at the back end. Each roll number is associated with personal information. The Clinic Check-in Queue is designed to be friendly to the user and easy to use.

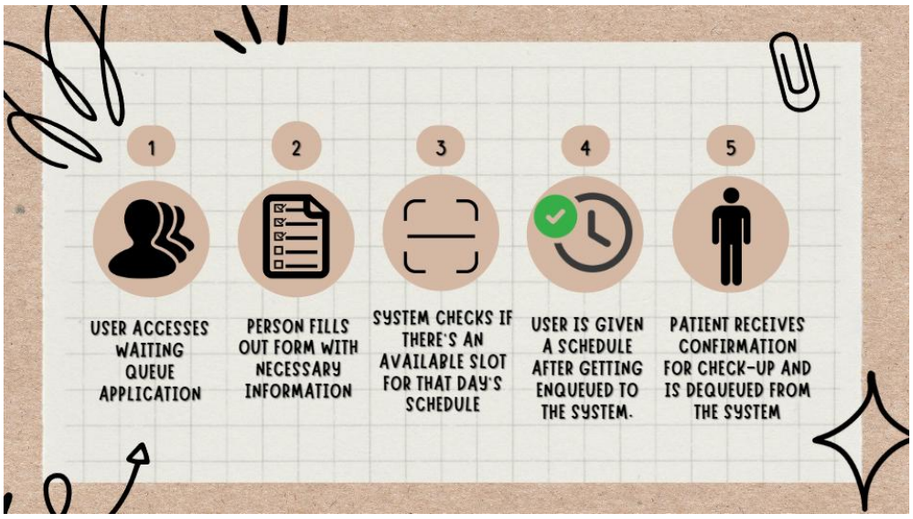
D. Time and Space Complexity

Time complexity is a measure of how long a function takes to run. In other words, it defines the amount of time an algorithm takes in terms of the algorithm's input on the other side, spatial complexity is a function related to the amount of space an algorithm requires in terms of the algorithm's input size The image seen here is of Queue. The space complexity and time complexity of the application. The space complexity of the Queue is O(n) since each input piece has a set number of k bytes



In a queue operation, the temporal complexity of enqueue and dequeue is 0 (1) since both actions only alter a few points (no looping) thing is taking place).

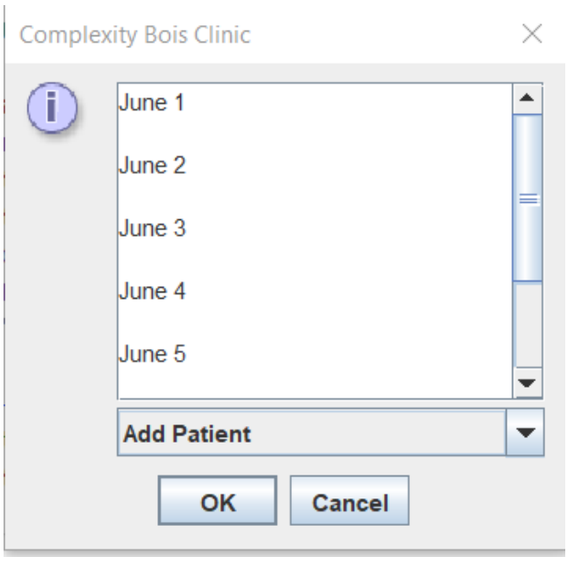
II. Project/System Prototyping



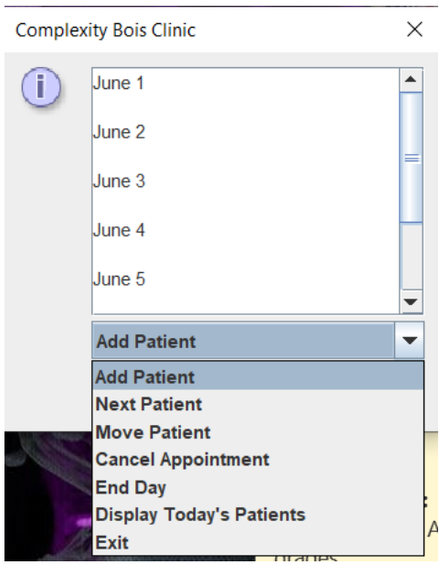
The figure above shows the flowchart for the Clinic Check-in System. When the application is used, the user will be asked when would they want to schedule an appointment. The user's input will then be added to the system, for the system to organize and then input the patient's desired schedule for the week. Once the patient has been successfully added to the queue, the system will then wait for more check-ins to schedule them.

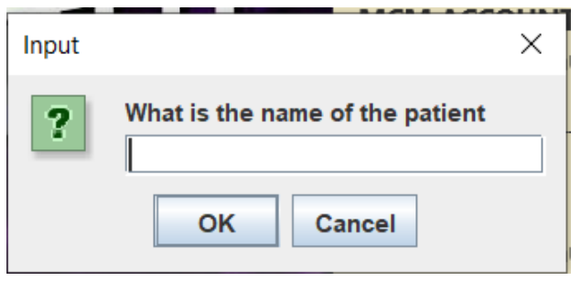
This process makes it easer for the system to take the user's input and convert it into data that the queue system needs.

III. Sample Input/Output

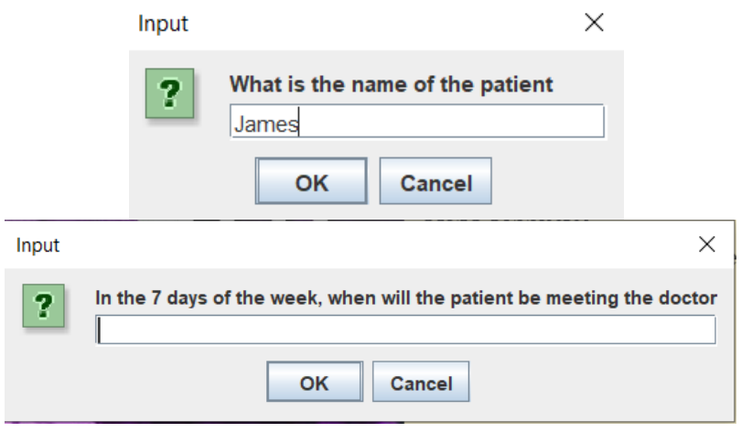


This is the main screen that appears when the user uses the program. A fairly basic and user-friendly interface for the patient. An option menu is visible leading to different options for the program to continue

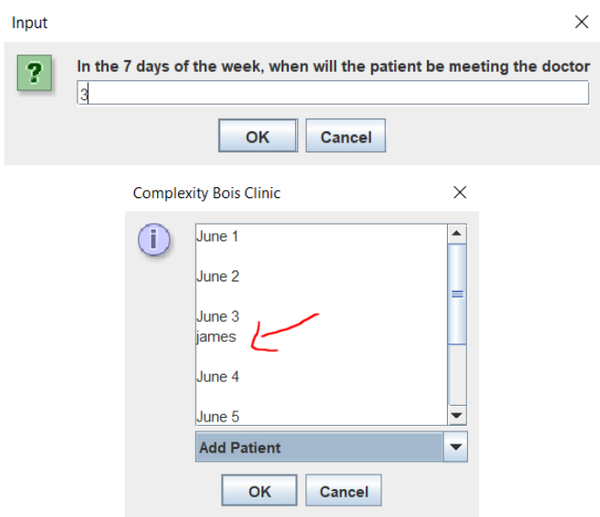




Once the options have been brought down by clicking on the down arrow, various options are able for the user to choose. For example, using the “Add Patient” option and clicking “ok” prompts the user to enter the name of the patient.



After inputting the name, the user is then prompted to input their desired day for check up in order for the system to add them to the queue.



The program uses a number system, you input the number aligned with the specific date you prefer for check-up.

IV. Source Code

A. Clinic Class

```
import javax.swing.*;


public class Clinic  {

   public static void main(String[]args){

      ClinicAppointment c=new ClinicAppointment();

      String[] menu={"Add Patient","Next Patient","Move Patient","Swap Patients","Cancel
Appointment","End Day",

            "Display Today's Patients","Exit"};

      String status,option;

      do{

         c.setMonth("June");

         c.merge();

         status= "Clinic opens at 9:00am to 7:00pm.\n10 patients per day only\n\n"+c.display();

         JTextArea textArea = new JTextArea (10,20);

         textArea.setText(status);

         textArea.setEditable(false);

         JScrollPane scroll = new JScrollPane (textArea);

         option= JOptionPane.showInputDialog(null,scroll,

               "Complexity Bois Clinic",1,null,menu,menu[0]).toString();
switch (option){

            case"Add Patient":

               c.addPatient();

               break;

            case"Next Patient":

               c.checkOutPatient();

               break;

            case "Move Patient":

               c.movePatient();

               break;
case"Swap Patients":

               c.swapPatientsInDay();

               break;

            case"Cancel Appointment":

               c.cancelPatient();

               break;
```

```
    case"Display Today's Patients":
                c.displayToday();
                break;
            case"End Day":
                c.endDay();
                break;
        }
     }while (!option.equals("Exit"));

  }
}
```

B. Clinic Appointment Class

```
import javax.swing.*;
import java.awt.datatransfer.StringSelection;
import java.util.*;



public class ClinicAppointment {
   private final int maxdays=7;
   private final int maxPatient=10;
   private int globalTimer=0;
   private String allList="";
   private String month;
   ArrayList<ArrayList<String>> clinic;

   public ClinicAppointment(){
      clinic = new ArrayList<>(maxdays);
      for(int i=0; i < maxdays; i++) {
         clinic.add(new ArrayList(maxPatient));
      }
   }

   private boolean isFull(int day){
      return clinic.get(day).size() == maxPatient;
}
   public void endDay(){
      globalTimer++;
   }
```

```
public void setMonth(String m){month=m;}


   private String dateToday(){
      return month+" "+globalTimer+1;
   }


   public void addPatient(){
      String name =JOptionPane.showInputDialog("What is the name of the patient");
      int day=Integer.parseInt(JOptionPane.showInputDialog("In the 7 days of the week, when will the " +
            "patient be meeting the doctor"));


      if (isFull(day-1)){
         JOptionPane.showMessageDialog(null,"The day is fully booked");
      }else {
         if (day < globalTimer + 1) {
            JOptionPane.showMessageDialog(null, "Cannot add patients to the days before today");
         } else {
            clinic.get(day - 1).add(name);
         }
      }
   }
public void movePatient(){
      String name =JOptionPane.showInputDialog("What is the name of the patient");
      int day=Integer.parseInt(JOptionPane.showInputDialog("In the 7 days of the week, when will the " +
            "patient be meeting the doctor"));
      if (isFull(day-1)){
         JOptionPane.showMessageDialog(null,"The day is fully booked");
      }else {
         if (day < globalTimer + 1) {
            JOptionPane.showMessageDialog(null, "Cannot add patients to the days before today");
         } else {
            cancelPatient(name);
            clinic.get(day - 1).add(name);
 }
      }
   }
public void cancelPatient(String name){
      name=name.toLowerCase();
      String placer;
```

```
for(int i=0;i<clinic.size();i++){
          for (int c=0;c<clinic.get(i).size();c++){
             placer=clinic.get(i).get(c).toLowerCase();
             if(placer.equals(name)){
                clinic.get(i).remove(c);
             }
          }
       }
    }
public void cancelPatient(){
       String name= JOptionPane.showInputDialog("What is the name of the Patient");
       name=name.toLowerCase();
       String placer;
       for(int i=0;i<clinic.size();i++){
          for (int c=0;c<clinic.get(i).size();c++){
             placer=clinic.get(i).get(c).toLowerCase();
             if(placer.equals(name)){
                clinic.get(i).remove(c);
             }
          }
       }
    }
    public void checkOutPatient(){
       clinic.get(globalTimer).remove(0);
    }


    public void swapPatientsInDay(){
       String first= JOptionPane.showInputDialog("What is the name of the Patient");
       first=first.toLowerCase();
       String second=JOptionPane.showInputDialog("What is the name of the Patient to be
swapped");
       second=second.toLowerCase();
       String placer;
       int indeX=0,indexY=0;
       for (int c=0;c<clinic.get(globalTimer).size();c++){
          placer=clinic.get(globalTimer).get(c).toLowerCase();
          if(placer.equals(first)){
             indeX=c;
          }
```

```
if(placer.equals(second)){
            indexY=c;
        }
    }
    Collections.swap(clinic.get(globalTimer),indeX,indexY);
  }


  public void merge(){
    int counter=1;
    String hold="";
    for(int i=0;i<clinic.size();i++){
      if(i<globalTimer){
        hold+=month+" "+counter+":Day Done\n";
      }else{
        hold+=month+" "+counter+"\n";
      }
      counter++;
      int patientCounter=1;
      int time=9;
      Object placer="";
      Queue q=new LinkedList();
      for (int g=0;g<clinic.get(i).size();g++) {
        q.add(clinic.get(i).get(g));
      }
      for (int c=0;c<clinic.get(i).size();c++){
        while (!q.isEmpty()){
          placer=q.remove();
          if(i<globalTimer){
            hold+= placer+"-did not come\n";
          }else{
            hold += "Priority "+patientCounter+" "+placer+" "+time+":00"+"\n";
          }
          patientCounter++;
          if(time==12){
            time=1;
          }
          time++;
        }
      }
```

```
hold+="\n";
    }
    allList=hold;
 }

 public String display(){return allList;}

 public void displayToday(){
    String hold;
    hold=dateToday()+"\n";
    for(int i=0;i<clinic.get(globalTimer).size();i++){
       hold+=clinic.get(globalTimer).get(i)+"\n";
    }
    JTextArea textArea = new JTextArea (10,20);
    textArea.setText(hold);
    textArea.setEditable(false);
    JScrollPane scrollPane = new JScrollPane(textArea);
    JOptionPane.showMessageDialog(null,scrollPane);
 }
```