Final Project Design Document
CS-273-1
Kailey Cozart

Requirements Specification

I will be building a computer simulation that simulates an emergency room for a town with a population of 2000 individuals. The program should allow users to see how efficient the emergency room can be. Moreover, users should be able to see the results of a simulation that occurs over a week on a minute-by-minute basis. The user will be allowed to input average hourly patient admittance rate, the number of doctors, and the number of nurses in the emergency room. In response, the computer will tell the user the average visit time of all the patients and will display the medical records of the treated patients when requested by the user.

Use Cases

Use Case for Entering Simulation Parameters

1. The user is prompted to enter the number of patients per hour in the simulation.
2. If the user enters a non-integer, or an integer less than 1 or greater than 60, the prompt reappears, allowing the user to enter a valid value.
3. The user is prompted to enter the number of doctors they want in the simulation.
4. If the user enters a non-integer, or an integer less than 1 or greater than 1000, the prompt reappears, allowing the user to enter a valid value.
5. The user is prompted to enter the number of nurses they want in the simulation.
6. If the user enters a non-integer, or an integer less than 1 or greater than 1000, the prompt reappears, allowing the user to enter a valid value.
7. The simulation is run.

Use Case for Each Clock Tick

1. The correct number of doctor and nurse objects have been created and added to the staff vector before the first clock tick.
2. At each clock tick, a statistical calculation based on the arrival rate determines if a patient arrives that minute.
3. If a new patient arrives that minute, a random number generator determines the severity of the patient's illness or injury.
4. If the doctor or nurse was working on a patient, the simulation checks if that staff member is done and removes that patient from them if they are done treating the patient.
5. If a patient is removed from a staff member because they are done being treated, their treatment time is added to the treatment time vector.
6. If a new patient arrived during that clock tick, a staff member is found for that patient.

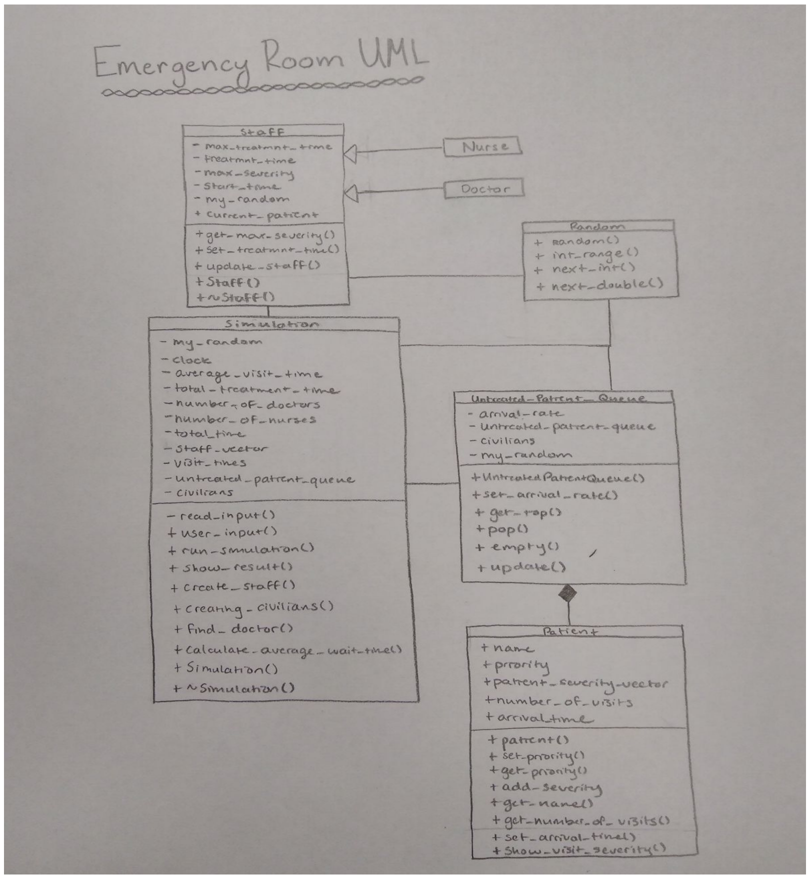
Use Case for Viewing Patients Treated at End of Simulation

1. When the simulation is complete, the average visit time is displayed and the user is asked if they would like to exit or see a list of treated patients.
2. If the user chooses to exit, the program terminates.
3. If the user chooses to see the list of patients, the list is displayed.

Use Case for Viewing Patient Records at End of Simulation

1. After the list of patients is displayed, the user is asked if they would like to exit or if they would like to see a patient record.
2. If they choose to exit, the program terminates.
3. If they choose to see a patient record, the user is prompted to enter the reference number next to the patient whose record they with to see.
4. Once that patient's record is displayed, the user is once again asked if they would like to exit or see another patient record.
5. This process continues until the user chooses to exit.

UML Diagrams



Pseudocode

Algorithm for read_input() in Simulation.h

```
1.  If the lower bound is greater than the higher bound
2.      Throw invalid argument exception
3.      Display the prompt again
4.      Allow the user to enter a new value
5.  If the user does not enter an integer value
6.      Thow ios base failure exception
7.      Display the prompt again
8.      Allow the user to enter a new value
9.  If the user enters a value that is not within the range set by the function parameters
10.     Display the prompt again
11.     Allow the user to enter a new value
```

Algorithm for create_staff() in Simulation.h

```
1.  For the number of doctors needed
2.      Push doctor objects into the staff vector
3.  For the number of nurses needed
4.      Push nurse objects into the staff vector
```

Algorithm for creating_civilians() in Simulation.h

```
1.  Open the file containing first names
2.  Open the file containing last names
3.  For 2000 civilians
4.      Take the nth line of the first name file
5.      Take the nth line of the last name file
6.      Combine those two lines into a string
7.      Assign the combined string to a name variable
8.      Create a patient object using the name variable
9.      Add the patient object to the map at index n
10. Close the file containing first names
11. Close the file containing last names
12. Return the map
```

Algorithm for find_doctor() in Simulation.h

```
1.  For the number of staff in the staff vector
2.      Remove any patients from the staff members that have finished treatment
3.      If a patient has finished treatment
```

```
4.          A total visit time is returned
5.          That total visit time is pushed to a vector
6.      Else if no patients finished treatment
7.          A value of zero is returned
8.          Zero is not pushed to the vector
9.      Check if a staff member has the capability to take care of the next patient
10.     If they can take the patient
11.         Remove the patient from the untreated patient queue
12.         Add a pointer from the staff member to the patient
13.         Set treatment time within staff to a randomized value
```

Algorithm for update_staff() in Staff.h, Doctor.h, and Nurse.h

```
1. If there is a patient and their treatment is complete
2.      Calculate the patient's total visit time
3.      Remove the staff member's pointer to that patient
4.      Make the staff's pointer null
5.      Reset start time
6.      Reset treatment time
7.      Return that patient's visit time
8. Else if there is no patient or the patient's treatment is not over
9.      Return 0
```

Algorithm for update() in Untreated_Patient_Queue.h

```
1. If a random double between 0 and 1 is less than the specified arrival rate
2.      Generate another random number between 0 and 1
3.      If the new random number is less than 0.7
4.          Generate a random number between 1 and 2000
5.          Choose a victim from the map of patients based on the random number
6.          Generate a random number between 1 and 10 for the patient's severity
7.          Set the arrival time to the current clock
8.          Push the victim to the untreated patient queue
9.      If the new random number is less than 0.9 and greater than .7
10.         Generate a random number between 1 and 2000
11.         Choose a victim from the map of patients based on the random number
12.         Generate a random number between 11 and 15 for the patient's severity
13.         Set the arrival time to the current clock
14.         Push the victim to the untreated patient queue
15.     Else
16.         Generate a random number between 1 and 2000
17.         Choose a victim from the map of patients based on the random number
18.         Generate a random number between 16 and 20 for the patient's severity
```

```
19.            Set the arrival time to the current clock
20.            Push the victim to the untreated patient queue
```