**Team 2 Washing Machine Web Application Design Doc - backend**

Introduction:

The administrators of the community laundry room have recently received lots of complaints from the residents. The residents often encounter scenarios such that the clothes are not being collected in a timely manner, or sometimes, they went there but all of the washers were occupied. In order to resolve this problem, the administrators want to design a web page to help the user to use the laundry room more effectively.

This design doc focuses on the MVP product. Further development phase should be modified and implemented based on project execution progress.

Scenarios(MVP phase)

1. Resident can check the availability of washing machine before heading laundry room. Washing Machine WebApp shows washing machine current availability and possible available time window, so residents can plan their laundry accordingly.
2. Residents can plan for pick up the completed laundry. WebApp shows the remaining time of washing task associated with user ID, so resident can plan for pick up based on the remaining time.
3. Residents will receive notification when laundry is completed. When laundry is completed, WebApp can notify user in several methods, Alart in WebApp, email notice,

text message.

Overview of Design

Tech Stacks:

* Presentation Layer (front end): HTML, CSS, JS, REACT(front end group will decide based on actual need)
* Logic Layer: Java/Apache Tomcat
* Data Base: MySQL
* AWS
* Git for sync codes (refer to LaiProject Flag Camp Section for requesting Git instruction video)

1. Use case

|  |  |  |
| --- | --- | --- |
| Produce Development | | |
| Phase 1 (MVP) | Phase 2 | Phase 3 |
| User login, logout, register | supervisor account (add/delete washing machine quantity/report issue) | implement variety of laundry modes( load, time) |
| Check machine availability | reservation | implement pricing options based on laundry mode |
| start a job & check current job status | punishment for improper usage (fine, blocking usage and etc) |  |
| Notification of job complete for pick up | payment management (auto pay, balance check) |  |

Listed above are use cases that our program should realize, in terms of development phase. MVP serves as the base requirement for our program development.

2. Class in Java and Table in MySQL

|  |  |
| --- | --- |
| Class in Java | Field |
| User | Password, username, possible other |
| Washing Machine | Washing machine ID, availability, associated user, start and end time |
| Logic Handling | any necessary fields during implementation |

|  |  |
| --- | --- |
| Tables In SQL | Schema |
| User | Password, username |
| Washing Machine | Washing machine ID, availability, associated user, start and end time |
| Washing Task | Machine ID, user, washing start and end time |

3. Public Interface



|  |  |
| --- | --- |
| Front End | RPC Handler |
| /login | login Handler |
| /logout | logout Handler |
| registration | registration Handler |
| /checkAvil | checkAvailable |
| /checkWashing | checkTask (task completion and remaining time) |
| /startWashing | initiate task |

|  |  |
| --- | --- |
| API User | MySQL Client API |
| RPC | login |
| logout |
| register |
| Logic Handler (handle washing task) | updateUser |
| updateWasher(update availability and associated user) |

|  |  |
| --- | --- |
| API User | Logic Handler API |
| RPC | checkAvailability |
| startWashing |
| checkWashingRemainingTime |
| notifyWashingEnd |

4. Implementation Details

* Discuss Session or Token base.
* Check availability can be implemented by polling machine status at certain time interval.
* Check availability, start job and check current job status can be implemented by a priority queue that handles current washing task, and a collection that lists available machines. Details refer to following session.

Define each washer as a singleton class, which means each washer has only one instance

e.g.

|  |
| --- |
| class Washer  {  private static Washer washer = null;  private boolean status;  private Washer(String id, ...)  {  ...  }  public static Washer getInstance(String id)  {  if (washer == null)  washer = new Washer(String id);    return washer;  }  } |

e.g.

|  |
| --- |
| PriorityQueue<Washer> pq = new PriorityQueue<>(# of washers, (Washer washer1, Washer washer2) -> {  return washer1.getEndTime() - washer2.getEndTime();  }); |

Whenever a user trying to do the laundry on a specific washer

1. the ‘status’ field of that washer will be marked as false
2. the user infos will be binded to the specific washer temporarily(from cur time to the end of the task)
3. update the start time and end time of the washer
   1. the starting time will be the current system time (e.g. LocalDateTime.now())
   2. the end time will be start\_time + time\_needed\_to\_do\_the\_laundry
4. the washer will be offered to the priorityqueue to simulate the laundry task

Main Logic:



The waiting room info (available washers) will be passed to the front end, so that user can choose whatever they want among them

The remaining time can be calculated using following logic:

user login → go to user table → find that user → if washer in user == true → find washer id → go to washer table → get end time

|  |
| --- |
| Remaining time = end\_time - current\_time |

If the task is done, and the washer has been dequeued from the priorityqueue, the user will get the notification, we allow a five minutes window for the status of the washer to be set as true. (user can pick up their clothes during that time)



If currently, all washers are occupied, namely, the waiting room is empty, then the user has to wait until the first (top in pq) washer finish its job. We will let the user know how long the user has to wait until he/she can start doing laundry.

|  |
| --- |
| Time to wait = (pq.peek().getEndTime() + 5 min ) - current\_time |

SQL:

1.When a user trying to use a washer to do the laundry task, the washer table will be updated in the mysql database.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Washer id | status(availability) | user id | start time ??  (if needed) | end time ??  (if needed) |
| #00001 | true → false | #abcdefg | NULL → 12:00 | NULL → 12:30 |

At the same time, update the user table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User id | First Name | Last Name | ~~balance~~  ~~(phase 2)~~ | washer in use | Washer id |
| #abcdefg | John | Smith | ~~50 → 30~~ | false → true | NULL → #00001 |

Whenever the laundry job is done (e.g. the washer dequeued from the priorityqueue, update the user table and washer table again)

User table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Washer id | status | user id | start time ??  (if needed) | end time ??  (if needed) |
| #00001 | false → true | #abcdefg | 12:00 → NULL | 12:30 → NULL |

Washer table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User id | First Name | Last Name | ~~balance~~  ~~(phase 2)~~ | washer in use | Washer id |
| #abcdefg | John | Smith | ~~30~~ | true → false | NULL |

5. Timeline

|  |  |  |
| --- | --- | --- |
| **Timeline** | **Milestone** | **To do** |
| 4/19 | Project starts |  |
| 4/23 | decided project name-Laundry management |  |
| 4/27 | MVP discussion |  |
| 4/30 | Design doc - MVP and technique |  |
| 5/4 | API definition, DB schema, assign task |  |
| 5/7 | Code Implementation |  |
| 5/11 |  |
| 5/14 | Integration |  |
| 5/18 | Integration and implementation Deadline |  |
| 5/24 | Test and deployment | prepare review document |
| 5/25 | Present Demo in Class |  |

6. Team Assignment

Regular Meet up:

|  |
| --- |
| **Team Meeting time** |
| Tuesday: 9:00pm eastern time/6:00pm pacific time |
| Saturday: 4:00pm esstern time/1:00pm pacific time |
| Group meeting through Zoom or Wechat voice |

Team Info:

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | **邮箱地址** | **Github账号** | **Assignment** |
| Shuya Zhang | shuyazhang2017@gmail.com | sheerryZ |  |
| Xiao Tang | xtang017@gmail.com | xtang7 |  |
| Di Peng | dpeng@hawk.iit.edu | <https://github.com/doradebop> |  |
| Xueting Bao | xuetingbao99@gmail.com | XuetingBao |  |
| zongliang zhang | willzhang1019@gmail.com | Willzhang1019 |  |
| Yu Jin | sysujinyu@gmail.com | yujin19 |  |
| Haoyu Li | hli73@ur.rochester.edu | <https://github.com/triml> |  |
| Hongxia Tang | thxella@gmail.com | EllaTang1 |  |
| Lu Yin | lu.yin2046@gmail.com | Cass2046 |  |

7. Poll for team assignment (back end)

|  |  |  |
| --- | --- | --- |
| **姓名** | **Git Responsibility** | **Work on which APIs** |
| Shuya Zhang |  | FrontEnd |
| Xiao Tang |  | RPC |
| Di Peng |  | MySQL |
| Xueting Bao |  | Logic Handler |
| zongliang zhang |  | MySQL |
| Yu Jin | GIT (frontend) | FrontEnd |
| Haoyu Li | GIT(backend) | Logic Handler |
| Hongxia Tang |  | RPC |
| Lu Yin | GIT(backend) | FrontEnd |

1. Deadline for coding : 5/18
2. Check with other members to see if you can test your logic. Don’t wait until the deadline to test.
3. Yu Jin is in charge of frontend Git.
4. LuYin and Haoyu are in charge of backend Git. Check with Yu Jin for any question.
5. A readMe.md or .txt in each Git.

* Back End readMe.md includes:

Java: Package, Class, API signature, Description, coder

MySQL schema: table name, primary keys, foreign keys, unique keys

1. Update front end scratch in this Design doc whenever front end team feel needed.
2. Before Git is ready, write readme in Design Doc.

Readme (frontend)

Readme(backend)