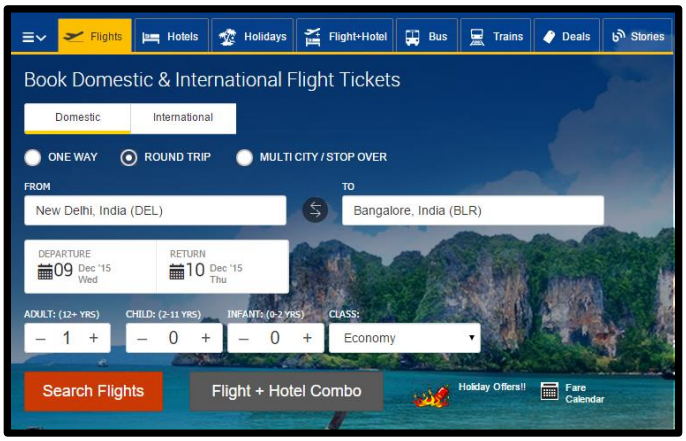
TEST DESIGN EXAMPLE: MAKE MY TRIP.



Let us consider searching and booking a flight ticket using make my trip as an example to illustrate the test design at different quality levels.

"Joe wants to book a round trip flight ticket from Bangalore to New Delhi for the dates 26th January 2016 and 7th February 2016." – **Business work flow**

What he needs to book the flight?

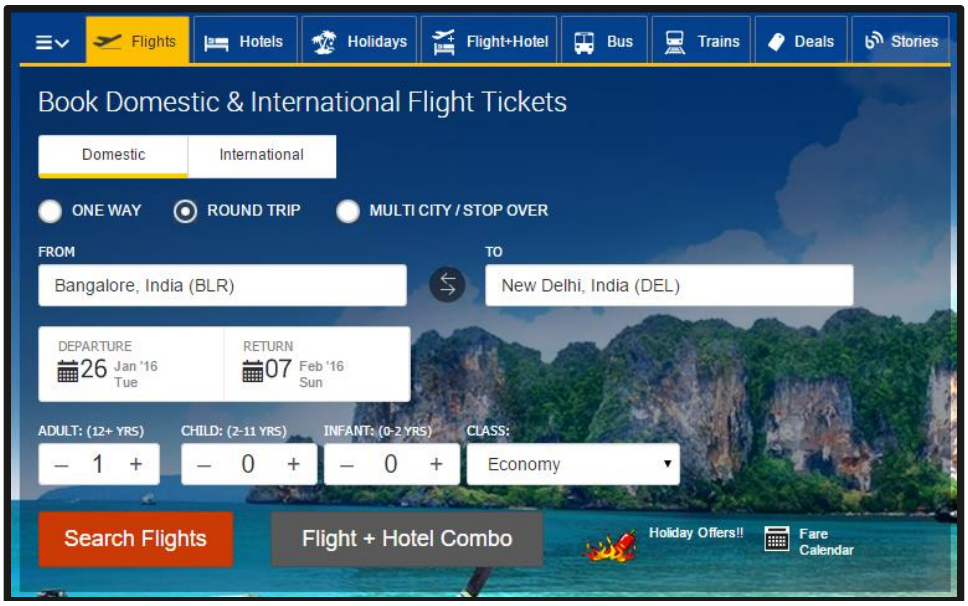
1. He should be able to search for the availability of flight
2. He should be able to select the flight, if available, suitable to his travel
3. He should be able to provide the information for booking the ticket
4. He should be able to make the payment
5. He should receive the confirmation of booking.
   (1-5 are the **user requirements** that is used to do the business work flow)

System will support these requirements by various technical features.

- System should allow the user to search
- System should display the search result
- System should allow to provide personal information
- System should allow making the payment etc.
  These are called the **technical features** of the system. These are used to perform a user requirement.

Finally these features of the system will be implemented using various design components like call to some API, using some class objects etc. These design components used to build/implement the feature are called **Structural Components**.

QL1 – Input cleanliness



As we know the objective of QL1 is to ensure that the bad values are rejected by the system. We need to consider this at the feature level as each feature will be taking various input values. So Entity under Test (EUT) at this level will be typically a Feature.
Let us consider the feature 'System should allow the user to search the flights' and see how to design QL1 scenarios and cases for this feature.

Steps

1. *Identify the inputs required by the feature and the specification of each input.*

No	Input	Input specification
1	Type of travel	Domestic or international. Selection from pre-defined values.
2	Trip type	One way, round trip or multi-city. Predefined values.
3	Departure city	List of cities available. Pre-configured set of values.
4	Destination city	List of cities available. Pre-configured set of values.
5	Departure date	Should not be less than current date. Selection using calendar.
6	Return date	Should not be less than departure date, current date. Selection using calendar.
7	No. of adult	Numeric 1-14
8	No. of child	Numeric 0-14
9	No. of infant	Numeric 0-9 should not be greater than #of adult.
10	Class of travel	Accepts Economy, Premium Economy and Business class. – Predefined set of values.

What to do if input specification not available for each input?

Since the objective is to validate whether invalid inputs are rejected by the system, it is mandatory to know what the bad values for each input are. If specification not available as written document, we may need to ask/question the concerned people and get the information.

2. *Identify the invalid values for each input as per the specification*

Test cases at QL1 are actually set of invalid set of values for each input. So by finding out the bad values for each input is directly coming up with the test data at QL1.

No	Input	Invalid data set (test data set/ test cases)
1	Type of travel	Any value other than domestic/international. Since the GUI is designed in such a way that you cannot enter any value other than these two, from GUI level this input no need to validate for bad inputs. What we need to verify is, the provided value for this field is correct or not. (That is we need to check the types are Domestic/International)
2	Trip type	Check for the correctness of options provided. (One way, round trip, multicity/stop over)
3	Departure city	Check that the list of values populated in the combo box is correct.
4	Destination city	Check that the list of values populated in the combo box is correct. This field is depending on Departure and the destination cannot be same as the departure city.
5	Departure date	Any date less than the current date, any date greater than the allowed advance booking date.
6	Return date	Any date less than the current date, any date greater than the allowed advance booking date, any date less than the departure date.
7	No. of passengers	Adult – less than 1 and greater than 14 Children – less than 0 and greater than 14 Infant – less than 0 and greater than 9 Total #passenger >14, #of infants > # of adults. These are the invalid values to be considered.
10	Class of travel	Check for the correctness of options provided. (Economy, Business, premium economy)

How do we look for the bad/invalid values for each input?

In HBT, potential defect types at QL1 is pre-defined. Irrespective of the application issues dute input validation can be categorized in to any of the below defect types. (1) Data Type issue, (2) Data Format issue, (3) Data Boundary issue, (4) Data Dependency issue and (5) Data value set issue. For each input think of validation issues of these types, in the current context and identify the invalid values for each input.

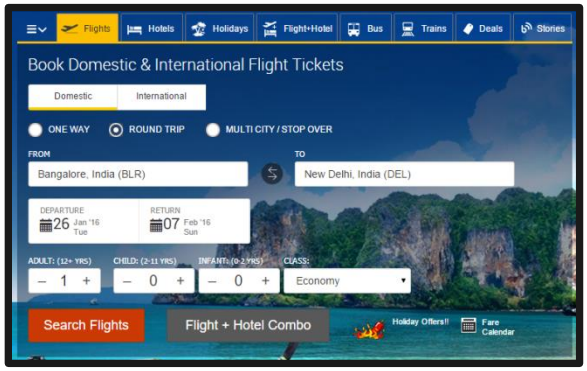
QL2 – Interface cleanliness

Objective of QL2 is to ensure that the interface through which the input receives is clean. In this example there are multiple interfaces for accepting input for search option, interface to display the search result, interface to confirm booking, payments etc.

Steps

1. *Identify the type of interface for the feature. Typically the interface can be a GUI or an API which will be invoking by command line or any other way of invocation.*

   For both GUI and API, there are standard set of potential defects defined/mapped by HBT that we need to look for at this level. In this case the interface is a GUI as shown below.



2. *Understand the interface specification and choose and understand the appropriate meaning of the selected potential defect in this context.*

   Some of the potential defects at this level are as listed below.

Potential Defect Type (PDT)	**Applicable?**	**What does it mean in the context?**
Missing elements	Yes	Travel type, Tripe type, from, to etc. are the interface elements expected.
Missing icons	Yes	Departure and Return date should be provided with calendar icon, Logo should present at bottom,
Grouping of elements	Yes	From/To dates, passenger number details have to be grouped separately.
Component alignments	Yes	Components should be left aligned, Buttons should be at the bottom, Icon for fare calendar should be at the bottom
Color/Font	Yes	"Search Flight" button should be in orange color background with white color label.
----------	----------	----------

Go through each of the PDTs and check whether it is applicable in this context, if yes understand what it means. Create a check list for the same and validate against the Feature Interface.

QL4 – Behavioral correctness

At this level, we ensure that functional behavior (completeness/correctness) of the EUT is implemented as indented. As we know behavior is nothing but a combination of conditions. So first we need to identify the conditions that govern the behavior of the entity.

Let us consider the feature "System should allow the user to search for the flight".

Steps

1. *Understand the descriptive behavior of the feature and identify the conditions that govern the behavior.*

   For our understanding, the descriptive behavior summary is, (No need to write down, this is for understanding)

   User should be able to search for the availability of domestic and international flights for booking.
   Minimum departure date should be the current date, return date should be always greater than the departure date. Flight for maximum of 14 passengers can be searched and no of infants cannot be more than the #of adults.

   How do we understand the conditions that govern the behavior of the feature?

   If we can extract the conditions from the descriptive behavior, then identify that. Another approach is to prescribe the descriptive behavior (break down into steps) and for each step identify the success/failure factors and then combine those deciding factors to get the behavioral conditions. This approach is called 'Descriptive-prescriptive approach'

   Write down the descriptive behavior of search flight procedure as a series of steps,
   1. Select the travel type
   2. Select the trip type
   3. Select the source
   4. Select the destination
   5. Select departure date
   6. ------

   For each step, identify the condition that determines the success and failure (valid/invalid) of the step.

   1. Select the travel type *(Domestic, International)*
   2. Select the trip type *(One way, Round trip, Multicity/stopover)*
   3. Select the source *(one from the given list)*
   4. Select the destination *(one from the given list, other than source. This selection depends on the source)*
   5. Select departure date *(should be greater than >= today)*
   6. ----------

   The combination of values we provide for each step, decides the behavior of the search flight feature. Now combine and identify the conditions that govern the behavior of 'search flight" feature and let us represent it in a Decision Table.

Conditions that govern the behavior	
1	Travel Type
2	Trip type
3	Departure place
4	Destination place
5	Departure date
6	Return date
7	# of passengers
8	Class of travel

2. *Identify the possible values of each of these conditions*

Conditions	Possible values
Travel Type	Domestic(Dom), International(Int)
Trip type	One way(OW), round trip(RT), multicity/stop over(MS)
Departure place	Place from the given list(Valid), No selected any source(NS)
Destination place	Different from source(Valid), same as source(SS), not selected(NS)
Departure date	Today <= Departure date <= allowed date (Valid), less than today(LET), date after the allowed date of advance booking(GAD)
Return date	>= departure date(Valid), date after the allowed date of advance booking (GAD).
#of passengers	1-14 (valid), #of adults < # of infants (invalid)
Class of travel	Economy(E), Business(B), Premium Economy (PE)

Why do we identify the possible values and how do we know the conditions and possible values are complete and correct?

As we know, the behavior of an entity is nothing but a combination of conditions. If all the conditions are satisfy, we say it is the intended flow and violation of any condition is deemed as the negative flow. To know whether the identified conditions and its corresponding possible values are complete, we need to discuss with the concerned people if the specification is not clearly written. The same specification what we used to derive condition might be used by the developers to implement the same. Any missing of condition/possible value at this stage may lead to missing one flow/scenario.

3. *Combine the possible values of each of these conditions to identify various behaviors/scenarios.*

Input condition	**Possible values**	**Condition combinations/ Rules/Scenarios**						
		R1	R2	R3	R4			Rx
Travel Type	Dom, Int, NS	Dom	Dom	Int	Dom	---	---	Dom
Trip type	OW, RT, MS	OW	RT	MS	OW	--	--	OW
Travel source	Valid, NS	Valid	Valid	NS	Valid			Valid
Travel destination	Valid, SS, NS	Valid	Valid	X	NS			Valid
Departure date	Valid,LET,GAD	Valid	Valid	X	X			Valid
Return date	Valid, GAD	Valid	Valid	X	X			Valid
#of passengers	Valid, invalid	Valid	Valid	X	X			invalid
Class of travel	E,B,PE	E	B	X	X			PE
Expected Outcomes								
O1	Result displayed as based on the search criteria	x	x					
O2	Cannot search without source			X				
O3	Cannot search without destination				X			
O4	Destination cannot be same as source							
O5	Departure date cannot exceed allowed date of advance booking							
O6	Return date cannot be less that departure date							
O7	# of passengers cannot be more than 14							
O8	#of infants cannot exceed # of adults							x

The above table basically represents the various conditions that govern the behavior of the entity, the combination of the conditions based on the possible values of each condition and the expected outcome from each such combination. Each Rule in the DT will be scenarios. Based on the context of the application/Feature under test, some combinations may not be relevant and can be eliminated.

Should we need to consider all the combinations of all conditions?

Based on the context of the application, we may need to ignore considering some combinations. In this case if the departure date is wrong, we may not need to combine with other conditions like #of passengers etc as anyway it is an invalid combination.

4. *Convert each scenario into descriptive format, to a human understandable format.*

TSID	Scenario description	Expected Output
TS1 (R1)	Ensure that the domestic flights availability will be displayed as per the search criteria	O1
TS2 (R2)	Ensure that the International flights availability will be displayed as per the search criteria	O1
TS3 (R3)	Ensure that flights search cannot be done without providing the source of travel.	O2
-----	----------	--------
-----	----------	----------
TSx(Rx)	Ensure that the flight search cannot be done if the #of infants provided greater than the #of adults.	O8

5. *For each scenario, generate the test cases by providing actual test data.*

TS1 : Ensure that the domestic flights availability will be displayed as per the search criteria
Data table

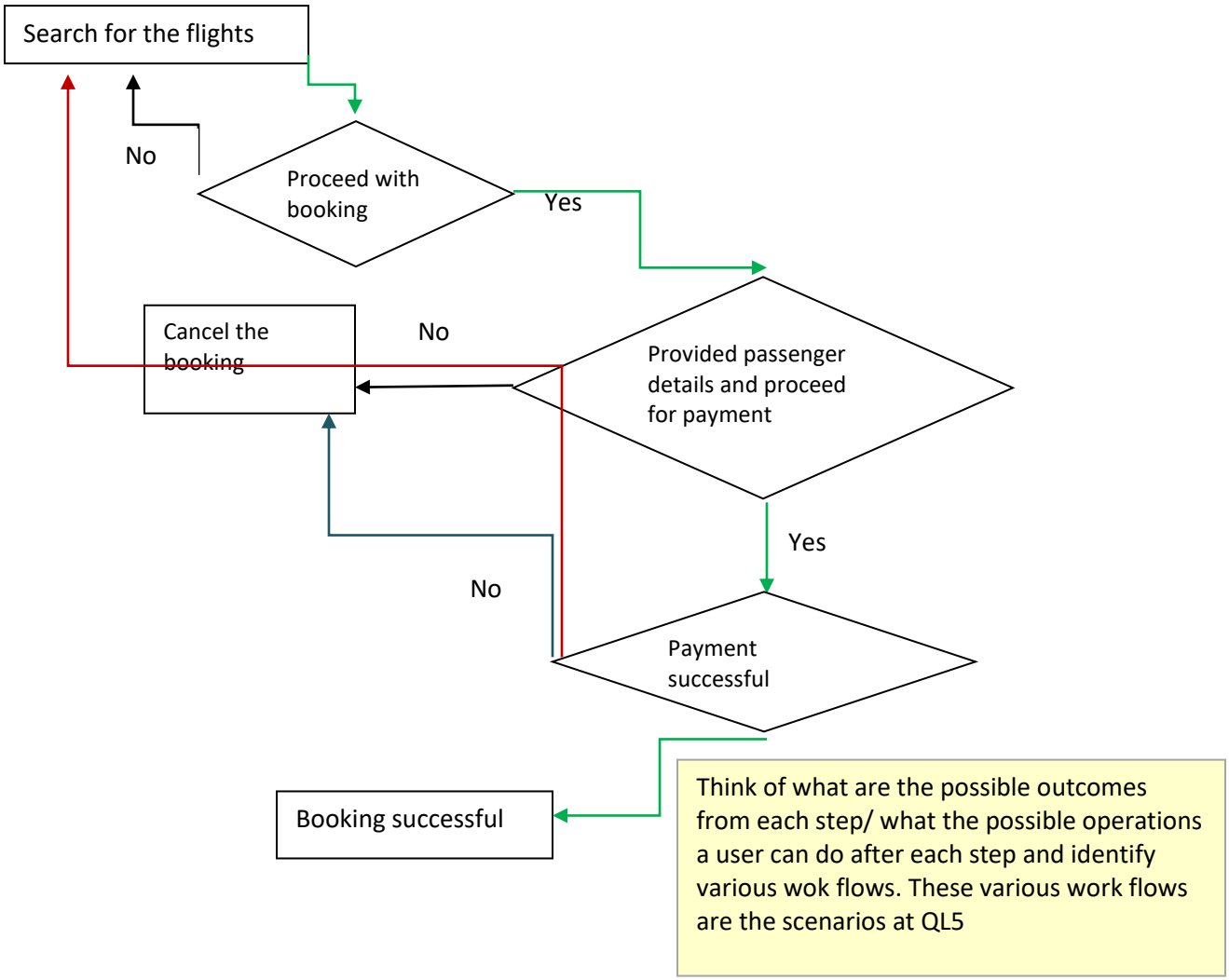
TCID	*Travel Type*	*Trip Type*	*Source*	*Destination*	*Dep date*	*# passengers*	*Travel class*	*Exp Result*
TC1	Domestic	OneWay	Chennai	Bangalore	Today	5	Economy	All the domestic flights available from Chennai to Bangalore should be displayed for today.
TC2	Domestic	OneWay	Chennai	Bangalore	One year ahead	3	Economy	All the domestic flights available from Chennai to Bangalore should be displayed for 360 day in advance.
-	-	-	-	-	-	-		

QL5 – Flow correctness

In quality level 5, we ensure that the individual feature we have tested will be working fine in the business flow. We have seen how the test design of "search flight" features. Ultimately the user wants to book a flight from one place to another after searching the flight availability.
So what we need to understand at this level, is the various flows involved by each feature under test.
In this case,

- The user may search for the flight availability; he will cancel the result and search again with the modified date/class of travel/ without changing other details.
- The user may proceed with the ticket booking and cancel at the time of payment.
- The user may book the ticket and cancel some of the passenger later.
- The user may book round trip ticket and cancel the return trip after reaching the destination and opt for another date of travel.
- ----------
- ---------

In QL5, we need to list down the various features tested individually and see how the features are part of business flow and what the various ways they will be used are. The best way to get an idea is, to represent the flow in a flow chart (or any flow representation) and then think of various options at each step.



.