xSchedule Class Diagram

This document provides more information about the classes that can be seen in the xSchedule Class Diagram. Included below are the fields and methods for each class along with pseudocode for the functions. Currently the class diagram shows a separation between the User objects and the queueSystem object which we believe is the simplest design. As more functionality is added, there may be changes to the functions or where they are located for efficiency.

User Interface:
Fields:
+ userType: string
+ account_id: int
Methods:
+ getUserType():
return userType
+getID():
return account_id

Customer implements User:
Fields:
+ prev_jobs: Int []
+ open_jobs: Int []

Methods:
+ getPriority(): int

```
len = length(prev_jobs);
if(len >= 5)
    return 0
else if(len < 5 && len >=3)
    return 1
else if(len == 1)
    return 2
else
    return 3
```

+ getBill(int id): int

```
job j = getJob(Id)
```

return j.total_bill;

Technician implements User:
Fields:

+ assignment: Int
+ payrate: Int

Methods:

+ getJobAssignment(): void

```
if assignment == -1
    id = getID() ;
    assignment = firstInQueue(id);
return
```

+ enterTimeArrival(): void

```
Job j = getJob(assignment);
j.time_arrival = Datetime.Now();
setJob(j) ;
```

+ enterTimeComplete: void

```
Job j = getJob(assignment);
j.time_complete = Datetime.Now();
timeworked = j.time_complete - j.time_arrival;
j.total_bill = payrate * timeworked;
setJob(j) ;
assignment = - 1;
```

Manager extends Technician:
Methods:

+ modifyJobRequest(int Id)

```
Job j = get Job(id)
[make changes ]
setJob(j)
return;
```

Job Object:
Fields:

+ job_id: int
+ customer_id: int
+ complexity: int
+ priority: int
+ time_submit: time

+ technician_id: int
+ time_arrival: time
+ time_complete: time
+ bill_total: int

---

queueSystem Object:
Fields:

+ queue: Int []
+ ongoing: Int[]
+ alljobs: List of <Job>
+ jobcounter: int = - 1

Methods:

+ getJob(int id) : Job

```
if (id <= jobcounter)
    return alljobs[id]
else return Null
```

+ setJob(Job j): void

```
int i = j.job_id;
alljobs[i] = j;
return j;
```

+ createJob(Customer c, Int difficulty): void

```
Job j = new Job();
jobcounter = jobcounter + 1;
j.job_id = jobcounter;
j.customer_id = c.getID();
```

```
        j.priority = c.getPriority();
        j.complexity = difficulty;
        j.time_submit = DateTime.Now();
        alljobs.add(j);
        [add job to "open" in order of priority]
        return;

    +   firstInQueue(int id) : int
                if len(open) > 0
                        int id = open[0];
                        alljobs[id].technician_id = id;
                        open = open.remove(0);
                        ongoing = ongoing.add(Id) ;
                        return id;
                else
                        return -1

+ getAvgWait(): int
+ getAvgLength(): int
+ getPrecentageEmpty(): int
+ getIncompleteJobs(): int
+ getTechicianHours(): List of <Technician, idleHours(int)>

+ generateDailyReport
+ generateMonthlyReport
```