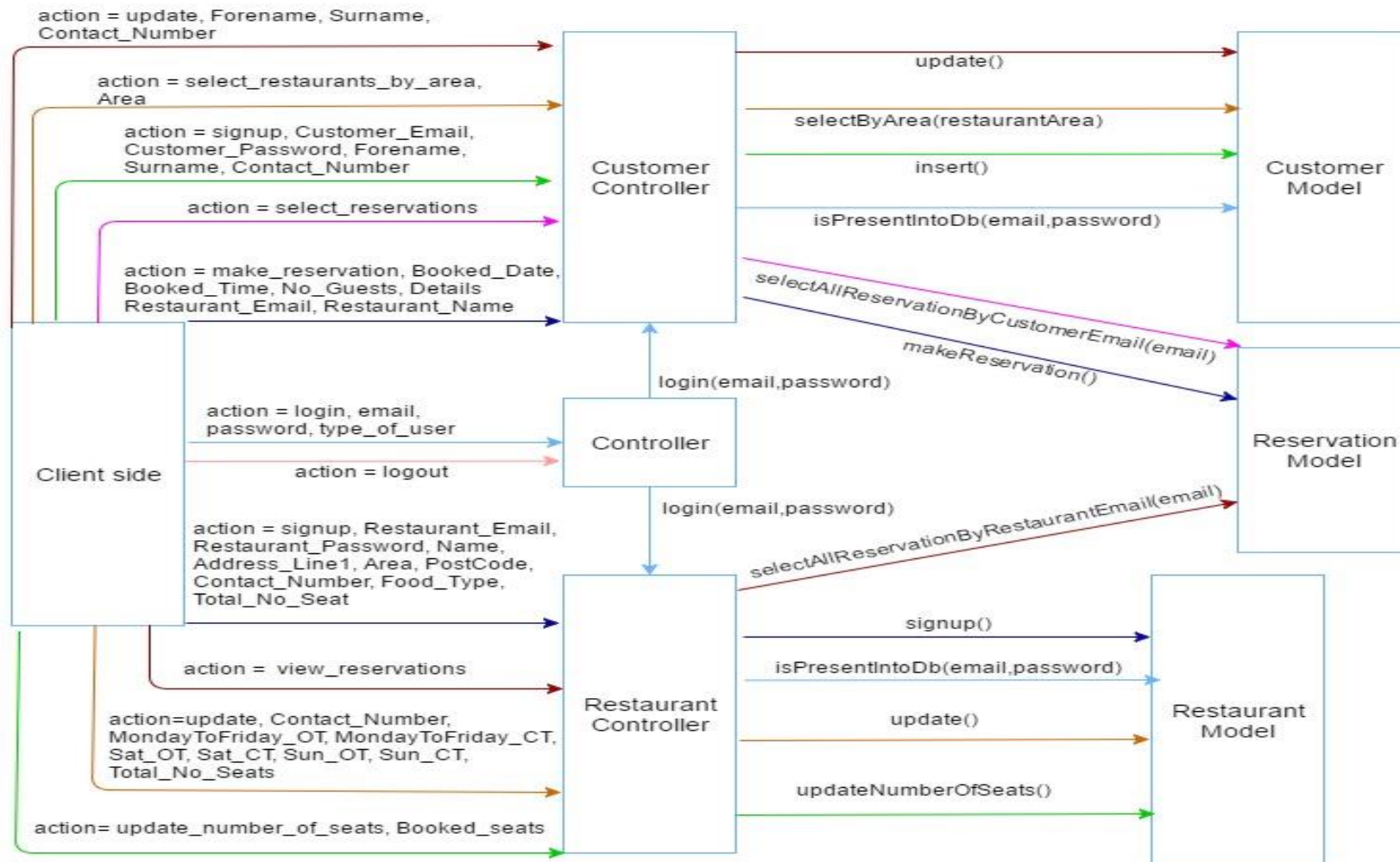


Testing Documentation

Testing Strategies

Unit testing: The test strategy adopted during unit testing is equivalence partitioning, specifically in the majority of cases, the tests using this strategy are tests of the controller component, with the task of receiving data from the browser (client side) and checking the correctness in terms of the intervals and validation. An example is when a Restaurant wants to update the opening and closing times, in this case, opening time must be before the closing time. In a similar way, to update the number of booked seats, the restaurant must send an integer value that is greater than 0. The tests ensure that these kind of conditions are checked in the program.

Component testing: The components in the system are the client side, the controller and the model. The client send parameters to the Controllers which, if necessary, calls the methods of the model to query the database or change/insert new data. Below is the diagram of the component interfaces.



System testing: The system was stress tested using the tool “Siege”. For each test the system simulates a number of users concurrently accessing a specific page over a set time frames, and returns the total number of transactions or interactions between the user and webpage. The number of failed transactions is also measured.

Testing Process

JUnit testing has been used to test the units. Below are the test cases where all of the classes of the system have been tested. Helium has also been used to test the web pages. Outputs can have these two meanings: True= Value(s) is accepted and False=Value(s) not accepted

Controller tests

Test	Input	Description	Expected Output	Result
Restaurant controller test cases				
1	Email: luc@luc.com Password: Lucinda1	Login: The email is correct as it exists in the database with the corresponding password Lucinda1. This tests that the login process works with the database.	True	True
2	Email: na@nta.com Password: Fraco1	Login: The email validation should fail because the email does not exist in the database. Therefore, the user should not be allowed to access the system.	False	False
3	Email: Randomly generated Password: Esco3553 Restaurant name: Jamie Rest Address line: example street Area: Southsea Postcode: PO1 2DJ Contact number: 04444028592 Food type: Indian Number of seats: 20	Restaurant sign up: This tests the process of registering a restaurant. The data entered must update the database with the corresponding values to the fields entered. All the data inserted into the database is correct, in particular the email which is a valid email and the number of seats is greater than 0 which is a valid entry.	True	True
4	Same data entered from test 3 with 1 change- Number of seats: 0	Restaurant sign up: This tests the validation of the number of seats which must be greater than 0. All data inserted is correct except the number of seats that is 0 which is an invalid value for that field.	False	False
5	Same data entered from test 3 with 1 change- Number of seats: -5	Restaurant sign up: All data inserted is correct except the number of seats that is -5 which should be greater than 0. This tests that the validation should not accept this value.	False	False
6	Same data entered from test 3 with 1 change- Email: aoeugn	Restaurant sign up: This tests the validation of the email field which must include at least 1 capital letter and number. The entered email is not a valid email and should be rejected.	False	False
7	Monday to friday opening time: 10:30 Monday to friday closing time: 12:30 Saturday opening time: 11:30 Saturday closing time: 12:30 Sunday opening time: 11:30 Sunday closing time: 15:30 Number of seats: 2 ,Contact number: 2535902503	Restaurant update: This tests whether the data that is inserted to update the restaurant account works and also updates the database with the correct correspondence to the restaurant logged in. Data to be updated is all valid. The opening times are always before the closing times and the total number of seats of the restaurant is positive, therefore the data should be accepted.	True	True

8	Same data entered from test 7 with 1 change- Monday to friday opening time: 12:31	Restaurant update: Tests the validation of opening and closing times. Data to be updated is invalid. Specifically the monday to friday opening time because the close time comes beforehand.	False	False
9	Same data entered from test 7 with 1 change- Saturday opening time: 12:21	Restaurant update: Tests the validation of opening and closing times. Data to be updated is incorrect. Specifically the saturday opening time which comes after the saturday closing time.	False	False
10	Same data entered from test 7 with 1 change- Sunday opening time: 15:21	Restaurant update: Tests the validation of opening and closing times. Data to be updated is incorrect. Specifically the sunday opening time which comes after sunday closing time.	False	False
11	Same data entered from test 7 with 1 change- Number of seats: 0	Restaurant update: Tests the validation of number of seats. Data to be updated is invalid. Specifically the total number of seats of the restaurant which is 0. The field must be greater than 0.	False	False
12	Same data entered from test 7 with 1 change- Number of seats: -5	Restaurant update: Tests the validation of number of seats. Data to be updated is incorrect. Specifically the total number of seats of the restaurant which is negative and should be a positive value.	False	False
13	Booked seats: 20	Update number of booked seats: The booked seats value is positive which is valid. This tests the acceptance of a correct value for booked seats.	True	True
14	Booked seats: 0	Update number of booked seats: The booked seats number is 0 and therefore is incorrect. Tests the validation for the booked seats field.	False	False
15	Booked seats: -15	Update number of booked seats: The booked seats number is negative and therefore is invalid. This tests that the validation for booked seats works.	False	False
Customer controller test cases				
16	Email: nota@nota.com Password: Francesco1	Customer login: This tests that the login process works with the database. Both password and email exist in the database.	True	True
17	Email: na@nta.com Password: Fraco1	Customer login: This tests that the data is checked against records within the database. The password and email do not exist into the database.	False	False
18	Email: Randomly created Password: Osuhr1 Forename: Julia Surname: sug Contact number: 0934567890	Signup: This tests that data entered when creating an account, updates the records in the database and allows the user access to the system. All data is correct, therefore this test should work.	True	True
19	Same data entered from test 18 with 1 change- Email: gsioun	Signup: This tests the validation of emails where an '@' must be present to make it valid. The email entered is not valid.	False	False
20	Restaurant email: luc@luc.com Customer email: nota@nota.com Number of guests: 4 Reservation date: Current date in milliseconds + 60 000 000 milliseconds Name: Mark	Make reservation: This tests that data entered when making a reservation updates the records in the database. The data inserted is correct so should work.	True	True

	Details: <i>details example string</i>			
21	Same data entered from test 20 with 1 change- Number of guests: 0	Make reservation: Data inserted is invalid, the number of guests must be greater than 0. This tests the validation of the guest number.	False	False
22	Same data entered from test 20 with 1 change- Number of guests: -5	Make reservation: Data inserted is incorrect, the number of guests in the reservation must be positive. Tests the number of guests validation.	False	False
23	Same data entered from test 20 with 1 change- Reservation date Current date in milliseconds - 60 000 000 milliseconds	Make reservation: Data inserted is incorrect. Specifically, the reservation date which is in the past. This tests the reservation date validation.	False	False

Model tests

The responsibility of model classes is to store data coming from the controller into the database, these tests check that no exception is thrown while inserting data into the database.

Test	Input	Description	Expected Output	Result
Customer model tests				
24	Email: <i>Randomly generated</i> Password: <i>Francesco1</i> Forename: <i>Mirko</i> Surname: <i>Vucinic</i> Contact number: <i>081924535</i>	Insert customer: Tests whether the data from the controller is stored in the database with no exception thrown.	No exception thrown	No exception thrown, moreover the insertion has been checked manually into the database
25	Forename: <i>Lucinda</i> Surname: <i>Benny</i> Contact number: <i>089999999</i> Email: <i>nota@nota.com</i>	Update customer: Tests whether the data from the controller is stored in the database with no exception thrown.	No exception thrown	No exception thrown, moreover the insertion has been checked manually into the database
26	Email: <i>nota@nota.com</i> Password: <i>Francesco1</i>	Check if an account is in the database: in this case, input data is in the database. Tests that the details are valid.	True	True
27	Email: <i>luc@mail.com</i> Password: <i>luc</i>	Check if an account is in the database: in this case, input data is not in the database. Tests whether the database is checked for login values before allowing users to proceed.	False	False
Reservation model tests				
28	Restaurant Email: <i>luc@luc.com</i> Customer Email: <i>nota@nota.com</i> Restaurant name: <i>Lucinda restaurant</i> Number of guests: <i>5</i> Booked date: <i>Current date</i> Details: <i>Details string reservation example</i>	Insert reservation: Tests whether the data from the controller is stored in the database with no exception thrown.	No exception thrown	No exception thrown

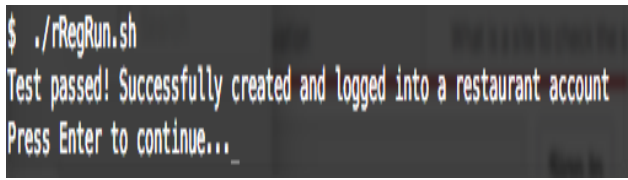
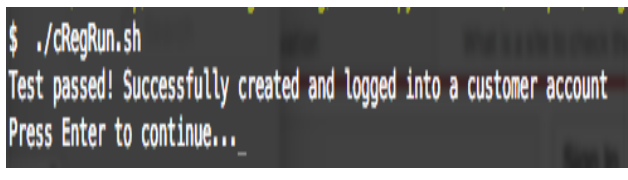
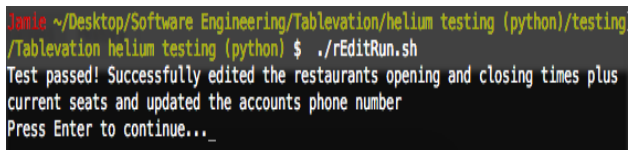
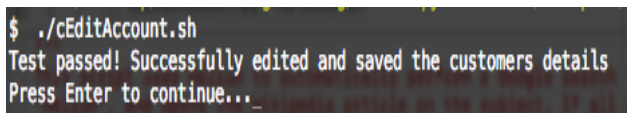
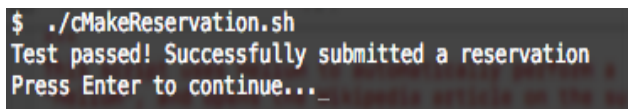
29	Restaurant Email: <i>luc@luc.com</i> Booked date: <i>Current date</i>	Already booked: Before testing the already booked method, a reservation with the current date as booked date is created and then, it is checked if the function already booked with date current date returns true and therefore works correctly.	true	true
30	Restaurant Email: <i>luc@luc.com</i> Booked date: <i>Current date + 6 000 000</i>	Already booked: The date and time are available and therefore the method already booked returns false. Tests the validation of booking dates to already booked dates.	false	false
31	Customer Email: <i>nota@nota.com</i>	Select reservation by customer email: The method returns a list of reservations and for all of them it's checked if the customer email is equal to the one used when the method is called.	Email used in the method = customer email of all elements in the list returned by the method	Email used in the method = customer email of all elements in the list returned by the method
32	Restaurant Email: <i>luc@luc.com</i>	Select reservation by restaurant email: The method returns a list of reservations and for all of them is checked if the restaurant email is equal to the one used when the method is called.	Email used in the method = restaurant email of all elements in the list returned by the method	Email used in the method = restaurant email of all elements in the list returned by the method
Restaurant model tests				
33	Email: <i>Randomly generated</i> Password: <i>Jamie1</i> Restaurant name: <i>Jamie Rest</i> Address line: <i>Jamie street</i> Area: <i>Southsea</i> City: <i>Portsmouth</i> County: <i>hampshire</i> Postcode: <i>PO1 2DJ</i> Contact number: <i>04444028592</i> Booked seats: <i>0</i> Food type: <i>Italian</i> Number of seats: <i>20</i> Monday to friday opening time: <i>Current time</i> Monday to friday opening time: <i>Current time in millis + 6 000 000 millis</i> Saturday opening time: <i>Current time</i> Saturday opening time: <i>Current time in millis + 6 000 000 millis</i> Sunday opening time: <i>Current time</i> Sunday opening time: <i>Current time in millis + 6 000 000 millis</i>	Insert restaurant: Tests whether the data from the controller is stored in the database with no exception thrown.	No exception thrown	No exception thrown, moreover the insertion has been checked manually into the database
34	Contact number: <i>0986545678</i> Number of seats: <i>25</i> Monday to Friday opening time: <i>Current time</i> Monday to Friday opening time: <i>Current time in millis + 6 000 000 millis</i>	Update restaurant: Tests whether the data from the controller is stored in the database with no exception thrown.	No exception thrown	No exception thrown. All the updated fields have been checked manually into the database

	Saturday opening time: <i>Current time</i> Saturday opening time: <i>Current time in millis + 5 000 000 millis</i> Sunday opening time: <i>Current time</i> Sunday opening time: <i>Current time in millis + 6 000 000 millis</i>			
35	Booked seats: 10	Update booked seats: Tests whether the data from the controller is stored in the database with no exception thrown.	No exception thrown	No exception thrown. All the updated field has been checked manually into the database
36	Email: <i>luc@luc.com</i> Password: <i>Lucinda1</i>	Check if an account is in the database: in this case input data is in the database. This tests whether the data enter corresponds to fields in the database.	True	True
37	Email: <i>kac@kaf.com</i> Password: <i>verrb124</i>	Check if an account is into the database: in this case input data is not into the database. This tests whether the data enter corresponds to fields in the database.	False	False
38	Area: <i>Southsea</i>	Select restaurant by area: The input area exists in the database and there are some restaurants within this area. This tests the data within the database for the specified area entered.	No exception thrown, expected size of the list of restaurant returned is greater than 0	No exception thrown, size is greater than 0
39	Area: <i>random string area</i>	Select restaurant by area: The input area does not exists in the database. This tests the data within the database for the specified area entered.	No exception thrown, expected size of the list of restaurant returned is 0	No exception thrown, size is 0

Website tests

All tests seen below in the table plus screenshots of the results are uploaded to our GitHub account running in python.

Description	Result	Problems
Owner logging into their account. This test will input the login credentials then click on the option of which account to log into (owner) and finally click login to show the owners homepage.	<pre>\$./oLoginRun.sh Test passed! Successfully logged into the restaurant account Press Enter to continue...</pre>	When originally testing this command we found that simply writing 'click("Login")' in the python file it would fail as it was a button rather than a link or radio button we needed to select so we had to write "click(button("Login"))" to correct the error.
Customer logging into their account. This test will input the login credentials then click on the option of which account to log into (customer) and finally click login to show the customer's homepage.	<pre>\$./cLoginRun.sh Test passed! Successfully logged into the customer account Press Enter to continue...</pre>	As this test is very similar to the owners login test we already knew where we went wrong before so this time there was no problems faced.

<p>Owner create an account and login. This test showed to steps and inputs that was needed to create an owner account meeting all the validation e.g. a password must need at least 6 characters one capital letter plus one numerical character.</p>		<p>This test caused a few problems to run as this time we need to make the test select the dropdown list then select from that list the area to which the restaurant is located plus type of food that the owner's restaurant produces. Originally we began to write out "click("Please select an area"), write("southsea")" but that cause the test to come back with errors so by looking at the API documentation on heliumhq we found out that we actually needed to write "click(button("Please select an area")) then click("southsea")" as the drop down list acts as a button then you.</p>
<p>Customer create an account and login. This test showed to steps and inputs that was needed to create a customer account meeting all the validation e.g. a password must need at least 6 characters one capital letter plus one numerical character. This test is also very similar to the previous test but this time we did not have to select from a drop down list.</p>		<p>We found no problems with this test.</p>
<p>This test shows the owner update their details such as their current email address and phone number. It also shows us how they would filling in the sections about their opening and closing times on certain days of the week.</p>		<p>During this test we did not face any problems as we had dealt this all types of input before in previous tests.</p>
<p>This test shows the customer editing their account details such as phone number, email address and their name.</p>		<p>No problems or errors were found here.</p>
<p>The last test we did was to check how the customer would input their reservation to the site.</p>		<p>No problems or errors were found here.</p>

System Testing	Test 1: 50 Users 20 Seconds		Test 2: 150 Users 20 Seconds		Test 3: 200 Users 20 seconds	
Requested Page / Resource	Total Transactions	Failed Transactions	Total Transactions	Failed Transactions	Total Transactions	Failed Transactions
index.jsp	1828	0	4454	45	4526	425
login.html	1896	0	5476	28	6982	81
createCustomerAccount. html	1829	0	5629	31	7079	73
createRestaurantAccount.html	1927	0	5647	40	6826	109
Login Functionality	1956	0	4981	24	5832	213
Select Restaurant Functionality	1816	0	3814	131	7369	51
Logout Functionality	1790	0	5500	44	7258	50
Select Reservation	1958	0	5505	33	6816	65
Update Number of Seats	1866	0	5464	32	6663	101

The results show that at around 150 concurrent users, most pages in the site start to experience a low frequency of failures, which usually only increases a small amount when the number of users is increased to 200. However, there are outliers such as login functionality where more computational power is needed as the database is being accessed, therefore, there are more failures at higher counts of concurrent users.

Usability Evaluation

Evaluation Goals and User Interface Aspects to Evaluate:

Specify User Interface Requirements, Evaluate Design features of Tablevation, Identify problems with the systems usability and Improve User Interface for better usability are the goals this evaluation should result in. The interactivity, layout, readability, accessibility, display characteristics and aesthetic aspects of Tablevation will be tested. These aspects will ensure that the buttons and dropdowns work, the content is understandable, the system gives users ease of use and to make sure the design is responsive.

Target Users:

The demographic of Tablevation covers people who are most likely to reserve tables at restaurants which is considered around the age of 18 to 60 years old. It is clear that each individual will have a varied experience with technology. The varied experience will be split into 3 categories consisting of: Basic Knowledge, Good Knowledge and High level of knowledge for those working in the industry of ICT.

Design of Experiment:

6 Participants will be used in this evaluation. Effectiveness and efficiency will be tested during each task, after completing all of the tasks, user satisfaction will be tested. The users will be observed in a controlled environment, the University Cafe. This location is due to the fact that users will be able to use the system from wherever as long as there is an internet connection. The system will be set up on a laptop which will be used by the user testing Tablevation. The users will do all of the tasks stated above and each task will be individually recorded. The data will be recorded during evaluation as a total tally while a timer is used to record the time it takes for tasks to be completed. The users will then be asked to fill in their answers to the SUS scale and the results will be calculated from there. User comments about the system will also be encouraged to isolate areas that need improving.

Usability Metrics

Metric	Test	Criteria
Effectiveness	This will be a quantitative metric which will consist of the number of errors recovered and the functions learnt by the user.	Errors Recovered = Errors Errors<5
Efficiency	The user will be timed during the execution of each task. The quantitative times will then be compared within each of the 3 ICT skill categories.	Within each category: Difference in results <=2 minutes
User Satisfaction	The System Usability Scale (SUS) will be used to measure the qualitative metric, user satisfaction of the system ('System Usability', 2013). The users will also be asked for their comments on improvements to understand the areas to refine.	>=68

Usability Data E=Error R=Recovery

ICT Skill Category		Basic Knowledge		Good Knowledge		High level	
Metric	Task	User A	User B	User C	User D	User E	User F
Effectiveness	1	1E 1R	0	1E 1R	1E 1R	1E 1R	1E 1R
Efficiency		49 seconds	59 seconds	42 seconds	40 seconds	33 seconds	31 seconds
Effectiveness	2	0	1E 1R	0	1E 1R	0	0
Efficiency		38 seconds	42 seconds	25 seconds	28 seconds	24 seconds	18 seconds
Effectiveness	3	2E 1R	3E 3R	1E 1R	1E 1R	0	0
Efficiency		59 second	1 min 4 sec	44 seconds	40 seconds	26 seconds	21 seconds
Effectiveness	4	0	0	0	0	0	0
Efficiency		12 seconds	11 seconds	8 seconds	6 seconds	6 seconds	9 seconds
Effectiveness	5	0	0	0	0	0	0
Efficiency		28 seconds	35 seconds	20 seconds	17 seconds	15 seconds	14 seconds
Effectiveness	6	0	0	0	0	0	0
Efficiency		5 seconds	7 seconds	6 seconds	4 seconds	5 seconds	4 seconds
Effectiveness	7	0	1E 1R	0	0	0	0
Efficiency		1min 19sec	1 min 27sec	52 seconds	46 seconds	41 seconds	37 seconds
Effectiveness	8	0	0	0	0	0	0
Efficiency		10 seconds	8 seconds	8 seconds	9 seconds	6 seconds	7 seconds
Effectiveness	9	2E 1R	0	0	1E 1R	1E 1R	0
Efficiency		30 seconds	21 seconds	15 seconds	18 seconds	9 seconds	13 seconds
Effectiveness	10	1E 1R	1E 1R	2E 2R	0	0	1E 1R
Efficiency		50 seconds	57 seconds	1 min 2 sec	40 seconds	28 seconds	33 seconds
Satisfaction	All	68	70	58	63	70	58

Improvements Suggested:

- Having to scroll down on many pages to get to the information required made it harder to figure out if you were on the right page.
- Users should be told about password format requirements before having to enter data.
- Selecting times via the make booking page requires user to click and type before the drop down appears with times available. Make the dropdown appear with the click.
- Make it clear which information is required when editing the restaurant account.
- Wasn't clear what was meant by customer / owner (on the login part)
- The limits for total number of seats isn't clear, should be stated before having to enter data.

Analysis of Results and Interface Improvements:

During the investigation it was clear that task 3 held the most errors, although almost all errors were recovered, there seemed to be a reoccurring problem. This was down to the 'Time' field of the reservation, the dropdown wouldn't appear until the user started typing. This problem will be fixed by changing the dropdown mechanism used. Most errors observed were the fact that users weren't entering valid data or filling in mandatory fields, this will be fixed by including more information on formats and limits on data that needs entering as well as the difference between users and owners. All errors made were under the set criteria of 5 which was a good outcome. The times recorded were also within the criteria but not just within their own ICT level category, but throughout all users testing the system, this was a good sign that the system was not too complex for the users. The SUS scale had 3 users who fit the criteria for satisfaction, whereas 3 did not find the system as usable. The lowest SUS was 10 away from being classified as usable, the questions analysed showed that question 2 and 3 brought that mark down heavily. Both users that resulted in the lowest SUS scores agreed that the system was unnecessarily complex and not easy to use. This could have been down to the comment with having to scroll to see the information they would like. This comment was made multiple times about the images presented below the navigation on many of the pages. To improve the user interface, this image will be removed to prevent the unnecessary need of users having to scroll.

References

System Usability Scale (SUS). (2013) . Retrieved from the usability website:

<https://www.usability.gov/how-to-and-tools/resources/templates/system-usability-scale-sus.html>

Appendix

Appendix 1: SUS Results Calculation

Question	Rating 1-5 (R)						Calculation
	User A	User B	User C	User D	User E	User F	
Q1	2	3	3	4	4	3	R-1
Q2	3	3	5	3	4	5	5-R
Q3	3	4	5	4	4	2	R-1
Q4	3	2	3	2	2	2	5-R
Q5	5	5	4	4	3	3	R-1
Q6	2	2	3	4	3	3	5-R
Q7	4	4	4	4	4	4	R-1
Q8	2	2	3	3	2	2	5-R
Q9	4	3	3	3	5	4	R-1
Q10	1	2	2	2	1	1	5-R
Total	27	28	23	25	28	23	X 2.5
Result	68	70	58	63	70	58	N/A

Declarations:

- I, Francesco David Nota, declare my contribution to the Unit and Component Testing of this testing document.

Francesco David Nota

A handwritten signature in black ink that reads "Francesco David Nota". The signature is written in a cursive style with a large, stylized 'F' at the beginning.

- I, Lucinda Ashdown, declare my contribution to the Usability Evaluation of this testing document.

Lucinda Ashdown

A handwritten signature in black ink that reads "Ashdown". The signature is written in a cursive style with a large, stylized 'L' at the beginning.

- I, Jamie Toloui, declare my contribution to the Web Testing of this testing document.

Jamie Toloui

A handwritten signature in black ink that reads "Toloui". The signature is written in a cursive style with a large, stylized 'T' at the beginning.

- I, Benjamin Harris, declare my contribution to the System Testing of this testing document.

Benjamin Harris

A handwritten signature in black ink that reads "BHarris". The signature is written in a cursive style with a large, stylized 'B' at the beginning.

- I, Mitchel Tendai Gwaze, declare my contribution to the Usability Evaluation of this testing document.

Mitchel Tendai Gwaze

A handwritten signature in black ink that reads "Mitchel". The signature is written in a cursive style with a large, stylized 'M' at the beginning.