# 3. Evaluation - Alex

[The following text describes the tests performed on each subsystem of The Quicket System to ensure functionality. Each subsystem was tested to the standards outlined by the design constraints in Table 3.

**Table 3 Technical Design Constraints**

|  |  |
| --- | --- |
| **Name** | **Description** |
| Read Count | The system identifies up to two tags at once. |
| Range | The reader operates within a 2-meter range of tags. |
| Response Time | The system signals errors within 2 seconds of attempting to read a tag. |
| Security | The system determines the number of individuals approaching the reader, up to four people. |
| Error Signaling | The system alerts both the tag holders and nearby personnel of any errors by lighting a minimum of 20 red LEDs for 4 - 5 seconds. |

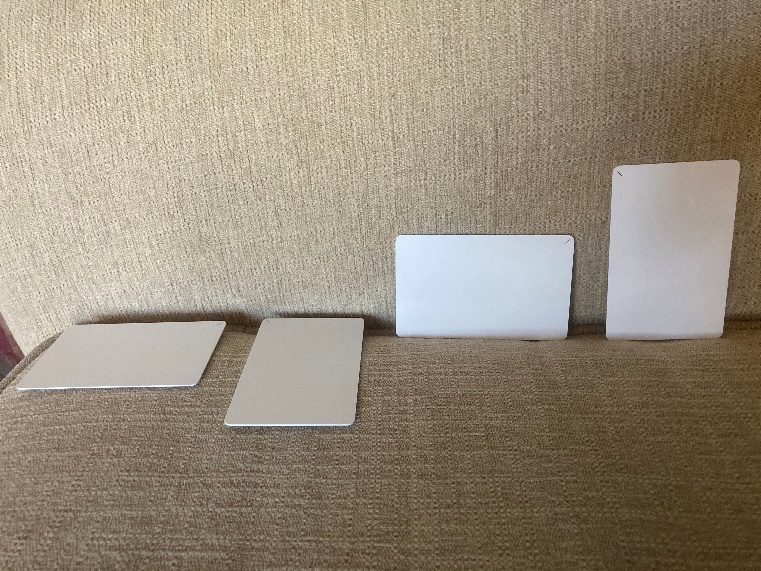
The following sections describe what tests were performed in order to ensure each subsystem met the established requirements.]

## 3.1 Test Certification – RFID Ticketing -- Drew

[The Quicket System relies on UHF RFID technology to process tags within the required ticketing range and needs to accurately read multiple tags up to a maximum range of 2 meters as defined in the technical design constraints. It is imperative to confirm these capabilities through a variety of tests involving both the selected reader and corresponding UHF RFID tags. The major tests involve determining tag readability through various tag orientations, locations, and numbers within the read area.

**3.1.1 Test Certification – Tag Orientation**

The first test focuses on determining the effect that different tag orientations have on the reader’s ability to process tags within the ticketing area. The tags were tested in four different orientation patterns, shown in the figure below, at reader output power levels of 15 dBm and 27 dBm. These two power levels give results from the middle of the reader’s power capabilities and at its maximum. Each tag was placed in direct line of the antenna at 1 meter away. If the tag in its orientation was read successfully, the tag was slowly moved away from the reader until it ceased to be read.



**Figure 3.1: 4 Tag Orientations for Testing**

This maximum read distances were recorded below in Table 3.1 and Table 3.2. Knowing the optimal tag orientation aids in ensuring the required read distance is met.

**Table 3.1: Tag Orientation Test with Output Power of 15 dBm**

|  |  |  |
| --- | --- | --- |
| **Tag Orientation** | **Read at 1 m** | **Max Range (m)** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |

**Table 3.2: Tag Orientation Test with Output Power of 27 dBm**

|  |  |  |
| --- | --- | --- |
| **Tag Orientation** | **Read at 1 m** | **Max Range (m)** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |

The team will discuss these results once the tests are performed and the data is recorded.

**3.1.2 Test Certification – Effective Coverage Area**

The second test for the RFID reader subsystem determines if the reader adequately covers the required ticketing area in compliance with ADA standards. ADA standards mandate that the reader be located 84 inches above ground level and that any walkways/corridors below have a minimum width of 5 feet. Therefore, the reader needs to cover a width of 5 feet at the 2-meter required read range design constraint. The test began with the tag 2 meters away and in direct line of the reader’s antenna. The tag was then slowly moved to the left and right until it ceased to be properly read. This offset distance was recorded in Table 3.3, and the same test was repeated for increments of 0.3 meters back to the reader to determine an overall effective field for the reader and antenna.

**Table 3.3: Effective Ticketing Area**

|  |  |  |
| --- | --- | --- |
| **Distance (m)** | **Max Left-Center Offset (m)** | **Max Right-Center Offset (m)** |
| 2 |  |  |
| 1.7 |  |  |
| 1.4 |  |  |
| 1.1 |  |  |
| 0.8 |  |  |
| 0.5 |  |  |
| 0.2 |  |  |

The team plans to further discuss the results once the test is completed and the data available.

**3.1.3 Test Certification – Multiple Ticket Reading**

The final test confirms that the reader properly detects the correct number of tickets simultaneously in the reading area. The maximum number of expected tickets in the read range is two; therefore, using the data from the previous tests, two tickets were placed in varying locations in the required area of the reader. The scenarios included two tickets near the reader, one close and one far, and both at the edge of the required range. The team repeated the overall test twice and recorded the data in Table 3.4.

**Table 3.4: Multiple Ticket Reading**

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario** | **Test Number** | **Total Tags Read** | **Pass/Fail** |
| Both Close | 1 |  |  |
| Both Close | 2 |  |  |
| One Close, One Far | 1 |  |  |
| One Close, One Far | 2 |  |  |
| Both Far | 1 |  |  |
| Both Far | 2 |  |  |

## The team plans to further discuss the test results once data is available.]

## 3.2 Test Certification – Raspberry Pi Event Timing– Preston

[To ensure efficiency of The Quicket System, the Raspberry Pi 4B needs to complete the necessary steps in the ticket validation process within the design constraint of a total of 2 seconds, maximum.

To test the timing of the different cases that make up this process, both a library to control the GPIO and a database to hold the tag information were needed. The database used for this simulation was stored locally on the Raspberry Pi and consisted of a 3-field, single-table SQL database. The represented fields include the RFID tag number, a “used” condition bit, and the name of the tagholder. To control the GPIO pins for the LED strip, an Adafruit circuitry module was used along with a rpi\_ws281x module to control the LED color.

To simulate the reading of a tag, a Python 3 script was started with user input representing the read event. After the user entered a tag number, the local database was queried. If the tag was in the database, the Raspberry Pi would receive the row of all the data row(s) with the tag number entered. Upon receipt of this row(s) of data, the Raspberry Pi 4B sent an additional query to the database to determine if the tag read had also been used previously. Lastly, if the tag had not been used, a final instruction was sent to update the tag as used. If error was determined, the red LEDs were turned on using the Python LED addressing libraries. Boolean variables were set when an error was detected to handle these output conditions.

Three tests were run for three different cases. The three cases consisted of situations where no error occurred (regular operation), a used tag was scanned, and an invalid database return was received. For a single test, the script ran once from start to finish. Once each event was completed, a timestamp was taken of the elapsed time since the tag was read.

Because the times for each case were calculated using simulated input conditions and a locally stored SQL database, slight increases may occur when factoring in remote server and hardware communication delays with full-scale prototype testing.

**3.2.1 Test Certification – Regular No-Error Operation**

Under no-error conditions, the LED strip remained green from the beginning to the end of single test. For these tests, tickets in the SQL database were all assigned a 0 value for the “used” field, and then three different tickets were entered on three separate runs of the python script. The elapsed time splits are given in Table 3.5.

**Table 3.5: Regular No-error Operation Timing**

|  |  |
| --- | --- |
| **Event** | **Average Elapsed Time (ms)** |
| Ticket Read | 7.78e-03 |
| Ticket Found | 2.52 |
| Used Query | 4.10 |
| Updated Database | 14.4 |

As seen in Table 3.5, the average total elapsed time of 0.014 seconds never exceeded the design constraint of 2 seconds. The longest individual test time for regular operating conditions was 0.01502 seconds. Under the simulated conditions, the results indicate that the Raspberry Pi handled all necessary database queries and hardware output signals well within the design constraints. Under regular operating conditions, the Raspberry Pi needed to execute all events.

**3.2.2 Test Certification – Valid Used Tag Re-scanned**

In the second case, a ticket that exists in the database but has already been used could be scanned. In this scenario, the same database was used but modified so that a couple tickets had the “used” field set to 1 prior to testing. Once the database was queried about the status of the tag’s used bit, the GPIO pin was set high using the LED addressing libraries to turn the LED strip red and indicate an error. The resulting elapsed time splits are shown in Table 3.6.

**Table 3.6: Re-used Tag Scan Timing**

|  |  |
| --- | --- |
| **Event** | **Average Elapsed Time (ms)** |
| Ticket Read | 8.02e-03 |
| Ticket Found | 2.51 |
| Used Query | 4.05 |
| Error Signal Sent | 8.77 |

When a re-used tag was scanned, the average total time until the red LEDs were signaled was 0.00877 seconds as shown in Table 3.6. The longest single test ran for 0.00998 seconds. The total elapsed time measurements were all well under the required 2 seconds for error detection.

**3.2.3 Test Certification – Improper/Missing Database Return**

The final condition tested included multiple error cases. As mentioned in the procedure above, when a query was sent to the database, information returned could include multiple rows or no rows of data from the table. If the tag was not found in the database or found multiple times, a Boolean was set to turn the LED strip to red using the LED addressing libraries.

**Table 3.7: Invalid Database Return Timing**

|  |  |
| --- | --- |
| **Event** | **Average Elapsed Time (ms)** |
| Ticket Read | 7.87e-03 |
| Error Signal Sent | 6.69 |

When a re-used tag was scanned, the average total time until the red LEDs were signaled was 0.00877 seconds. The longest single test ran for 0.00674 seconds. The total elapsed time measurements were all well under the required 2 seconds for error detection.]

## 3.3 Test Certification - Entry Counting -- Garrett

[The most important part of the entry counting subsystem is its ability to accurately count the number of people who are within the camera’s view. If the number of people attempting entry is more than the number of tickets detected, the entry counting subsystem notifies other subsystems of such.

**3.3.1 Test Certification – People Counting**

Tests for the people counting functionality consisted of having a set number of people walk into the camera’s view. A test case passes if the entry counting subsystem’s output accurately reflects the number of people present. This test was performed five times and the results are shown in Table 3.8.

**Table 3.8: People Counting**

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of people in camera’s view** | **Test number** | **Number of people counted** | **Pass/fail** |
| 0 | 1 | 0 | PASS |
| 1 | 2 | Various numbers between 0 and 2 | FAIL |
| 2 | 3 | Various numbers between 0 and 2 | FAIL |
| 3 | 4 | Various numbers between 0 and 4 | FAIL |
| 4 | 5 | Untested | Untested |

While there were several instances where the number of people counted by the subsystem was accurate, the team did not feel that these instances were consistent enough to consider any but the first test case passed.

**3.3.2 Test Certification – Position Tracking**

The entry counting subsystem must accurately count the number of users visible and their position relative to the ticket reading area. The accuracy of the location tracking is important so that ticket holders entering are not considered unauthorized entries and vice versa.

Position tracking tests were performed by having two groups of people stand in two different positions within the camera’s view. The test case passes if the numbers of people in each position were accurately reflected by the entry counting subsystem’s output. This test was performed four times and the results are shown in Table 3.9.

**Table 3.9: Position Tracking**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Number of people within read range** | **Number of people outside of read range** | **Test number** | **Number of people reported to be within read range** | **Number of people reported to be outside of read range** | **Pass/fail** |
| 0 | 1 | 1 |  |  |  |
| 1 | 1 | 2 |  |  |  |
| 2 | 1 | 3 |  |  |  |
| 1 | 2 | 4 |  |  |  |
| 2 | 2 | 5 |  |  |  |

The team plans to further discuss the test results once data is available. A pass on each of the cases detailed in the 3.3.1 Test Certification – People Counting is necessary in order to conduct position tracking testing.]

**3.4 Test Certification – Database -- Carol**

[The database stores a record of all tickets registered for a given event and informs the system whether or not those tickets have been used. To ensure proper operation of The Quicket System, the database must be able to properly check if a ticket is valid and then mark it as used. It must also be able to return an error for already used or missing tickets.

**3.4.1 Test Certification – Ticket ID Verification**

The most basic feature of the database is its ability to correctly check an incoming ticket ID and verify that the ID is in the database. If the ticket ID passed to the database matches one that is registered for the event, then the database must mark that ticket as used and pass an “OK” signal back to the system; however, if the ticket ID does not exist in the database, then a “not OK” signal must be returned. This test was performed six times and the results are shown in Table 3.10.

**Table 3.10: Ticket ID Verification**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID in Database?** | **Test Number** | **Signal from Database** | **Ticket Marked as Used** |
| Yes | 1 | 1 row in set | Query OK |
| Yes | 2 | 1 row in set | Query OK |
| Yes | 3 | 1 row in set | Query OK |
| No | 4 | Empty set | N/A |
| No | 5 | Empty set | N/A |
| No | 6 | Empty set | N/A |

In order to check if a ticket exists in the database, a query was sent to select from the database all entries that matched the incoming ticket ID. Below an example of such query is shown.

SELECT \* FROM guests WHERE ticketID = 1234567;

If the ticket exists within the database, then the query reports the entry found matching the ticket ID. Next, the database receives a command to update the entry so that the given ticket is marked as used. Below an example of this update command is shown.

UPDATE guests SET used = 1 WHERE ticketID = 1234567;

**3.4.2 Test Certification – Used Ticket ID Reporting**

The second feature required of the database is the ability to report that an incoming ticket has already been used. If a received ticket ID has already been checked in at an event, then the database must return an error to the system to indicate the problem. This test was performed six times and the results are shown in Table 3.11.

**Table 3.11: Used Ticket ID Reporting**

|  |  |  |
| --- | --- | --- |
| **Ticket Used** | **Test Number** | **Signal from Database** |
| Yes | 1 | Empty set (“Not OK”) |
| Yes | 2 | Empty set (“Not OK”) |
| Yes | 3 | Empty set (“Not OK”) |
| No | 4 | 1 row in set (“OK”) |
| No | 5 | 1 row in set (“OK”) |
| No | 6 | 1 row in set (“OK”) |

This test of database functionality looked specifically at the database’s ability to properly report whether or not a ticket had already been used. To accomplish this task, a query was sent to select the entry that matched the ticket ID and checked that the given ticket ID was unused. Below, an example of the query is shown.

SELECT \* FROM guests WHERE ticketID = 1234567 AND used = 0;

If the ticket was used, then the query returned no entries; however, if the ticket was not used, then an entry matching the ticket ID was returned. ]

**3.5 Test Certification – Error Signaling -- Alex**

[Error signaling is the final and most critical stage of The Quicket System. A streamlined ticketing process is useless if people can enter without a ticket. The system approaches error signaling on a pass-until-fail basis, which means the system needed to be tested on whether or not the green LEDs remained on for no ticketing error and if the system switched to the red LEDs when an error occurred. Per the design constraints, at least 20 of the correct color LEDs and 0 of the incorrect color LEDs must be on in each scenario. Additionally, when a mismatch is detected, the system must switch to the red LEDs for a minimum of 4 seconds.

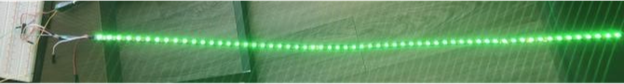
**3.5.1 Test Certification – Return to “OK” State**

One of the most important features of the pass-until-fail system is that the LEDs stay green when no errors exist and return to this state after an error has been indicated. This feature creates a streamlined process by limiting the disruption to groups of people who have the correct number of valid tickets. This feature was tested with a simulated “Not OK” signal from the Raspberry Pi indicating that the number of people detected matched the number of valid tickets scanned; after the system responds to the error, the LEDs are checked for the “OK” state output. This test was performed three times and the results are in Table 3.12.

**Table 3.12: Return to Green Test**

|  |  |  |
| --- | --- | --- |
| **Signal** | **Test Number** | **LEDs Return to Green** |
| Not OK | 1 | TRUE |
| Not OK | 2 | TRUE |
| Not OK | 3 | TRUE |

After the system responded to the error signal sent from the Raspberry Pi, the LEDs continually returned to an “OK” status with green LEDs.



**Figure 3.2 “OK” Status from LEDs**

Figure 3.2 above shows the LEDs returning to this “OK” state after responding to error.

**3.5.2 Test Certification – Invalid Comparison**

A feasible ticketing system must flag an error when a mismatch between the number of tickets and the number of people occurs; otherwise, the system becomes responsible for a loss of revenue and its usefulness is eliminated. The Quicket System’s error signaling was tested with a simulated “Not OK” signal from the Raspberry Pi indicating that the number of people detected did not match the number of valid tickets scanned. The system’s test case passed if the LEDs switched from green to red upon receiving this error signal. This test was performed three times and the results are in Table 3.13.

**Table 3.13: Invalid Comparison Test**

|  |  |  |
| --- | --- | --- |
| **Signal** | **Test Number** | **LEDs Turn Red** |
| Not OK | 1 | TRUE |
| Not OK | 2 | TRUE |
| Not OK | 3 | TRUE |

Upon receiving the error signal from the Raspberry Pi, the system immediately switched from the green LEDs to the red LEDs, properly signaling for outside intervention.



**Figure 3.3 “Not OK” Status from LEDs**

Figure 3.3 above shows The Quicket System responding to the error by changing the LED output to red.

**3.5.3 Test Certification – Number of LEDs**

The Quicket System must provide enough indication that a problem with the number of valid tickets occurs and be able to call for human intervention. Per the design constraints, a minimum of 20 LEDs of the proper color and 0 LEDs of the improper color must light up in each case. This feature was tested by providing both an “OK” signal that no problems arose and a “Not OK” signal that a problem with tickets exists. Each case was tested three times with the number of LEDs lit up in each case recorded, and these data are found in Table 3.14.

**Table 3.14: LED Number Test**

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal** | **Test Number** | **Number of Green LEDs** | **Number of Red LEDs** |
| OK | 1 | 60 | 0 |
| OK | 2 | 60 | 0 |
| OK | 3 | 60 | 0 |
| Not OK | 4 | 0 | 60 |
| Not OK | 5 | 0 | 60 |
| Not OK | 6 | 0 | 60 |

In each case, all of the LEDs on the LED strip lit up to the correct color: green for no error and red for an error, with zero instances of the incorrect color.

**3.5.4 Test Certification – Time Red**

Per the design constraints, the system must turn the LEDs red for a minimum of four seconds and a maximum of five seconds after a “Not OK” signal is received. This timeframe provides an opportunity for human intervention for current problems without hindering the next group’s means of entry. The system was tested three times through a simulated “Not OK” signal from the Raspberry Pi and the time taken to switch back from the red LEDs to the green LEDs was recorded and these data are in Table 3.15.

**Table 3.15: LED Time Test**

|  |  |  |
| --- | --- | --- |
| **Signal** | **Test number** | **Time LEDs are Red** |
| Not OK | 1 | 4.007 s |
| Not OK | 2 | 4.007 s |
| Not OK | 3 | 4.007 s |

The system consistently held the LEDs red for 4.007 seconds, thus meeting the system requirements outlined by the design constraints.]