***Non Functional Test Plan***

*Table of Contents*

1. *Introduction*
2. *Test Scenarios*
   1. *numSeatsAvailable API Cases*
   2. *findAndHoldSeats API Cases*
   3. *reserveSeats API Cases*
   4. *Common Cases*
3. *Automation*
4. *To-Do*
5. *References*

*Introduction*

The purpose of this document is to capture use cases for functional tests which are required to test all components of the Ticket Service system according to the given requirements.

**Test Scenarios**

*FT1: numSeatsAvailable API Cases*

*Setup the data based on the preconditions mentioned. Then make the API call using the url specified under Test*

|  |  |  |  |
| --- | --- | --- | --- |
| *Test#* | *Description* | *Steps* | *Expected Results* |
| *FT1-1* | *Basic Test: When no reservation/holds are available, check the number of seats available* | *Precondition:*  *#of seats available for booking = Max*  *Test:*  <https://ticketsystem.com/numSeatsAvailable> | *# of seats returned by API=Max*  *JSON Response:*  JSON Response:  {"status": "OK", "mobile\_url": "", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "url": "https:/[ticketsystem.com/numSeatsAvailable](https://ticketsystem.com/numSeatsAvailable)/", "Date": "2017-12-27T11:11:42", "available": **17820**} |
| *FT1-2* | *Basic Test: When all seats are either reserved or on holds, check the number of seats available* | *Precondition:*  *#of seats available for booking = 0*  *Test:*  <https://ticketsystem.com/numSeatsAvailable> | *# of seats returned by API=0*  JSON Response:  {"status": "OK", "mobile\_url": "", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "url": "https:/[ticketsystem.com/numSeatsAvailable](https://ticketsystem.com/numSeatsAvailable)/", "Date": "2017-12-27T11:11:42", "available": **0**} |
| *FT1-3* | *When all seats are either reserved or on hold, except the 4PM shows on all days* | *Precondition:*  *#of seats available for booking at 8PM on all days = 0*  *Test:*  <https://ticketsystem.com/numSeatsAvailable> | *# of seats returned by API=8910*  JSON Response:  {"status": "OK", "mobile\_url": "", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "url": "https:/[ticketsystem.com/numSeatsAvailable](https://ticketsystem.com/numSeatsAvailable)/", "Date": "2017-12-27T11:11:42", "available": **8910**} |
| *FT1-4* | *When all seats are either reserved or on hold, except the 8PM shows on all days* | *Precondition:*  *#of seats available for booking at 4PM on all days = 0*  *Test:*  <https://ticketsystem.com/numSeatsAvailable> | *# of seats returned by API=8910*  JSON Response:  {"status": "OK", "mobile\_url": "", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "url": "https:/[ticketsystem.com/numSeatsAvailable](https://ticketsystem.com/numSeatsAvailable)/", "Date": "2017-12-27T11:11:42", "available": **8910**} |
| *FT1-5* | *When all seats are either reserved or on hold, except the 4PM shows on a selected day* | *Precondition:*  *#of seats available for booking at 8PM on 1/1/2018 = 0*  *Test:*  <https://ticketsystem.com/numSeatsAvailable> | *# of seats returned by API=297*  JSON Response:  {"status": "OK", "mobile\_url": "", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "url": "https:/[ticketsystem.com/numSeatsAvailable](https://ticketsystem.com/numSeatsAvailable)/", "Date": "2017-12-27T11:11:42", "available": **297**} |
| *FT1-6* | *When all seats are either reserved or on hold, except the 8PM shows on a selected day* | *Precondition:*  *#of seats available for booking at 4PM on 1/1/2018 = 0*  *Test:*  <https://ticketsystem.com/numSeatsAvailable> | *# of seats returned by API=297*  JSON Response:  {"status": "OK", "mobile\_url": "", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "url": "https:/[ticketsystem.com/numSeatsAvailable](https://ticketsystem.com/numSeatsAvailable)/", "Date": "2017-12-27T11:11:42", "available": **297**} |
| *FT1-7* | *Check seat calculation logic in Hold status* | *Precondition:*  *#of seats in Hold status = 4*  *#of seats in Reserve status = 0*  *#of seats booked = 0*  *Test:*  <https://ticketsystem.com/numSeatsAvailable> | *# of seats returned by API=17816*  JSON Response:  {"status": "OK", "mobile\_url": "", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "url": "https:/[ticketsystem.com/numSeatsAvailable](https://ticketsystem.com/numSeatsAvailable)/", "Date": "2017-12-27T11:11:42", "available": ***17816***} |
| *FT1-8* | *Check seat calculation logic in Reserve status* | *Precondition:*  *#of seats in Reserve status = 10*  *#of seats in Hold status = 0*  *#of seats booked = 0*  *Test:*  <https://ticketsystem.com/numSeatsAvailable> | *# of seats returned by API=17810*  JSON Response:  {"status": "OK", "mobile\_url": "", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "url": "https:/[ticketsystem.com/numSeatsAvailable](https://ticketsystem.com/numSeatsAvailable)/", "Date": "2017-12-27T11:11:42", "available": ***17810***} |
| *FT1-9* | *Check Seat selection logic in Hold and Reserve status* | *Precondition:*  *#of seats in Hold status = 4 and Reserve status = 10*  *#of seats booked = 0*  *Test:*  <https://ticketsystem.com/numSeatsAvailable> | *# of seats returned by API=17806*  JSON Response:  {"status": "OK", "mobile\_url": "", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "url": "https:/[ticketsystem.com/numSeatsAvailable](https://ticketsystem.com/numSeatsAvailable)/", "Date": "2017-12-27T11:11:42", "available": ***17806***} |

*FT2: findAndHoldSeats API Cases*

*Setup the data based on the preconditions mentioned. Then make the API call using the url specified under Test*

|  |  |  |  |
| --- | --- | --- | --- |
| *Test#* | *Description* | *Steps* | *Expected Results* |
| *FT2-1* | *Basic Test: User holds 15 seats with customerEmail address=abc@abc.com* | *Precondition:*  *#of seats available for booking = Max*  *Test:*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats?seats=15&customerEmail=abc@abc.com* | *# of seats held by API=15 for* [*email=abc@abc.com*](mailto:emaol=abc@abc.com)  *JSON Response:*  {"status": "OK", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "Date": "2017-12-27T11:11:42", “Confirmation”: “**hold**”, “Settime”:”300”, “*customerEmail* ”:”[**abc@abc.com**](mailto:abc@abc.com)”, “seats”:”**15**”, “holdId”:”**4**”} |
| *FT2-2* | *Call API with empty/Null/type mismatch numSeats parameter* | *Precondition:*  *#of seats available for booking = Max*  *Test:*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats?seats=abc&customerEmail=abc@abc.com*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats?seats=&customerEmail=abc@abc.com*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats?customerEmail=abc@abc.com* | *Graceful Error handling with user friendly message* |
| *FT2-3* | *Call API with empty/Null/type mismatch customerEmail parameter* | *Precondition:*  *#of seats available for booking = Max*  *Test:*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats?seats=15&customerEmail=*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats?seats=15&customerEmail=1*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats?seats=15* | *Graceful Error handling with user friendly message* |
| *FT2-4* | *Call API with empty/Null/type mismatch numSeats and customerEmail parameters* | *Precondition:*  *#of seats available for booking = Max*  *Test:*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats?numSeats=&customerEmail=*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats?numSeats=sfs&customerEmail=fs* | *Graceful Error handling with user friendly message* |
| *FT2-5* | *Call API with numSeats, customerEmail and 2 extra parameters* | *Precondition:*  *#of seats available for booking = Max*  *Test:*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats?seats=15&customerEmail=abc@abc.com&extra1=12&extra2=23* | *Graceful Error handling with user friendly message* |
| *FT2-6* | *Combine API Scenario: A User selects 2 seats, the tickets were the last available tickets in the system. Another User calls Discover API* | *Precondition:*  *#of seats available for booking = 2*  *Test:*  *User 1:* [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats?seats=2&customerEmail=abc@abc.com*  *User 2:*  <https://ticketsystem.com/numSeatsAvailable> | *# of seats held by API=****2*** *for* [*email=abc@abc.com*](mailto:emaol=abc@abc.com)  *JSON Response:*  {"status": "OK", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "Date": "2017-12-27T11:11:42", “Confirmation”: “**hold**”, “Settime”:”300”, “*customerEmail* ”:”[**abc@abc.com**](mailto:abc@abc.com)**”**, “seats”:”**2**”, “holdId”:”**4**”}  *For User 2:*  *# of seats returned by API=0*  *JSON Response:*  JSON Response:  {"status": "OK", "mobile\_url": "", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "url": "https:/[ticketsystem.com/numSeatsAvailable](https://ticketsystem.com/numSeatsAvailable)/", "Date": "2017-12-27T11:11:42", "available": **0**} |
| *FT2-7* | *Combine API Scenario: User selects 2 seats, does not like the seats returned by the system, clicks refresh again. Another user calls Discover API* | *Precondition:*  *#of seats available for booking = 2*  *Test:*  *User 1:* [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats?seats=2&customerEmail=abc@abc.com*  *Wait 400 seconds*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats?seats=2&customerEmail=abc@abc.com*  *User 2:*  <https://ticketsystem.com/numSeatsAvailable> | *# of seats held by API=2 for* [*email=abc@abc.com*](mailto:emaol=abc@abc.com)  *JSON Response:*  {"status": "OK", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "Date": "2017-12-27T11:11:42", “Confirmation”: “**hold**”, “Settime”:”300”, “*customerEmail* ”:”[abc@abc.com](mailto:abc@abc.com)”, “seats”:”**2**”, “holdId”:”**4**”}  *# of seats held by API=2 for* [*email=abc@abc.com*](mailto:emaol=abc@abc.com)  *JSON Response:*  {"status": "OK", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "Date": "2017-12-27T11:11:42", “Confirmation”: “**hold**”, “Settime”:”300”, “*customerEmail* ”:”[abc@abc.com](mailto:abc@abc.com)”, “seats”:”**2**”, “holdId”:”**44**”  }  For user 2 -  *# of seats returned by API=0*  *JSON Response:*  JSON Response:  {"status": "OK", "mobile\_url": "", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "url": "https:/[ticketsystem.com/numSeatsAvailable](https://ticketsystem.com/numSeatsAvailable)/", "Date": "2017-12-27T11:11:42", "available": **0**} |
| *FT2-8* | *Combine API Scenario: User selects 2 seats and before the*  *server response is received, user abandons the request, the tickets were the last available tickets in the system. Another user calls Discover API* | *Precondition:*  *#of seats available for booking = 2*  *Test:*  *User 1:* [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*findAndHoldSeats?seats=2&customerEmail=abc@abc.com*  *Disconnect from internet*  *User 2:*  <https://ticketsystem.com/numSeatsAvailable> | *# of seats held by API=2 for* [*email=abc@abc.com*](mailto:emaol=abc@abc.com)  *JSON Response:*  {"status": "OK", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "Date": "2017-12-27T11:11:42", “Confirmation”: “**hold**”, “Settime”:”300”, “*customerEmail* ”:”[abc@abc.com](mailto:abc@abc.com)”, “seats”:”**2**”, “holdId”:”**4**”}  For user 2:  *# of seats returned by API=0*  *JSON Response:*  JSON Response:  {"status": "OK", "mobile\_url": "", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "url": "https:/[ticketsystem.com/numSeatsAvailable](https://ticketsystem.com/numSeatsAvailable)/", "Date": "2017-12-27T11:11:42", "available": **2}** |

*FT3: reserveSeats API Cases*

*Setup the data based on the preconditions mentioned. Then make the API call using the url specified under Test*

|  |  |  |  |
| --- | --- | --- | --- |
| *Test#* | *Description* | *Steps* | *Expected Results* |
| *FT3-1* | *Basic Test: User reserves 15 seats with customerEmail address=abc@abc.com* | *Precondition:*  *#of seats available for booking = Max*  *holdid=4*  *Test:*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*reserveSeats?holdId=4&customerEmail=abc@abc.com* | *# of seats reserved by API=15 for* [*email=abc@abc.com*](mailto:emaol=abc@abc.com)  JSON Response:  {"status": "OK", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "Date": "2017-12-27T11:11:42", “Confirmation”: “**reserve**”, “Settime”:”600”, “*customerEmail*”:”[abc@abc.com](mailto:abc@abc.com)”, “Seats”:”**15**”.”ReservationId”:”**res4**”, “holdId”:”**4**”} |
| *FT3-2* | *Call API with empty/Null/type mismatch holdId parameter* | *Precondition:*  *#of seats available for booking = Max*  *Test:*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*reserveSeats?holdId=&customerEmail=abc@abc.com*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*reserveSeats?holdId=4ljjf&customerEmail=abc@abc.com*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*reserveSeats?customerEmail=abc@abc.com* | *Graceful Error handling with user friendly message* |
| *FT3-3* | *Call API with empty/Null/type mismatch customerEmail parameter* | *Precondition:*  *#of seats available for booking = Max*  *Test:*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*reserveSeats?holdId=4&customerEmail=*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*reserveSeats?holdId=4*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*reserveSeats?holdId=4&customerEmail=78* | *Graceful Error handling with user friendly message* |
| *FT3-4* | *Call API with empty/Null/type mismatch holdId and customerEmail parameters* | *Precondition:*  *#of seats available for booking = Max*  *Test:*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*reserveSeats?holdId=&customerEmail=*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*reserveSeats?holdId=4jk&customerEmail=90*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*reserveSeats* | *Graceful Error handling with user friendly message* |
| *FT3-5* | *Call API with holdId, customerEmail and 2 extra parameters* | *Precondition:*  *#of seats available for booking = Max*  *Test:*  [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*reserveSeats?holdId=4&customerEmail=abc@abc.com&extra1=12&extra2=23* | *Graceful Error handling with user friendly message* |
| *FT3-6* | *Combine API Scenario: A User registers 2 seats, the tickets were the last available tickets in the system. Another User calls Discover API* | *Precondition:*  *#of seats available for booking = 2*  *Test:*  *User 1:* [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*reserveSeats?holdId=2&customerEmail=abc@abc.com*  *User 2:*  <https://ticketsystem.com/numSeatsAvailable> | *# of seats held by API=****2*** *for* [*email=abc@abc.com*](mailto:emaol=abc@abc.com)  *JSON Response:*  JSON Response:  {"status": "OK", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "Date": "2017-12-27T11:11:42", “Confirmation”: “**reserve**”, “Settime”:”600”, “*customerEmail*”:”[abc@abc.com](mailto:abc@abc.com)”, “Seats”:”2”.”ReservationId”:”**res4**”, “holdId”:”**4**”}  *For User 2:*  *# of seats returned by API=0*  *JSON Response:*  JSON Response:  {"status": "OK", "mobile\_url": "", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "url": "https:/[ticketsystem.com/numSeatsAvailable](https://ticketsystem.com/numSeatsAvailable)/", "Date": "2017-12-27T11:11:42", "available": **0**} |
| *FT3-7* | *Combine API Scenario: User selects 2 seats and before the*  *server response is received, user abandons the request, the tickets were the last available tickets in the system. Another user calls Discover API* | *Precondition:*  *#of seats available for booking = 2*  *Test:*  *User 1:* [https://ticketsystem.com/](https://ticketsystem.com/numSeatsAvailable)*reserveSeats?holdId=2&customerEmail=abc@abc.com*  *Disconnect from internet*  *User 2:*  <https://ticketsystem.com/numSeatsAvailable> | *# of seats held by API=****2*** *for* [*email=abc@abc.com*](mailto:emaol=abc@abc.com)  *JSON Response:*  JSON Response:  {"status": "OK", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "Date": "2017-12-27T11:11:42", “Confirmation”: “**reserve**”, “Settime”:”600”, “*customerEmail*”:”[abc@abc.com](mailto:abc@abc.com)”, “Seats”:”2”.”ReservationId”:”**res4**”, “holdId”:”**4**”}  *For User 2:*  *# of seats returned by API=0*  *JSON Response:*  JSON Response:  {"status": "OK", "mobile\_url": "", "hash": "a7fd6484f414f467aaa56d4564fd06c1", "url": "https:/[ticketsystem.com/numSeatsAvailable](https://ticketsystem.com/numSeatsAvailable)/", "Date": "2017-12-27T11:11:42", "available": **2**} |

*FT4: Common Cases*

*These tests are common to all 3 API above*

|  |  |  |
| --- | --- | --- |
| *Test#* | *Description* | *Expected Results* |
| *FT4-1* | *Check Malformed URL* | *Graceful Error handling with user friendly message* |
| *FT4-2* | *Check Return Type* | *Graceful Error handling with user friendly message* |
| *FT4-3* | *Check Response Code* | *Graceful Error handling with user friendly message* |
| *FT4-4* | *Check JSON format* | *Graceful Error handling with user friendly message* |
| *FT4-5* | *Check Request Header* | *Graceful Error handling with user friendly message* |
| *FT4-6* | *Check Response Header* | *Graceful Error handling with user friendly message* |
| *FT4-7* | *Log File Verification* | *Benchmark Info, warnings, errors and exceptions. Enable debug mode* |
| *FT4-8* | *Miscellaneous* | *Using browsers developers tool check console logs, network logs, performance logs, Memory logs, Security logs Tab. API time outs should be handled gracefully.* |

*Automation:*

Using TestNG along with JSON parser and HttpURLConnections, all the use cases above can be automated once the API definition is available. A very simple test has been implemented for demo.

Assuming Struts are used to create API’s, using testNG annotations, we can make a direct call to the API using HttpURLConnections. We can analyse the request and response received from the API call to a> match the requirements, b> test scenarios c> edge cases d> data validation

In the example provided:

Assuming the data is available according to test plan

Call the API Test:

*URL url = new URL(“http://localhost:/numSeatsAvailable");*

Check Response Code:

*if (conn.getResponseCode() != 200) {*

*throw new Exception(”HTTP error code : ”+ conn.getResponseCode());*

*}*

To unit test the API code, the tests highlighted in Common tests scenario can be added as a JUnit Assert

*To-Do:*

1. Document sign off criteria
2. Document mobile user scenario
3. Copy test cases to TestRail Account
4. Elaborate on automation tests

*References:*

[*https://seleniumbycharan.wordpress.com/2015/07/26/api-testing-using-selenium-webdriver-and-java/*](https://seleniumbycharan.wordpress.com/2015/07/26/api-testing-using-selenium-webdriver-and-java/)

[*https://testing.googleblog.com/2011/09/10-minute-test-plan.html*](https://testing.googleblog.com/2011/09/10-minute-test-plan.html)

[*https://seleniumbycharan.wordpress.com/2015/07/26/api-testing-using-selenium-webdriver-and-java/*](https://seleniumbycharan.wordpress.com/2015/07/26/api-testing-using-selenium-webdriver-and-java/)