# Quality Plan

* [Quality Assurance](#30j0zll)
  + [Quality assurance list](#1fob9te)
  + [Quality Assurance Tools](#3znysh7)
    - [Findbugs](#2et92p0)
* [Quality Control Activities](#tyjcwt)
  + [Risk register](#3dy6vkm)
  + [Requirements Verification Matrix](#1t3h5sf)
  + [Testing](#4d34og8)
    - [Test Level Decisions](#2s8eyo1)
      * [Unit Test](#17dp8vu)
      * [Integration Test](#3rdcrjn)
      * [System Test](#26in1rg)
    - [Functional Testing Technique](#lnxbz9)
    - [Structural Testing Technique](#35nkun2)
  + [Training](#1ksv4uv)
  + [Testing Schedule](#44sinio)
  + [TODO](#2jxsxqh)
  + [Quality Assurance Log](#z337ya)
  + [Quality Control Log](#3j2qqm3)

# **Quality Assurance**

## **Quality assurance list**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Subject** | **Quality Attribute Requirement (Work)** | **Stakeholders** | **Function**  **Categories** | **Entry criteria** | **Exit criteria** | **Related standards? Regulations? Purpose?** | **Additional Resource** | **Third party dependency** | **Related** |
| QA01 | Parking Extendibility | The design of the system should allow developers to easily accommodate new parking spots and parking configurations | System | Configuration interface | need a clear UI for the interface Developers want | UI/UX specification | Accessibility | No | No |  |
| QA02 | Parking Extendibility | New parking spots are **recognized by the system within 10** minutes of system startup. | System | Booting | Related Product research | Found WIFI hardware recognizing parking spots satisfied within 10 minutes of system startup. | Usability | No | Yes |  |
| QA03 | Consistency | The system must maintain an **accurate view of garage state** during normal operation.  The Tartan Garage system should be out of sync with the g**arage hardware by no more than 10 seconds.** | System,  Garage hardware | Synchronization | Related Product research | Found WIFI hardware recognizing spots within 10 seconds. | Usability | No | Yes |  |
| QA04 | Parking Security | The system should **not allow unauthorized entry** to the parking garage during normal operation.  The system will prevent unauthorized entry 100% of the time. | System | Security | - | check secure communication protocol | Security | No | No |  |
| QA05 | Payment Security | The system shall prevent customer payments from being compromised when transmitted over a network.  The system will prevent unauthorized access of payment information 100% of the time. | Payment | Security | - | check secure communication protocol decision | Security | No | No |  |
| QA06 | Visibility | The system shall allow administrators to **easily collect data and statistics on the parking system.** | System | Logging interface | - | need to identify the interface or method how admin wants to access them | Usability | No | No |  |
| QA07 | Invariant System Properties | Reservations can only be redeemed once. | Customer | Reservation / | - | - | Reservation policy | No | No |  |
| QA08 | Invariant System Properties | A reservation must be unique within the system by customer name, vehicle ID (i.e. license, plate), or time. | Customer | Reservation | - | Determine UI to get the information | Reservation policy | No | No |  |
| QA09 | Invariant System Properties | Reservations must always be paid for before entry to the garage is granted. | Customer | Operation | - | - | Operation policy | No | No |  |
| QA10 | Invariant System Properties | Reservation fees and penalties must be accurate. | Customer | Reservation | - | - | Operation policy | No | No |  |
| QA11 | Invariant System Properties | Reservations for a parking spot must not overlap. The system should never reserve the same | Customer | Reservation | - | - | Operation policy | No | No | QA08  QA12 |
| QA12 | Invariant System Properties | Reservations for a parking spot must not overlap.  The system should never reserve the same parking spot for more than one car for the same time period.  That is, each parking spot must have at most one reservation per time period. | Reservation | Reservation | - | - | Operation policy | No | No | QA08  QA11 |

## **Quality Assurance Tools**

### Findbugs

It is important to find defects in the code without executing the code. We can detect faults in advance through static analysis. We can also find defects that are difficult to detect with dynamic testing. I think static analysis is important because we can also find dependencies and inconsistencies in the software model.

==Static Analysis result==  
TartanGarage.jar:215 Suspicious comparison of Integer references in edu.cmu.tartan.service.ParkingService.handleGarageEntry(Reservation) [Scariest(1), High confidence]  
TartanGarage.jar:303 Possible null pointer dereference of selectedRsvp in edu.cmu.tartan.service.KioskService.handleRedeemReservation(HashMap) [Scary(8), Normal confidence]  
TartanGarage.jar:397 Possible null pointer dereference of results in edu.cmu.tartan.service.ReservationService.handleRedeemReservation(HashMap) [Scary(8), Normal confidence]  
TartanGarage.jar:66 Incorrect lazy initialization and update of static field edu.cmu.tartan.hardware.TartanGarageConnection.connection in edu.cmu.tartan.hardware.TartanGarageConnection.getConnection(String) [Scary(8), Normal confidence]

# **Quality Control Activities**

## **Risk register**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Risk ID** | **Condition** | **Consequence** | **P** | **I** | **E** | **Mitigation Activities** |
| R-01 | Hardware damaged during development | Damaged hardware causes inaccurate results |  |  |  | Dedicate safety engineer to monitor and repair hardware. |
| R-02 | Sync with the garage hardware by 10 seconds | Frequency interference in an external environment |  |  |  |  |

## **Requirements Verification Matrix**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Requirement** | **Requirements Ref.** | **Quality characteristic** | **VerificationTechnique** | **When/ TestingLevel** | **Verification ProcedureRef.** | **Status** |
| UC01 ~ UC15 | SRS | Functionality | Testing, Demonstration | Integration Test |  |  |
| The design of the system should allow developers to easily accommodate new parking spots and parking configurations. New parking spots are recognized by the system within 10 minutes of system startup. | SRS | Extendibility /  Performance | Testing | System Test |  |  |
| The system must maintain an accurate view of garage state during normal operation. The Tartan Garage system should be out of sync with the garage hardware by no more than 10 seconds. | SRS | Consistency | Testing | System Test |  |  |
| The system should not allow unauthorized entry to the parking garage during normal operation. The system will prevent unauthorized entry 100% of the time. | SRS | Security | Testing | Integration Test |  |  |
| The system shall prevent customer payments from being compromised when transmitted over a network. The system will prevent unauthorized access of payment information 100% of the time. | SRS | Security | Source Code Checking  (implemented security layer) | - |  |  |
| The system shall allow administrators to easily collect data and statistics on the parking system. | SRS | Functionality | Demonstration | System Test |  |  |
| Reservations can only be redeemed once | SRS | Constraints | Testing | Unit Test |  |  |
| A reservation must be unique within the system by customer name, vehicle ID (i.e. license plate), or time. | SRS | Constraints | Testing | Unit Test |  |  |
| Reservations must always be paid for before entry to the garage is granted. | SRS | Constraints | Testing | Unit Test |  |  |
| Reservation fees and penalties must be accurate. | SRS | Constraints | Testing | Unit Test |  |  |
| Reservations for a parking spot must not overlap. The system should never reserve the same parking spot for more than one car for the same time period. That is, each parking spot must have at most one reservation per time period. | SRS | Constraints | Testing | Unit Test |  |  |

## **Testing**

### **Test Level Decisions**

#### **Unit Test**

* Services of the Parking Application
  + ParkingService
  + KioskService
  + PaymentService
  + ReservationService

Testing Tool

* **JUnit / Mockito**
  + Parking Application is implemented in Java. Therefore, it is possible to verify the method unit function by using JUnit. The unit tests can be easily done using Mockito.
* **JaCoCo**
  + Identify code / branch coverage

Using Test double : **Fake, Stub**

* Verification of the parking application is required without dependency on the hardware. Therefore, it is necessary to replace the Stub Hardware (Arduino).
* For test it is not always possible to access the actual Database. Therefore, a fake database that can replace the database is needed.

#### Integration Test

* Parking Application - Tartan Garage Software
* Parking Application - Reservation Database

Using Test double : **Fake**

* To test the parking application and the target garage software, we need to use the database as a fake.   
  It is not necessary to make a real connection to the database every time because it is the purpose of integration test like each system without dependency on DB.

#### **System Test**

* Parking Application - Database - Target Garage Software - Hardware

### Functional Testing Technique

* Boundary Testing
  + Basic Testing for every function
* Robustness Testing
* need to be robustness about anything that is done by user.

### **Structural Testing Technique**

* MCDC
  + Most reasonable testing technique without commercial tools.

## **Training**

|  |  |
| --- | --- |
| **Title** | **purpose** |
| how to use the JUnit4, Mockito | Unit test library In JAVA |
| the overview connectivity hardware feature  (WIFI, parking lot sensors, IR Detector, ) | To get a meaningful test case |

## **Testing Schedule**

We assume we have a year of time.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Schedule** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **Details** |
| Analysis | 2m | |  |  |  |  |  |  |  |  |  |  | Review quality levels, specification and, quality assurance in the existing tartan requirements. |
| Design |  |  | 3m | | |  |  |  |  |  |  |  | Design test environment, test suit, and staffing, architecture |
| Test Environment |  |  |  | 3m | | |  |  |  |  |  |  | Set up test environment . |
| Test execution |  |  |  |  |  |  | 3m | | | 2m | | 1m | Verification period, We do 3 cycles. |
| Completion report |  |  |  |  |  |  |  |  |  |  |  | 1m |  |

## *TODO*

## *Quality Assurance Log*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Trial # | Date | Process Measured | Required Value | Actual Measured | Acceptable? (Y/N) | Recommendation | Date Resolved |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

## *Quality Control Log*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Cable # | Date | Item Measured | Required Value | Actual Measured | Acceptable? (Y/N) | Recommendation | Date Resolved |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |