Group 4 – Spartan Healthcare

John, Sharanitha, Nick, Jackie, Germell

CSC3610 – Advanced Programming

December 03, 2019

Data Structure Documentation

* Array – We implemented the Array data structure to hold appointment times, as they are a set time schedule, and there are an exact number of appointment times every single day. This allows the use of the array structure since we know that the amount of appointments will never change, and the appointment times can be hard coded because they will never change.
* ArrayList – We implemented the ArrayList data structure to hold all of the objects involved in our program. These include users, patients, appointments, and payments. Since the amount of users, patients, appointments, and payments will all vary day by day; it made sense to read in the data into an ArrayList for each of these object types.
* Queue – We implemented the queue data structure by creating a “walk-in” feature for our program. The administrator can input a “walk-in” appointment, and it will be added to the queue that can be viewed by the administrator.
* Stack – We implemented the stack data structure using a “most recent” appointment indicator in our program. When the user views their list of appointments, the most recent appointment will be at the top of the list, as it was retrieved using the stack data structure.
* Hashmap – We implemented the Hashmap data structure by using it as a “search” function within our program. Each patient has a key assigned to them, and when you search through the patients as an administrator you are using the Hashmap data structure to find the patient you requested.
* LinkedList – We implemented the LinkedList data structure to help sort the list of appointments for each patient. When the list of appointments for that specific patient is loaded from the database it is implemented into a LinkedList before being presented to the user.
* Iterator – We implemented the iterator interface to help display reports, as it iterates over the appointment objects and displays the data without having to implement large amounts of code and retrieve the data piece by piece.
* Comparator – We implemented the comparator by using it to compare the user class to the patient class. This allows the program to know whether a patient has been created with a user ID that already exists from the user and patient database tables.
* Source Code Control – We used GitHub to maintain source code control during our project development. Anytime work was completed as a group, we ensured that everyone could upload and download the newest version of the project. Anytime work was completed individually, we ensured that each person checked each other’s work and confirmed that any updates would not break the current program status. Any time that the project was changed, which includes documentation regarding data structures, class information, and project roles, this was also uploaded to our source control. With the collaboration of both the project code files and the documentation, we were able to maintain high levels of source code control.