***In the design phase you will be:***

1. Design and prioritize test cases according to the Software Requirements Specification using positive and negative testing.
2. Design and build test environment.
3. Identify and prepare test data and ensure it is properly loaded in the test environment.
4. Create a test script that includes Test suite, Test ID’s, Test title, Test description, Precondition, Test steps, Test Data and Expected result

**Auth - Create Token**

T1 – Type of username must be string

Positive testing:

1. Execute a happy scenario where all the requirements are achieved:

* default username and password.
* Response should be token number of string datatype.
* Status code must be 200 OK.

1. A string username with numbers.
2. A string username can have special character “\_” in between example: “mo\_yosry23”.
3. Ensure that the API response returns a proper error value (404)

Negative testing:

1. Blank string username.
2. String Username special character only (equivalence partitioning).
3. String Username with numbers only (equivalence partitioning).
4. String Username with spaces at the beginning (boundary value analysis).
5. String Username with spaces at the end (boundary value analysis).
6. String Username with spaces in the middle (boundary value analysis).
7. A string username with special character at the beginning except for “\_” (boundary value analysis).
8. A string username with special character at the end except for “\_” (boundary value analysis).
9. Long string username “mmmmmmmmmmmmmmmmm.....” (performance test).
10. Integer username.
11. Boolean username.
12. Empty username body ("username" :).
13. Delete the whole username line from the body.
14. Delete the whole password line from the body.
15. GET, PUT, PATCH, and DELETE requests shouldn’t work.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T1.1 Password must be String.

Negative testing:

1. Blank string password (equivalence partitioning).
2. String Password smaller than 8 digits
3. String with special characters only (equivalence partitioning).
4. String with numbers only (equivalence partitioning).
5. Long string (performance testing).
6. String password with spaces at the beginning (boundary value analysis).
7. String password with spaces at the end (boundary value analysis).
8. Password without the whole string example: ("password" : )
9. Integer data type password.
10. Boolean data type password.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T1.2 – Username default value is always “admin” (test to see if it can be changed and if we can add numbers, special characters, leave it blank or have first character blank).

Negative testing:

1-Different username than admin (already tried numbers, special characters, leave it blank or have first character blank in the first condition).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T2 – Type of token must be String (already verified above).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T3 – Status code upon success is 200 OK (already verified above).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T4 – Content type is JSON (there must be content type in header so try to test it without the content type).

Negative testing:

1- Send request without content type in header.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T6 – This only works with POST call (already verified above).

T7 – URL is <https://restful-booker.herokuapp.com/auth> (try to add parameters to it and test it to see if it will show different results).

Negative testing:

1-Change the URL parameters only (it should show an error)

**Booking - GetBookingIds**

T8 – First and Last name **parameters** must be String.

Positive testing:

1-Happy scenario:

* First and last name as parameters with string type
* First name only
* Last name only
* Check-in and out as parameters in the date format
* Check in only
* Check out only
* URL without parameters
* Status code must be 200 OK
* Response is booking ids of JSON type
* Header has Accept and application/json.
* The request is GET

Negative testing:

1. Numbers instead of first name.
2. Special characters instead of first name.
3. White spaces instead of first name.
4. First name with numbers.
5. First name with special character.
6. Upper case letter name.
7. First name with a long length.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T9 – Check-in and check-out are both Strings **parameters** but can only be in the date format CCYY-MM-DD ex: 2014-03-13.

Negative testing:

1. Wrong date format (year only, month only, day only).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T10 – All the above is optional and the URL can be executed without them.

Positive testing: try URL without any parameters

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T12 – Upon success all the booking ids are returned in the response

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T13 – Header must have “Accept” of type JSON default value application/json.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T14 – The URL only works with the GET call.

Negative testing: try with POST, PUT, PATCH, DELETE

**Booking - GetBooking**

T17 – Header must have “Accept” of type JSON default value application/json.

Positive testing:

1-Happy scenario:

* Header has accept of value application/json.
* Parameter is an ID number that exists.
* Status code is 200 OK.
* Response is the full body.
* Request is GET

Negative testing:

1- Without accept in the header.

2- Parameter is ID number that doesn’t exist.

3- Parameter is long ID number.

4- Parameter is Special characters.

T20 – Upon successful response the returned data must contain firstname (String), lastname (String), totalprice (Numeric), depositpaid (Boolean), bookingdates (Object), checkin (String date), checkout (String date), additional needs (String).

Negative testing on the URL:

1. Try request DELETE.

**Booking - CreateBooking**

T28 – Content type and accept are both of type JSON.

Positive testing:

1-Happy scenario:

* Content type and accept are both of type JSON.
* status code must be 200.
* Request has the full body with the right data types and the right format.
* Response must be the fully **created** body with the booking id number.
* Request POST.

Negative testing:

1-PATCH, PUT, and DELETE requests

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T30 – Upon successful response the returned data must contain bookingid (Numeric), booking (Object), firstname (String), lastname (String), totalprice (Numeric), depositpaid (Boolean), bookingdates (Object), checkin (String date), checkout (String date), additional needs (String).

Negative testing: try to write the wrong body format to see if you will get the same response

T31 – First and Last name must be String.

Negative testing:

1. String Numbers.
2. String Special characters.
3. String White spaces (only white spaces and at the beginning and in the middle and at the end)
4. String Long name length.
5. String Upper case letter name (mixed, only first letter, and upper case only).
6. First name without a value.
7. First name without a key.
8. First name value of type Integer.
9. First name value of type Boolean.
10. First name value of special characters only.
11. Key of first name is of Integer and Boolean value example (123: “Jim” | true : “Jim”).
12. Both key and value empty.

*All the above can be applied to the last name*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T32 – Total price must be numeric.

Negative testing:

1. Total price long length number.
2. Total price with a float number.
3. Total price value is empty (“totalprice”: ).
4. Total price key itself is empty (“”:111).
5. Total price key is integer and Boolean.
6. Total price value of String data type.
7. Total price value of Boolean data type.
8. Total price value is special characters only.
9. Both key and value empty.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T33 – Deposit paid must be Boolean.

Negative testing:

1. Leave the value empty.
2. Add white spaces.
3. Add special characters in the value.
4. Change data type (int, string).
5. Leave the key empty.
6. Add different data types in the key.
7. Leave both key and value empty.

T34 – Booking dates must have 2 values, checkin and checkout.

Negative testing:

1. Leave both values empty.
2. Add white spaces to both values.
3. Add white spaces at the beginning and the end and in the middle (boundary value analysis).
4. Add Invalid date format (missing year, month, or day, date with / instead of - ) (Equivalence Partitioning).
5. Add the value with different data types (integer, Boolean, objects and arrays).
6. Add special characters to the value.
7. Long length string (performance testing).
8. Leave key empty
9. Test dates in the past, future, and present.
10. Check for date ranges with invalid order (checkin after checkout).
11. Check how long the person can be staying so extend the range between checkin and checkout as large as possible.

T35 – Checkin and checkout must be String (negative testing above).

T36 – Checkin and checkout must be in the format CCYY-MM-DD ex: 2014-03-13. (negative testing above)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T37 – Additional need must be String.

T38 – Additional need is optional.

Negative testing:

1. Try to write a very long sentence to see the boundaries of the additional needs
2. Leave the value empty
3. Leave the key empty
4. Enter different data types in value (integer, Boolean, array, objects).
5. Enter white spaces only in the value.
6. Try to enter special characters only inside the string.
7. Try special characters instead of the string itself example ("additionalneeds" : @#$#@$@).

Negative testing:

1. Try with an empty body.
2. Try with some parts of the body deleted.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Booking - UpdateBooking**

T38 – Content type and accept are both of type JSON.

Positive testing:

1-Happy scenario:

* Content type and accept are both of type JSON.
* status code must be 200.
* Request has the full body with the right data types and the right format.
* Response must be a fully **modified** body.
* Request PUT.
* Authorization can be in form of Cookie (token value) or Authorization (Default value: Basic YWRtaW46cGFzc3dvcmQxMjM=) in the header.

Negative testing:

1. Without content Type
2. Without accept
3. Without Authorization
4. Without all 3 of them.
5. Change the token value.
6. Change cookie value.

T39 – There must be authorization token to access PUT endpoint. (Negative testing above).

T40 – Authorization can be in form of Cookie (token value) or Authorization (Default value: Basic YWRtaW46cGFzc3dvcmQxMjM=) in the header. (Negative testing above).

T41 – Upon success update status code must be 200 OK.

T42 – URL must have parameter of id number that will be updated.

Negative testing:

1. Try without an id.
2. Try with an id that doesn’t exist.

T43 – All the request body must be written since this is a PUT call but not all of it must be updated.

T44 – Upon successful response the returned data must contain firstname (String), lastname (String), totalprice (Numeric), depositpaid (Boolean), bookingdates (Object), checkin (String date), checkout (String date), additional needs (String).

T45 – First and Last name must be String.

Negative testing:

1. String Numbers.
2. String Special characters.
3. String White spaces (only white spaces and in the beginning)
4. String Long name length.
5. String Upper case letter name (mixed, start, and upper case only).
6. First name without a value.
7. First name without a key.
8. First name key of type Integer.
9. First name key of type Boolean.
10. First name value of special characters only.
11. Key of first name is of Integer and Boolean value example (123: “Jim” | true : “Jim”).
12. Both key and value empty.

*All the above can be applied to the last name*

T46 – Total price must be numeric.

Negative testing:

1. Total price long length number.
2. Total price with a float number.
3. Total price value is empty (“totalprice”: ).
4. Total price key itself is empty (“”:111).
5. Total price key is integer and Boolean.
6. Total price value of String data type.
7. Total price value of Boolean data type.
8. Total price value is special characters only.
9. Both key and value empty.

T47 – Deposit paid must be Boolean.

Negative testing:

1. Leave the value empty.
2. Add white spaces.
3. Add special characters in the value.
4. Change data type (int, string).
5. Leave the key empty.
6. Add different data types in the key.
7. Leave both key and value empty.

T48 – Booking dates must have 2 values, checkin and checkout.

Negative testing:

1. Leave both values empty.
2. Add white spaces to both values.
3. Add white spaces at the beginning and the end and in the middle (boundary value analysis).
4. Add Invalid date format (missing year, month, or day, date with / instead of - ) (Equivalence Partitioning).
5. Add the value with different data types (integer, Boolean, objects and arrays).
6. Add special characters to the value.
7. Long length string (performance testing).
8. Leave key empty
9. Test dates in the past, future, and present.
10. Check for date ranges with invalid order (checkin after checkout).
11. Check how long the person can be staying so extend the range between checkin and checkout as large as possible.

T49 – Checkin and checkout must be String (tested above).

T50 – Checkin and checkout must be in the format CCYY-MM-DD ex: 2014-03-13 (tested above).

T51 – Additional need must be String.

Negative testing:

1. Try to write a very long sentence to see the boundaries of the additional needs
2. Leave the value empty
3. Leave the key empty
4. Enter different data types in value (integer, Boolean, array, objects).
5. Enter white spaces only in the value.
6. Try to enter special characters only inside the string.
7. Try special characters instead of the string itself example ("additionalneeds" : @#$#@$@).
8. Leave both key and value empty

Negative testing:

1. Try with an empty body.

**Booking – PartialUpdateBooking**

**Same test cases as above except that you don’t have to enter the full body.**

**Booking - DeleteBooking**

T66 – There must be authorization token to access DELETE endpoint.

Positive testing:

1. URL must have parameter of id number that will be DELETED.
2. With Authorization (token or cookie).
3. Status code 201.
4. DELETE request.

Negative testing:

1. Without authorization.
2. Change token and cookie value.
3. Change request

T67 – URL must have parameter of id number that will be DELETED.

T68 – Upon success update status code must be 201 Created.

**Ping – HealthCheck**

Positive testing:

1- **GET request with the URL** <https://restful-booker.herokuapp.com/ping>

2- it should return status code 201

3-it won’t return anything in the response it just for checking the ping

Negative testing:

1. Try different requests.

E2E Testing Scenarios:

1. Create token
2. Book a room
3. Make sure room is added check book id is in the main list
4. Check the room data it self do a get booking with the room’s id
5. Update room by put request
6. Get request to check if room is updated
7. Partial update with Patch request
8. Get request to check if room is updated
9. Delete room
10. Get all id’s to check if it’s deleted
11. Do a ping check