

8 – Testing

Contents

- 8.1 Test Strategy 2
- 8.2 Objective List 3
- 8.3 Testing & Evidence..... 4
 - 8.31 Alpha Testing 4
 - 8.32 Beta Testing..... 45
- 8.4 Objective Summary 46

8.1 Test Strategy

To test my system effectively I will carry out two stages of tests to extensively test the quality of my system and whether it meets the objects that were set out. **Firstly, alpha testing** will be used in hand with **white box testing**. As I designed and created the program, I will be running these tests by objective, testing features on each screen under various conditions. These tests will include **functionality, navigation, performance, logic and validation tests** as well as brief comments on file handling techniques. This will help me determine whether my system works as intended in **both typical and unusual conditions**, but also responding to any incorrect inputs from the user. As I will be using white box testing, I will be running the program while having the **source code open beside it**, to be able to see the methods being run during most process, in the event of any unusual behaviour from the system. If this is the case, I will be carefully repeating tests to identify the reasons behind any issues and **discussing potential refinements** of the system and **implementing these refinements if** appropriate. However, if the problem is a larger scale issue or revolves around an ineffective approach, I will not be correcting these problems instead they will not pass my tests and would be ideas for a larger scale version of my system. I will also make comment on the effectiveness of each screen overall regardless of its functionality and will be **considering factors such as user-friendliness, efficiency of processes and whether the human-computer-interaction is intuitive**, to make a judgement on future refinements that could potentially be implemented.

For the second stage of my testing, **I will be running beta tests** using a mixture of user who both competent users (**A-level Computer Science Students**) and **expected user (those who would realistically use the system)** using these two sets of users I would be able to obtain a **wide range of feedback**. As computer science students would be able to **find potential errors or limitations** in my system as they would look for a way to make my system fail, to identify its issues. The general user would be able to give me **feedback in terms of the systems efficiency and user friendliness, and whether the design and implementation is suitable for them in everyday use**. When conducting the beta test, I will let my chosen users use the system, **with no real guidance** other than information they will require such **as login data and the basic purpose of the system**. I will then let them use the system independently and will ask them to fill in answers to a couple of questions I have created as well as allowing them to **make general comments on the system overall**. As I have tested the systems functionality during the alpha test this beta test is designed to gain an in-depth view on the human-computer interaction. **This will also include black box testing** as the users will not have access to the source code of my system, **as a real user would**.

8.2 Objective List

Objective #	Objective Description
1	Security – verify staff logins
2	Read from a text file – staff logins
3	Security - Verify customer logins
4	Read from a text file – customer logins
5	Read from a text file – booking file
6	View bookings – display array data into a table
7	Sort/Filter bookings by date
8	Sort/Filter bookings by name
9	Add bookings to the list
10	Write to a file – booking database
11	Write to a file – customer info
12	Write to a file – payments database
13	Calculations to decide price
14	Delete bookings from the database
15	Search for a booking by name
16	Search for a booking by date
17	Search for a booking by booking ID

8.3 Testing & Evidence

8.31 Alpha Testing

Objective # & Description

1&2: Security – verify staff logins and read from staff login text file

Discussion of Expected Outcomes

For this objective, I will be running a functionality test, navigation test, and various validation tests. The functionality test will allow me to see if the login works, which will also display the effectiveness of the navigation. The validation tests are to see how the system will handle different inputs and whether it is equipped to deal with any issues in an effective way.

- **Functionality (and Navigation) test** - I will initially **enter a correct username and password** which I expect that, once I press the login button, the data will be compared to each index within the login array to see if both username and password match the record of that index. **If correct, the user is allowed access to the rest of the system** by switching to the menu screen, a screen that cannot be access via a button.
- **Validation (Incorrect username and password)** – secondly, I will enter both an incorrect username and password. As these **inputs would not match any record** stored in the text file the output from the system **should be an error message**, displayed via pop-up.
- **Validation (Incorrect username and correct password Or vice versa)** – thirdly I will use semi-correct data by entering either a correct password, or username which do not match the same record, **but not both correct**. This validation will test whether the system is able **to recognise correct data in an incorrect form** and be able to inform the user that the login data **is partially correct**.
- **Validation (no data)** – for the final validation test of the login I will enter no login data onto the screen and will try to login. This should **prompt another error message**, this time advising the user to enter data to login.
- **Functionality (stress testing)** – for this test I will be performing a stress test on the system by continuously **entering incorrect login data and trying to login**. I will be aggressively pressing the button and entering a variety on erroneous data to see whether the **login will remain functional**.
- **Functionality (Read from staff login file)** - this test will be to show that the system is able **to effectively read from the staff login** file and store the data in the login array for the login to able to work. This method will **produce outputs onto the command line** rather than to the user as this process occurs as the system starts up, allowing for the user to use the system straight away.

Tests Planned

Test ID	Test Type	Purpose	Actual Data Used	Pass / Fail
1A	Functional	To test if the login works	Shak: Shakz_01	P
1B	Validation	To prevent incorrect login data	shak: Shakz_01	P
1C	Validation	To detect partially correct data	Shak: Qwerty	F
1D	Validation	To detect and prevent null data input	Null	P
1E	Navigation	To see if correct screens are shown	Button pressed	P
1F	Functional	To see if login remains functional after stress test	-	P
1G	Functional	To show the login file is read from	-	P

Test Commentary and Refinements Needed

1A/1E – the test for a functional login when **using correct data was successful**. I entered a correct login record which was confirmed as a correct login by the system. The **screen then changed to the staff menu screen** and an output was displayed on the command line with a message saying I have logged on with correct username and password, but also the actual value of the login used. This **functioned effectively whilst also navigating to the menu page** with no other input required, **allowing the user to seamlessly carry on using the system with no delay**. However, the one slight issue is that the user must click the login button, which makes the system **slightly less intuitive**. Although **clicking the button manually is an effective** and reliable form of input, there could be other means of input implemented, **such as pressing a key on the user's keyboard, which would improve efficiency**.

1B – for this validation test I used the same login data as the previous test, yet I did not include the capital letters in either **username or password, therefore making the data incorrect**. Once I pressed the login button a **pop-up was displayed saying that both the username and password were incorrect**, and that I should try again. After clicking away from the error message, the screen remained the same, with the incorrect input still in the textboxes. **This validation worked successfully**, as the system was unable to recognise the data entered **therefore deciding it was an incorrect login**. This is also **quite intuitive** as the data stays in the text boxes, which can **allow the user to easily edit their input, in case that they made a simple error** when entering the data.

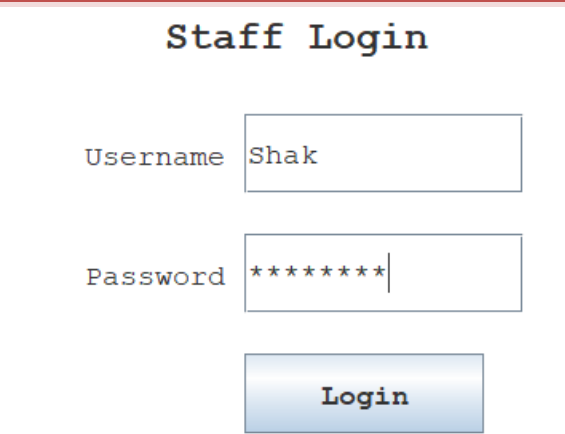
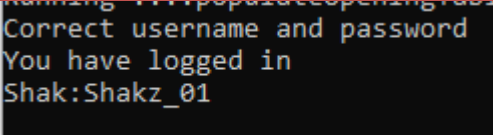
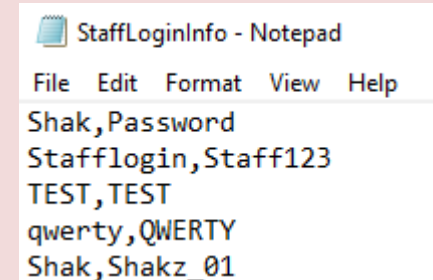
1C – This validation technique is used to **recognise partially correct login data**. The data I entered was a correct username, yet **it was paired with an incorrect password**. However, once I pressed login, the **output remained the same as the previous test**, an error message saying both username and password are wrong. Therefore, this form of **validation was not successful in** correctly identifying a partially incorrect login. As I am doing **white box testing, in hand with alpha testing**, I was able to see the exact methods being run when I pressed the login button. Whilst running the program again I **noticed that during an if statement, a Boolean variable that I had created to state whether the login was partially incorrect, was never being checked as a sperate Boolean variable, to state that the login was incorrect**, would be checked first, meeting the condition to display the error message that both username and password are incorrect. To correct this, **I decided to use 'nested if statements'** which if the login was incorrect, another if statement will be run to determine the type of incorrect input and **produce an output accordingly**.

1D – This validation test is a simple test as I ran the **login without entering any input data**. Once I pressed the login button, **an error message was displayed saying** that I should enter both username and password. This was **successful in identifying** no input and produced a suitable output tailored to the issue at hand.

1F – this test is **one which stress tests the login**, where I search for a way to make it **behave incorrectly**. To do this is entered a **range of incorrect data**, yet the main test was to **repeat these many times and see how the system responded and whether the system would remain functional** or if it would fail. The system was successful in coping with such aggressive inputs and remained functional each time the login was extensively used.

1G – this final test is to show that the **staff login file is read from**. Once I run the program, the file is automatically read from, splitting each line where a comma is found, to **split the record into each attribute and is then stored into a temporary string array**. Again, as I am using **white box testing**, I **could see the exact methods being run**. Each time a line from the file was read, the temporary array was passed into a **separate method as a parameter to be stored into an array** of login data. Although **there was no visual output**, in the form of a GUI pop-up, to show the methods in effect they displayed the data being read each time on to the command line.

Results

Test ID	Evidence
1A	<div></div> <div></div> <div></div>

1B

Username shak

Password *****

Login

Message



Incorrect username and password, try again

OK

Partially Incorrect
shak:Shakz_01

1C

Username Shak

Password *****

Message



Incorrect username and password, try again

OK

Incorrect
Shak:qwerty

Username Shak


Password ***

Message



One of the components of the login is incorrect, try again

OK

1D	<div data-bbox="422 241 981 488"> <div>Username <input type="text"/></div> <div>Password <input type="password"/></div> </div> <div data-bbox="443 488 1077 734"> <div>Message ✕</div> <div>  Please enter both username and password </div> <div>OK</div> </div> <div data-bbox="336 745 544 857"> <pre> Incorrect : </pre> </div>
1F	<div data-bbox="336 887 766 1525"> <pre> Incorrect : Incorrect : Incorrect fxdg:fxg Incorrect fxdg:fxg Incorrect fxdg:fxg Incorrect fxdg:fxg Incorrect fxdg:fxg Incorrect fxdg:fxg Incorrect 123456789:1234567 Incorrect 123456789:1234567 Incorrect 123456789:1234567 </pre> </div> <div data-bbox="766 887 1401 1525"></div>
1G	<div data-bbox="336 1525 970 1874"> <pre> Running reading from Staff Login File Running add To Staff Login List Shak,Password Running add To Staff Login List Stafflogin,Staff123 Running add To Staff Login List TEST,TEST Running add To Staff Login List qwerty,QWERTY Running add To Staff Login List Shak,Shakz_01 </pre> </div> <div data-bbox="970 1525 1401 1874"></div>

Objective # & Description

3&4: Security – verify customer logins and read from customer login text file

Discussion of Expected Outcomes

To test this objective, I will be testing functionality, navigation, validation. The functionality test will show that the login works effectively, as well as showing the navigation in effect too. The validation tests are to see how the system will handle both correct and incorrect inputs and whether it will remain functional during this.

- **Functionality (and Navigation) test** - I will initially **enter a correct username and password** which will be compared to the data within the login array to see if both username and password match the record stored in that index. **If correct, the customer is allowed access to the rest of the system** by switching to the menu screen.
- **Validation (Incorrect username and password)** –for this test I will enter an incorrect username and password. This data **wouldn't match any record** stored in the array which would prompt **an error message**, displayed via pop-up, to the user.
- **Validation (Incorrect username and correct password Or vice versa)** –for this test I will use a partially-correct login. This should not be accepted as both username and password should match the same record for the user to be allowed access on to the system. This **validation will test whether my conditional statements** within my source code are **working effectively** and can identify the correct problem and provide the **necessary output** to the user.
- **Validation (no data)** – for this validation test I will attempt to **login without entering any login data**. This should prompt another error message, this time advising the user to enter data to login, as the system will look for **empty fields and if this condition is met the error message is shown**.
- **Functionality (read from customer login file)** – this test will be to show the that the system is able to **effectively read from the customer login file** and store the data in the login array for the login to able to work. This method will **produce outputs onto the command line** rather than to the user as this process occurs as the system starts up, allowing for the user to use the system straight away.

Tests Planned

Test ID	Test Type	Purpose	Actual Data Used	Pass / Fail
2A	Functional	To test if the login works	Shak: Shakz_01	P
2B	Validation	To prevent incorrect login data	shak: Shakz_01	P
2C	Validation	To detect partially correct data	Shak: Qwerty	P
2D	Validation	To detect and prevent null data input	Null	P
2E	Navigation	To see if correct screens are shown	Button pressed	P
2F	Functional	To show the login file is read from	-	P

Test Commentary and Refinements Needed

2A/2E – for this test I used the correct login data which prompted the screen to switch to the customer menu screen, showing that the data was correct. This showed that as well as confirming the login, the system was able to navigate to the customer menu screen correctly, without displaying any other **output, making the output from the system intuitive.**

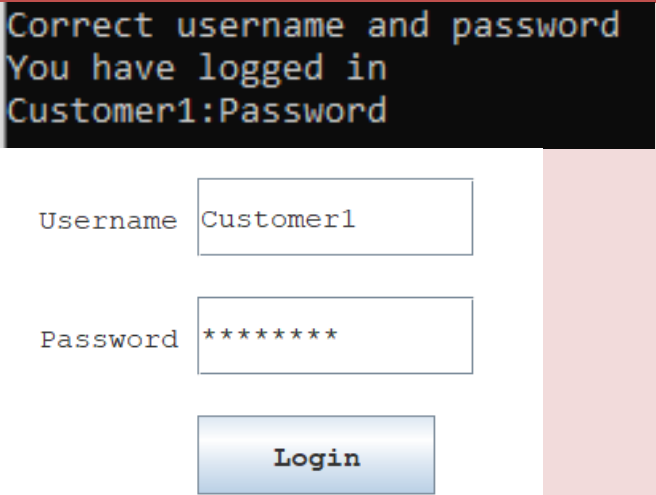
2B – for this validation test I used an incorrect login, where both password and username were incorrect. Once I pressed the login button a **pop-up was displayed saying that both the username and password were incorrect. This worked successfully,** as the system was able to identify **it was an incorrect login.**

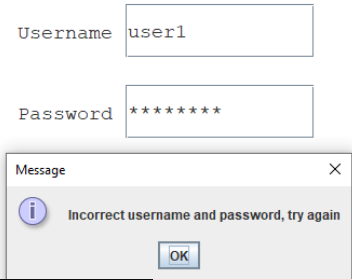
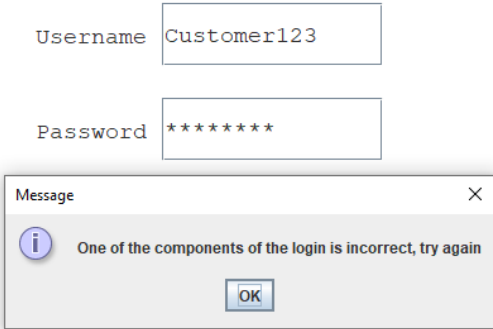
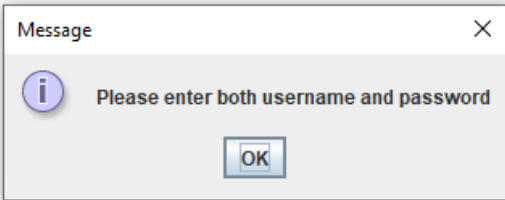
2C – This validation technique is used to **recognise partially correct login data.** I entered a login that was not fully correct and tried to run the system. This prompted the system to display a pop-up saying that the part of the login was incorrect. This **was successful in identifying this problem but also displaying the appropriate error message.** Although the error message **doesn't state which part of the login is correct,** which could be a possible refinement of the system, this can **also act as a security measure** so that other users may not be able to identify another user's login details.

2D – for this test I tried to login without **entering any data into the** login fields. After I pressed the login button, an **error message was displayed** informing me to enter both username and password. This was successful in identifying no input and produced a suitable output to allow the user to amend the mistake and provide the correct data.

2F – this test shows that the customer login file is being read from **when the system is started up.** Once the program is run the methods to read the text file and store each line of data into an array. Although there was **no visual output, in the form of a GUI pop-up,** to show the methods in effect they displayed the data being read each time on to **the command line.**

Results

Test ID	Evidence
1A	
1B	

	 <p>Incorrect user1:password</p>	
1C	 <p>Partially Incorrect Customer123:Password</p>	
1D		
1F	<pre> Running reading customer login file Running addToCustomerLoginList Customer1,Password Running addToCustomerLoginList SShah1,Shak12345 Running addToCustomerLoginList BobSmith1,Bob321 Running addToCustomerLoginList Customer4,Qwerty123 Running addToCustomerLoginList Customer5,QWERTY </pre>	

Objective # & Description

5&6: Read from booking file and view bookings in a table

Discussion of Expected Outcomes

To test these objectives, I will be **running numerous functionality tests as well as performance tests** to see the full capabilities of these processes and how they can handle different situations and see if they remain both **functional and efficient to maintain a seamless human-computer-interaction**.

- **Functionality (Read from the booking file and add to list)** – to test the functionality of this process I will be running the system **along side my source code to see each method that is being run**, as they are displayed onto the command line. This will be able to highlight the method which **will read from the file** as well as another method which will display each record of data on the command line too as each record is added and **stored to an array of booking data**. I expect the methods to be run **correctly with all the data** from the file being stored into the booking array.
- **Functionality (populating table with array data)** – to test whether the system can view bookings in the table, **again I will be using white box testing techniques** to view the methods being run and comparing expected behaviour to current behaviour as the methods being run **will display messages on the command line**, which is necessary as this process doesn't require any set output to the user. I expect that the table will be **populated with all the data from the booking array** and will be clear in displaying the bookings in a user-friendly way.
- **Performance (reading bookings 1000 & 5000)** - to fully test the effectiveness of this process I will need to **stress test the system under both typical and extreme conditions**. I expect the system to be able to cope with both amounts of bookings although they may differ in speeds of execution.

Tests Planned

Test ID	Test Type	Purpose	Actual Data Used	Pass / Fail
5A	Functional	To show it can read from the booking file and add data to the array	-	P
5B	Functional	To show the table is populated correctly	Button pressed	P
5C	Performance	To measure performance in different conditions	-	P

Test Commentary and Refinements Needed

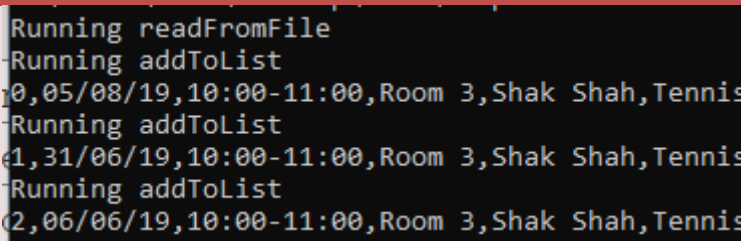

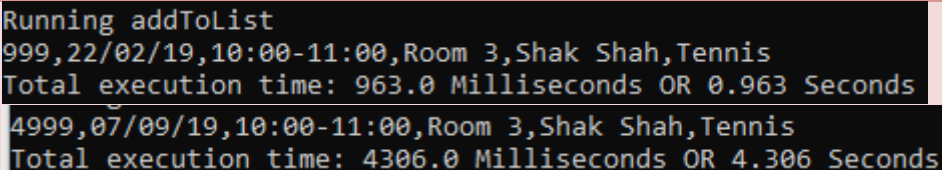
5A – for this test I simply ran the program for my system **which should run the all read file methods including read from booking file**. This process does not require any formal output to the user as it **occurs in the background** without any input required. Therefore, for this process I **ensured the method displayed the data** on the command line of each record that was read from the file and added to the booking array. This allowed me to see if all the data

was being read as it should, which it did showing that the **read from booking file method is functional**.

5B – for this test I clicked to go onto the view bookings page from the menu. Due to my **white box testing** I was able to see once that button **was clicked the method to populate the table** was run. In this method the record for each booking is split up into each attribute and stored into another temporary string array which is the used to add the data to the table. Each record of data is **also displayed onto the command line again** as this is not relevant to the user to see, only the complete table. This process worked effectively displaying the booking data into a table **which when using the system in real time occurred** before the page was changed to the view bookings page. As these methods occur behind the scenes it allows the user interaction to feel seamless **and makes the system more user friendly and responsive**.

5C – for this test I added separate **1000 and 5000 bookings**, each time closing the system completely so that it can be run again to allow it to read all the bookings files again. When I read the booking file containing 1000 bookings it took **approx. 0.9 seconds which is quite effective and efficient**. When I ran the program using 5000 bookings the read time was **approx. 4 seconds**. Although **this speed is a lot slower than** reading 1000 bookings, due to these processes occurring as the system starts up and away from the **user's eye it may not seem as long in real time**. This will therefore not reduce the efficiency of the system massively and may even have no effect at all.

Results

Test ID	Evidence
5A	 <pre>Running readFromFile Running addToList 0,05/08/19,10:00-11:00,Room 3,Shak Shah,Tennis Running addToList 1,31/06/19,10:00-11:00,Room 3,Shak Shah,Tennis Running addToList 2,06/06/19,10:00-11:00,Room 3,Shak Shah,Tennis</pre>
5B	<div> <div> Moved to Booking Page Running clearbookingtable Running populatebookingtable 0 05/08/19 10:00-11:00 Room 3 Shak Shah Tennis 1 31/06/19 10:00-11:00 Room 3 Shak Shah Tennis 2 06/06/19 10:00-11:00 Room 3 Shak Shah Tennis </div> <div>  </div> </div>
5C	 <pre>Running addToList 999,22/02/19,10:00-11:00,Room 3,Shak Shah,Tennis Total execution time: 963.0 Milliseconds OR 0.963 Seconds 4999,07/09/19,10:00-11:00,Room 3,Shak Shah,Tennis Total execution time: 4306.0 Milliseconds OR 4.306 Seconds</pre>

Objective # & Description

7: Sort bookings by date

Discussion of Expected Outcomes

To test this objective, I will be running **functional and logic** tests on my bubble sort, which sorts the bookings by date. To do this I will be testing the process works effectively in standard situations and **assessing its efficiency when it is run**. I will also be running some performance measurement tests with various amounts of bookings to identify any limitations to the process and **the potential impacts on efficiency and user experience**. When the sort is run the bookings in the table should be sorted by date, in either ascending or descending order (chosen by the user).

- **Functional and logic** – this test will be run in standard conditions with both sorts on date being run. This will be able to **portray the sort in effect and will display the method being run on the command line**. Once the sort has been run the table is re populated, using the now sorted data, to display the data to the user. There will **be no other output to the user**, other than the changes made within the table.
- **Performance (Sorting 500 & 1000 bookings)** – in the standard conditions test there was 100 bookings, therefore, to test its performance I will run **it using 500 and 1000 bookings**. This will be able to show how well the process can handle **larger amounts of data at once but also how efficient the process is altogether**. I would expect both tests to pass successfully although I would expect a slower performance when using 1000 bookings. After the test **I will be commenting on any potential refinements** that could be implemented if at all and whether they would be impactful on the human-computer-interaction.
- **Performance (Sorting 5000 bookings)** – I will also be testing the process in extreme circumstances, which will further show how well the process can cope with larger data sets and will show me any potential limitations of the system. These again will help me identify possible area which could require any refinements but also if the current implementation is an effective and efficient way of sorting data within this system.

Tests Planned

Test ID	Test Type	Purpose	Actual Data Used	Pass / Fail
7A	Functional & Logic	Shows the sort working effectively	Text file data – booking records	F
7B	Performance	efficiency of sorting Bookings (500&1000 Bookings)	Text file data – booking records	P
7C	Performance	efficiency of sorting Bookings (5000 Bookings)	Text file data – booking records	F

Test Commentary and Refinements Needed

7A – for my first test I simply loaded up the view bookings screen and ran the **sort by date (oldest bookings first)**, by selecting the option from the combo box and clicking the filter button. Once I had done this the table **updated its data and applied the filter** on to the bookings. However, once the data was updated as I scrolled through the table, I **had realised that, although there was no error** with the code or the functionality of my sort method, the sort **did not work as I intended**. Instead of sorting the date, considering days, months and year, it **had only considered the day of the booking** and had sorted each booking according to that. Although the functionality of the sort was effective it did **not pass the logic test** as bookings that were made recently would be shown near the top of the list, due to them being at the end of the month. To solve this, I **changed the approach to my sort** and decided to make use of the **LocalDate class**, by converting the dates from the bookings into a LocalDate object. Once I had done this, I **could use the 'isBefore/isAfter' methods** to effectively sort my bookings in the way I had initially intended, to **which they did once I re-tested the sort**.

7B – To test my sort extensively I decided to run it under various conditions, with **100 bookings** and **with 1000 bookings**, to see how efficient it is and whether it will affect the user experience in how well the sort performs. I initially tested it with 100 bookings, using both date sorts and it worked effectively both times. To measure the performance, I **implemented a few lines of code that would calculate the execution time for each sort**. When I ran the sort using 100 bookings the execution time **was approx. 0.094 seconds**. (with each sort being run on the original order of the data to ensure accurate results) Using 1000 bookings when the sort was run the **execution time was approx. 1.2 seconds**. This showed me **the performance of the sort in two contrasting situations** and both were able to function effectively and complete the sort in an efficient way. However, when the sort was performed on 1000 bookings although one second is still fast, **its relative speed was not**. This also prompted a slight delay on the updating of the table **which could be a potential refinement** in the future to make the process as seamless as possible, even though there was minimal effect on the user experience.

7C – as I was testing the sort with contrasting booking sizes I also decided to try and run it **with 5000 bookings** to see whether the system would still be able to run the process in a relatively suitable amount of time. However, when I ran the sort the **system effectively crashed**, where the **screen froze** and would not take any other inputs or make outputs. The system did begin to function again after completing the sort, which took **approx. 25 seconds**, an extremely long time to complete. This clearly identified a limitation of the process in handling large amounts of records and showed that my current **way to sort the data may not be the most effective way of doing so**, especially in a much larger scale system.

Results

Test ID	Evidence			
7A			4	02/01/19
			6	02/01/19
			77	02/01/19
	87	08/07/19	78	02/01/19
	57	08/08/19	15	05/01/19
	91	08/08/19	95	05/01/19
	64	08/09/19	23	06/01/19
	65	08/09/19	35	08/01/19
	82	09/01/19	66	08/01/19
	56	09/02/19	88	08/01/19
7B	Running Bubble sort by latest date Total execution time: 97.0 Milliseconds OR 0.097 Seconds Running populatebookingtable			
	Running Bubble sort by oldest dates Total execution time: 94.0 Milliseconds OR 0.094 Seconds Running populatebookingtable			
	Running Bubble sort by latest date Total execution time: 1379.0 Milliseconds OR 1.379 Seconds Running populatebookingtable			
	Total execution time: 1235.0 Milliseconds OR 1.235 Seconds Running populatebookingtable			
	Running Bubble sort by latest date Total execution time: 25892.0 Milliseconds OR 25.892 Seconds Running populatebookingtable			
	Running Bubble sort by oldest dates Total execution time: 25938.0 Milliseconds OR 25.938 Seconds Running populatebookingtable			
7C	Total execution time: 21718.0 Milliseconds OR 21.718 Seconds			

Objective # & Description

8: Sort bookings by name

Discussion of Expected Outcomes

To test this objective, I will be running **functional and performance** tests on my bubble sort, which sorts the bookings by date. I will be running the performance tests with various amounts of bookings to see how well it can handle larger amounts of data and still work efficiently.

- **Functional** – this test will be run the sort by name on the data as it is in the text file. This will be able to show that the sort works effectively and once the sort has been run the table is updated to display the sorted data to the user.
- **Performance (Sorting 500 & 1000 bookings)** –to test its performance I will run the sort on 500 and 1000 bookings. This will be able to show how efficient the process is and how the user's interaction with the system will be affected. After the test I will be deciding whether there could be any refinements that could be implemented to improve the system's efficiency and make it more intuitive.
- **Performance (Sorting 5000 bookings)** – I will also be testing the with 5000 booking records in the table, which will again show how well the process can cope with larger data sets and whether the system remain functional. I still expect the sort to work yet it may take longer to update the table with the sorted data.

Tests Planned

Test ID	Test Type	Purpose	Actual Data Used	Pass / Fail
8A	Functional	Shows the sort working effectively	Text file data – booking records	P
8B	Performance	efficiency of sorting Bookings (500&1000 Bookings)	Text file data – booking records	P
8C	Performance	efficiency of sorting Bookings (5000 Bookings)	Text file data – booking records	P

Test Commentary and Refinements Needed

8A –for this test I ran the **sort by name on the original data** in the table. Once I had done this the table updated showing the new sorted data. The data **was sorted in alphabetical order and was able to complete the sort smoothly without any delays**. There was no other output when run apart than the updated table. However, to ensure the correct methods were also being run it was displayed onto the command line, showing that **the table is repopulated after the sort method has been run**.

8B –I initially tested the sort with 500 bookings and measured the performance. The execution time was approx. **0.013 seconds**. Sorting **1000 bookings took approx. 0.028 seconds**. This showed me that even with two amounts of data the sort remained functional but also efficient as they were executed in less than a second each time. The **larger amount of records to be sorted made very little difference** and would not affect the user interaction at all.

8C – when testing the sort on **5000 bookings it took approx. 0.5 seconds** to complete the process, which again is a very quick and efficient time to execute the search as it completed in less than a second. Although **it was slightly slower than the other tests**, using an extreme amount of bookings to sort, **it still performed smoothly without any delays**, or crashes to the system.

Results

Test ID	Evidence	
8A	Name	Name
	Bob Smith	Boone Kidney
	Lilian Spellar	Boone Kidney
	Boone Kidney	Bronny Vanyakin
	Boone Kidney	Bronny Vanyakin
	Lilian Spellar	Bronny Vanyakin
	Lilian Spellar	Granthem Zarfati
	Bob Smith	Granthem Zarfati
	Lilian Spellar	Granthem Zarfati
	Stuart Dearl	Haley Vowden
	Lilian Spellar	Kelsy Kellog
	<pre>Moved to Booking Page Running clearbookingtable Running populatebookingtable running bubble sort by name Total execution time: 1.0 Milliseconds OR 0.001 Seconds Running populatebookingtable</pre>	
8B	<pre>running bubble sort by name Total execution time: 13.0 Milliseconds OR 0.013 Seconds Running populatebookingtable</pre>	
	<pre>running bubble sort by name Total execution time: 28.0 Milliseconds OR 0.028 Seconds Running populatebookingtable</pre>	
8C	<pre>running bubble sort by name Total execution time: 509.0 Milliseconds OR 0.509 Seconds Running populatebookingtable</pre>	

Objective # & Description

9 & 13 Add bookings to the database and calculate price of bookings

Discussion of Expected Outcomes

To test that my bookings can be added to the database I will running multiple **validation tests, a navigation test and a functional test**. The validation tests will show how the system is able to deal with different **data inputs (typical/extreme/erroneous)** and whether the user's data is suitable enough for the data to be added as a booking. The navigation test will ensure the **correct screen is displayed in correspondence** with user input, to allow the next step of adding a booking (the payments screen) finally the **navigation test will ensure that all of the data entered by the user is obtained** from the text fields and stored within the booking array, for other uses such as writing to file and/or viewing bookings.

- **Validation (all fields blank)** – I will initially test this process by entering in no data at all and trying to confirm the booking. This **will illustrate a presence check** by having no fields lefts blank, and an appropriate error message should be displayed to the user informing them to enter data.
- **Validation (unselected drop-down menus)** – I will attempt to add the booking without selecting any of the options from the drop-down menu. This **form of validation test will show whether the system is able to identify that the user hasn't selected a certain option** from one of the drop-down menus. This should be done by ensuring the selected index of the combo-box is not zero (the first index) and if **it is an error message will be displayed**, again informing the user to select an option from the menus.
- **Validation (incorrect email)** – for this test I will be entering an **email address without the '@'**. This valuation technique will be **a format check to ensure that the data** entered by the user is a **correct email address and** therefore would require an '@' contained within the input made by the user. If this symbol isn't found within the input, there will again **be a suitable error message** displayed to the user.
- **Validation (incorrect input for phone numbers)** – for this test I will be entering letters contained within my phone number along with other numbers too. this type check will be used to ensure that the user's input for the phone number will be of the correct data type. Although I am storing the phone numbers as string values, this form of validation will try to parse each character of the data into an integer ensuring that no other characters have been entered. This will again display an error message if one of the characters is unable to be parsed into an integer.
- **Validation (incorrect phone number)** – for this test I will be entering a phone number which is not of the correct length for a phone number, more specifically with a phone number that is too short and one that is too long. This final validation check will be a length check on the text field which contains the customer's phone number and will require the phone number entered to be of a certain length (11 digits). An error message will be displayed if the data is not of the correct length and will inform the user that it is not a suitable phone number.

- **Functionality and Navigation test** – once the booking data is suitable to be added and the booking button is pressed, the screen should change to the payments page. At the same time a method should be run which obtains all the data from the text fields and menus and adds it to the booking list array.
- **Functional test (Calculating price)** – for this test I will be creating multiple bookings all with different activities to show that the calculations for the price work effectively. The price is a set price and will depend on the activity chosen by the user which will be checked for the price to be displayed on the payments screen.

Tests Planned

Test ID	Test Type	Purpose	Actual Data Used	Pass / Fail
9A	Validation	To identify null input	-	P
9B	Validation	To identify null input (combo box)	-	P
9C	Validation	To identify incorrect format for emails	Shak.gmail.com	P
9D	Validation	To identify letters in phone number input	07Q5A42S279	P
9E	Validation	To identify incorrect length of phone number	0775742427 077574242791	P
9F	Navigation	To show that it can move to the correct screen when required	-	P
9G	Functional	To show the booking data is obtained and added to the booking list	-	P
9H	Functional	Price for booking is calculated	-	P

Test Commentary and Refinements Needed

9A – for this test I entered in no data into the **text fields for name, phone number or email**. Once I tried to confirm the booking **an error message was displayed via** a pop-up that I should fill in all the boxes. This was successful in identifying that there were blank fields and displayed an error message effectively. A **possible refinement would be to highlight the blank fields to make it clear to the user on which fields they would have to enter**, as the user may be a novice user and it would make it more user friendly.

9B – for this test I **did not select any data from the drop-down menus**, but I did enter suitable data for the text fields. When I tried to confirm the booking using this data another pop-up was displayed saying that I should select an option from the menus. This again was a **successful use of validation and can effectively check** that the data from the menus had not been selected. Again, **similarly to the first test a way of highlighting** the specific menus that were left blank could be **a possible refinement to make the user interaction slightly smoother** to avoid any confusion between the system and the user.

9C – for this test I **entered an email that did not contain the '@' symbol** whilst entering all other data correctly to allow this to be the only error on the screen. Once I tried to confirm the booking similarly to the other **an error message was displayed on the screen**, advising me to

enter a valid email address. This **validation test worked effectively** and was able to identify the error and inform the user on correcting it.

9D – for this test I **entered a mix of letters and number in the phone number** field and tried to add the booking. Once I pressed the search button a pop-up was **displayed that prompted me to enter a valid phone number**. This validation worked and was able to try and parse each character of the data into an integer to check if the data contained any non-integers. If found the error message would be displayed.

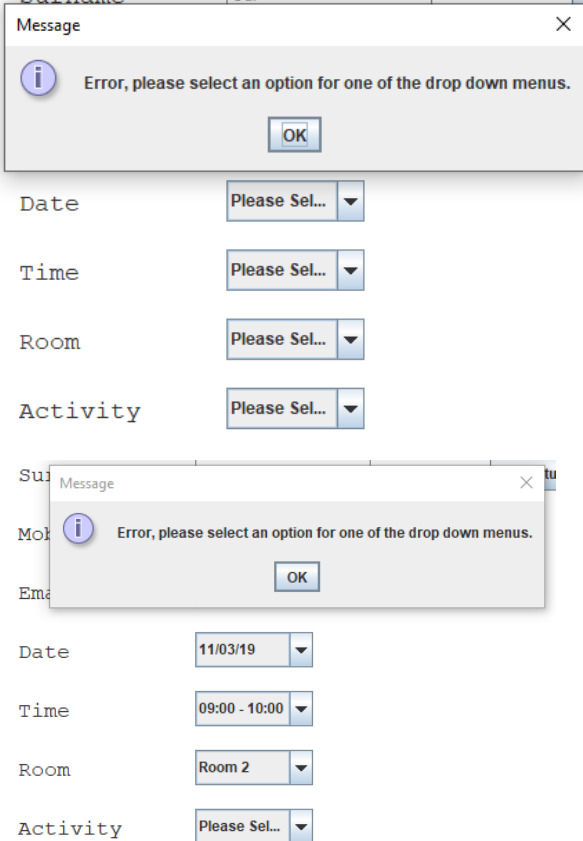
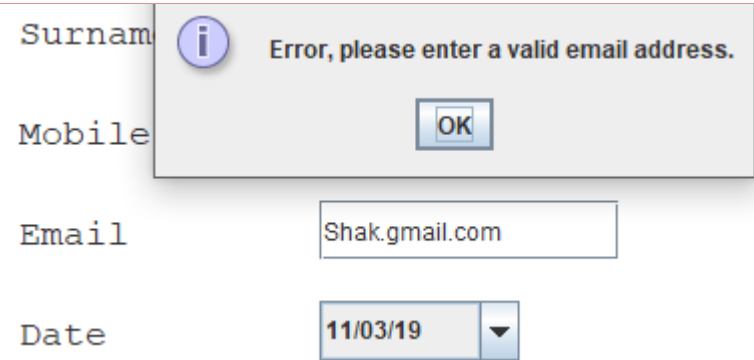
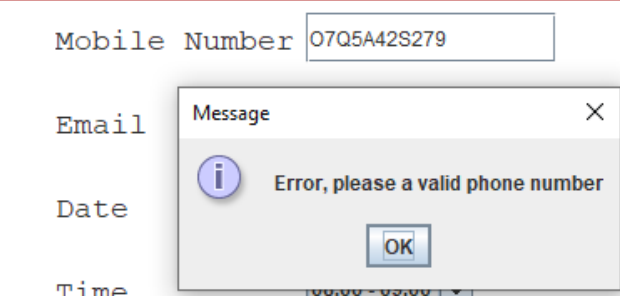
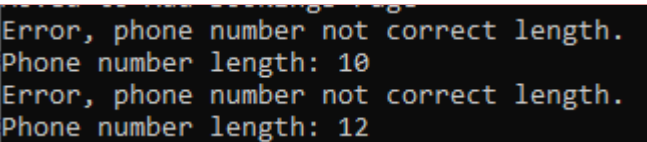
9E – for this test I **entered a phone number that was too long and too short to be a phone number**. Once I had pressed the enter button for each set of data error message was displayed saying the phone number is not of **a correct length and to enter a suitable phone number**. **As I am using white box testing**, I was able to see the exact method and ‘if statement’ being run to ensure this validation was successful. As a slight extension of this **validation I decided to also output the length of the phone number entered each time on the command line**. Although the user wouldn’t see this in general use, for me at the alpha testing stage it is helpful to **confirm the validation checks are behaving as they should**.

9F & 9G – for the final test I **simply entered correct data for all the text fields** and tried to confirm the booking. As the data was legal, there was no error message displayed onto the screen and the screen then **changed to the payments screen**. Although there was no other feedback/output to the user on the command line it displayed a message saying the ‘add to list’ method was being run as well as showing the record of data entered. This test was successful as the booking data **was successfully added to the booking array as a record, with the navigation also working effectively**.

9H – for this test I **created 5 bookings each** with a different activity. As I made each booking for the price would be displayed on the payments page depending on the activity I had chosen. Also, the data was shown on the **command line showing the activity and price for that**. This worked effectively as all the prices were as intended making it an effective process.

Results

Test ID	Evidence
9A	<div><div><div>First Name</div><div></div></div><div><div>Surname</div><div></div></div><div><div>Mobile Number</div><div></div></div><div><div>Email</div><div></div></div></div> <div><div>Message</div><div><div></div><div>Error, please fill in all boxes</div><div>OK</div></div></div>

9B	 <p>Message</p> <p>Error, please select an option for one of the drop down menus.</p> <p>OK</p> <p>Date Please Sel... ▼</p> <p>Time Please Sel... ▼</p> <p>Room Please Sel... ▼</p> <p>Activity Please Sel... ▼</p> <p>Message</p> <p>Error, please select an option for one of the drop down menus.</p> <p>OK</p> <p>Date 11/03/19 ▼</p> <p>Time 09:00 - 10:00 ▼</p> <p>Room Room 2 ▼</p> <p>Activity Please Sel... ▼</p>
9C	 <p>Surname</p> <p>Mobile</p> <p>Email Shak.gmail.com</p> <p>Date 11/03/19 ▼</p> <p>Message</p> <p>Error, please enter a valid email address.</p> <p>OK</p>
9D	 <p>Mobile Number 07Q5A42S279</p> <p>Email</p> <p>Date</p> <p>Time 10:00 - 05:00 ▼</p> <p>Message</p> <p>Error, please a valid phone number</p> <p>OK</p>
9E	 <p>Error, phone number not correct length. Phone number length: 10</p> <p>Error, phone number not correct length. Phone number length: 12</p>

9H	<pre>113,15/03/19,10:00 - 11:00,Room 2,Bob Smith ,Basketball Running writeListToFile For Bookings Running addToList 114,0658497810346857,B Smith,20 Running writeListToFile For payments Running writeListToFile For CustomerInfo Activity: Basketball Price: 20</pre>	<pre>114,19/03/19,13:00 - 14:00,Room 3,Bob Smith ,Badminton Running writeListToFile For Bookings Running addToList 115,0658497810346857,B Smith,15 Running writeListToFile For payments Running writeListToFile For CustomerInfo Activity: Badminton Price: 15</pre>
	<pre>115,16/03/19,18:00 - 19:00,Room 5,Bob Smith ,Table Tennis Running writeListToFile For Bookings Running addToList 116,0658497810346857,B Smith,10 Running writeListToFile For payments Running writeListToFile For CustomerInfo Activity: Table Tennis Price: 10</pre>	<pre>116,15/03/19,19:00 - 20:00,Room 4,Bob Smith ,Tennis Running writeListToFile For Bookings Running addToList 117,0658497810346857,B Smith,15 Running writeListToFile For payments Running writeListToFile For CustomerInfo Activity: Tennis Price: 15</pre>
	<pre>117,17/03/19,12:00 - 13:00,Room 5,Bob Smith ,Football Running writeListToFile For Bookings Running addToList 118,0658497810346857,B Smith,25 Running writeListToFile For payments Running writeListToFile For CustomerInfo Activity: Football Price: 25</pre>	

Objective # & Description
10: Write to a file – booking database

Discussion of Expected Outcomes

To test this objective, I **will be running a functional test as to see whether the methods for the process works effectively**. As this is only writing to file, I will only require a functional test to be run, yet similarly to the other objectives I will also **be measuring the performance under** different conditions and stress testing it too see how well the file handling is under both typical and extreme data sizes. For the **performance test I have created a method within my system that will add many bookings at the same time**, which will stress test the system to identify any inefficiencies or limitations. **(note that the method used is only for testing purposes and is not an implementation)**

- **Functional test (write to booking file)** – to test this extensively I will initially be adding a booking as a user would normally interact with the system. I will **be entering correct data and confirming the booking** (which occurs on the payments page). I expect that the booking will be added effectively and will also be able to **update the booking table simultaneously**. To confirm this, I will check the text file where the booking is added to as well as the, view bookings table to ensure that consistency and integrity is maintained throughout the system.
- **Performance tests (adding 100/1000 bookings at once)** – to test the systems effectiveness at file handling used the method I created to add 100 and 1000 bookings at once to the system, yet to do this the data for each booking is identical, expect for the booking ID. I expect both to work effectively although they may differ from each other in terms of speed and inefficiency. **I will also be commenting on any potential downfalls** of large data handling in terms of its **effect on the user** and whether it will affect the user-friendliness
- **Performance test (adding 5000 bookings at once)** – I will also be adding 5000 bookings at once as an extreme example of adding bookings data, which will be able to also **identify any faults in the system but also whether my current implementation is the most efficient way to write to the file**.

Tests Planned

Test ID	Test Type	Purpose	Actual Data Used	Pass / Fail
10A	Functional	Shows the write to file method working effectively	-	P
10B	Performance	efficiency of adding Bookings (100&1000 Bookings)	-	P
10B	Performance	efficiency of sorting adding (5000 Bookings)	-	F

Test Commentary and Refinements Needed

10A – for this test I **entered suitable and correct data** for all the booking data inputs. Once I had pressed to confirm the booking and switched to the payments page, I also entered the correct data for that screen to. Once I **had confirmed the payments there is a pop up** displayed to the user saying that the booking has been added as well as payments have been confirmed. Although this is **the only output to the user, on the command line once the method is run, the data that has been added it displayed**, as well as the actual record stored in the text file **which includes the booking ID which is automatically assigned**. This worked effectively and is a simple way for the user to interact with the system, with any feedback also being simple and easy to understand. **This process is one which doesn't require much refinement as all process** occur off screen therefore not impacting the user experience in any way making it intuitive.

10B – for this test I used the method I created to **add 100 bookings at once**. As well as that I had added the code, I used to time the method so that the performance of the process could be measured. **When adding the 100 bookings at once it too approx. 0.8 seconds** to add them, which is an efficient amount of time to add bookings at once without any delays. When I added **1000 bookings at once it took approx. 10 seconds to do so**. This could be quite a problem if it were to happen as the difference in execution times between the two data sizes is quite large. **Adding 1000 bookings at once could cause a slowdown in the system which would cause issues for any user on the system at the same time**. This could be a potential refinement of the system in the future of the current solution or instead using a different approach.

10C – for this final performance test I **used 5000 bookings to be** added at once to see the outcome of an extreme number of bookings being added and how the system would handle that. **When I did this, it took 45 seconds to added them all**, which is a very long time. As well as that the system would not respond to any other requests, even simple navigation, therefore although the system can add these many bookings at once, it does not do it efficiently. This **helps me identify a clear limitation of my system** which even though are under extreme conditions, may show that my current approach to adding files **is not effective in a larger scale system**.

Results

Test ID	Evidence												
10A	<div><div>Total Payment: Â£15</div><div>umber</div><div>older</div><div><div>Message</div><div><div>i</div><div>Booking Added</div></div><div>OK</div></div></div> <div><div>otal Payment: Â£15</div><div>r</div><div>r</div><div><div>Message</div><div><div>i</div><div>Payment Made of: Â£ 15</div></div><div>OK</div></div></div> <div><div>Running addToCustomerInfoList</div><div>Jim Smith,01204430819,Jim@gmail.com</div><div>Running addToList</div><div>100,14/03/19,11:00 - 12:00,Room 4,Jim Smith,Tennis</div><div>Running writeListToFile For Bookings</div><div>99,03/08/19,10:00-11:00,Room 3,Shak Shah,Tennis</div><div>100,14/03/19,11:00 - 12:00,Room 4,Jim Smith,Tennis</div><table><tr><td>99</td><td>03/08/19</td><td>10:00 - 11:00</td><td>Room 3</td><td>Shak Shah</td><td>Tennis</td></tr><tr><td>100</td><td>14/03/19</td><td>11:00 - 12:00</td><td>Room 4</td><td>Jim Smith</td><td>Tennis</td></tr></table></div>	99	03/08/19	10:00 - 11:00	Room 3	Shak Shah	Tennis	100	14/03/19	11:00 - 12:00	Room 4	Jim Smith	Tennis
99	03/08/19	10:00 - 11:00	Room 3	Shak Shah	Tennis								
100	14/03/19	11:00 - 12:00	Room 4	Jim Smith	Tennis								
10B	<div><div>Running writeListToFile For Bookings</div><div>Total execution time: 846.0 Milliseconds OR 0.846 Seconds</div><div>Running writeListToFile For Bookings</div><div>Total execution time: 10890.0 Milliseconds OR 10.89 Seconds</div></div>												
10C	<div><div>Running writeListToFile For Bookings</div><div>Total execution time: 45011.0 Milliseconds OR 45.011 Seconds</div></div>												

Objective # & Description

11: Write to a file – customer information file

Discussion of Expected Outcomes

For this objective, I will be **testing functionality and measuring its performance** to see whether the process works effectively under different conditions and stress testing it too see how well the file handling is. As **this process would usually occur alongside the 'write to booking file' process**, this does not require any validation or navigation tests meaning there is no output to the user on this process as it will occur in the background only. Therefore, for this objective I will be using command line outputs to support **evidence of its functionality and effectiveness**.

- **Functional test (write to customer information records file)** – for this test I will be adding a booking and making the payment for it too to allow the method for this objective to be executed. I expect that all the data obtained to form the record of customer information will be as intended and effectively written to the file.
- **Performance tests (adding 100/1000 customer information records at once)** –for this I will run the method to add these many bookings all at once. After each number of records is added I will be comparing the times of their execution to see if there is a difference in them and the potential positives or drawbacks of the process. I expect the processes to work effectively and be relatively efficient.
- **Performance test (adding 5000 customer information records at once)** – this test will be a stress test designed to discover any flaws in this process and whether it will impact others during its execution. I will be measuring its performance in terms of speed but also whether it remains functional when handling such large amounts of data.

Tests Planned

Test ID	Test Type	Purpose	Actual Data Used	Pass / Fail
11A	Functional	Shows the write to file method working effectively	-	P
11B	Performance	efficiency of adding customer records (100&1000 Bookings)	-	P
11B	Performance	efficiency of adding customer records (5000 Bookings)	-	F

Test Commentary and Refinements Needed

11A – for this test I added a **booking to the system and made the payment to allow the writing to file processes to be run**. The method to write to the **customer information file** was run alongside other processes and it worked effectively as it was able to obtain data from an earlier stage in the system and store the data as a record into the file. **This process wouldn't require any refinement as it occurs off screen** along with other processes therefore not impacting the user in any way making it user friendly but also as it can be run alongside **other write methods it is able to perform in a real situation**.

11B – for this test I used the method I created **to write 100 and 1000 records to the customer file** and measured the time of its execution. When adding the **100 customer information** records at once it **too approx. 0.3 seconds** to write them to the file, which is a relatively efficient amount of time to write to the file being almost instantaneous. When I added **1000 records at once it took approx. 9.8 seconds to do so**. Although this was a large difference in the between the two execution times, this may **not be an issue as this process is not viewable** for the user to see therefore it would not affect their use with the system.

11C – for the final test I added **5000 records at once**. When I did this, it took **111 seconds to add them all and froze the system** whilst running this process. This allows me to see a clear issue with the way my system can handle large amounts of data and file handling **which would be a refinement of the approach to writing to files**.

Results

Test ID	Evidence
11A	Running writeListToFile For Bookings Running addToList 1,0645 8789 1658 4903,S Shah,20 Running writeListToFile For payments Running writeListToFile For CustomerInfo
11B	Running writeListToFile For CustomerInfo Total execution time: 377.0 Milliseconds OR 0.377 Seconds Running writeListToFile For CustomerInfo Total execution time: 9871.0 Milliseconds OR 9.871 Seconds
11C	Shak Shah,07757424279,Shak@gmail.com Running writeListToFile For CustomerInfo 4999 Total execution time: 111125.0 Milliseconds OR 111.125 Seconds

Objective # & Description

12: Write to payments file

Discussion of Expected Outcomes

To test these objectives, I will be running a functional test as to see whether the methods for the process works effectively, along with a few validation tests to see whether the process is able to deal with different variations of data inputs, such as typical, extreme, erroneous data. I will also be measuring the performance under different conditions and stress testing it too see how well the file handling is under both typical and extreme number of bookings. For the performance test I will be using the method I used to test a previous objective where, the system will add many bookings at the same time, which will stress test the system to identify any inefficiencies or limitations. **(again, this method used is only for testing purposes and is not an implementation)**

- **Functional test (write to payments file)** – to test this extensively I will be entering correct data and confirming the payment of the booking. **I expect that the payment information will be added to the array and thus the payments file effectively** and will also be able to update the payments table as well. To confirm this, I will check the text file where the payment data is stored as well as the, view payments table to **ensure the correct data is updated throughout the entire system.**
- **Validation (null inputs for both fields)** – for this test I will try to make a payment without providing any input. This will test for **a presence check validation**, which I expect will work effectively with the system responding with an appropriate error message.
- **Validation (letters in card number)** – for this test I will be using letters instead of number for the card number input. I expect that the system will **effectively identify this error by parsing each character of the data into an integer**, and if it can't do so the data will be **flagged causing an error message** to be prompted.
- **Validation (incorrect length of card number)** – this validation test will ensure that the data entered for the **card number is of the correct length of 16 digit**. This will again display another error message to the user but will also display the number of **digits onto the command line so that I can ensure its effectiveness.**
- **Performance tests (making 100/1000 payments at once)** – to test the systems effectiveness at file handling I will be using the method I created to add **100 and 1000 payments at once to the system**, although the data will be identical, expect for the booking ID. As the records for payments contain less attributes and are smaller than that of a booking, I expect both to work effectively and maybe even write to the file faster.
- **Performance test (making 5000 payments at once)** – I will also be making 5000 payments at once as a stress test, which will be able to also identify any faults in the system but also whether my current implementation is the most efficient way to write to the file.

Tests Planned

Test ID	Test Type	Purpose	Actual Data Used	Pass / Fail
12A	Functional	To show the file is written to	Shown in evidence	P
12B	Validation	To identify null input	Null input	P
12C	Validation	To identify letters in card number field	Shown in evidence	P
12D	Validation	To identify incorrect length of card number	Shown in evidence	P
12E	Performance	efficiency of making payments (100&1000 Bookings)	-	P
12F	Performance	efficiency of making payments (5000 Bookings)	-	F

Test Commentary and Refinements Needed

12A – for this test I entered correct data for the payment which **would not prompt an error message**. Once this was **confirmed two pop-ups were displayed** once saying the booking was added and another to say the payment was made. **This test was successful as the payment data was successfully stored to the payments file**. The data was also displayed on the command line so that I could see that the data was correctly obtained and written to the file.

12B – for this test I entered **in no data into the text fields for the payments data**. Once I tried to confirm the payment **a pop-up error message was displayed** to fill in the all fields. This was successful in identifying that there were blank fields and displayed an error message accordingly. This is **quite user friendly as it can help novice users learn how** to use the system effectively in case, they make a mistake during the payment process.

12C – for **this validation test I entered a card number with a mix of letters as well as numbers** and a correct input for the card holder field. Once I pressed the pay button an error message was **displayed saying that I should enter a valid card number**. As the input contained both number and letters the validation **check parsed each character of the input into an integer and stopped when it came across a letter** that couldn't be parsed, prompting the error message. This check worked effectively and allowed for **the prevention of incorrect data being written to the payments file**.

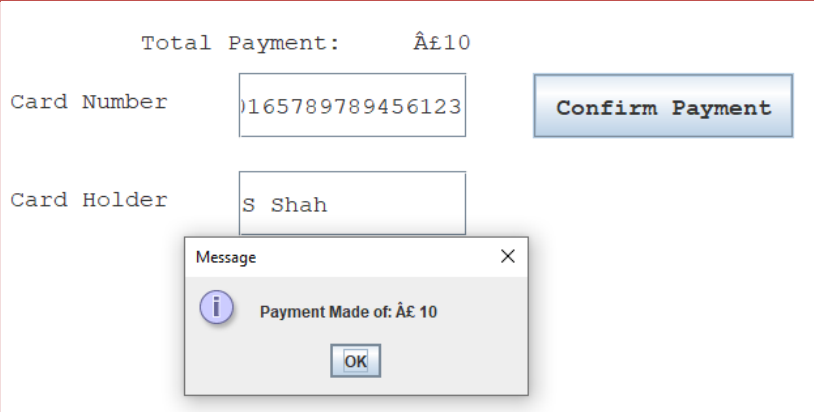
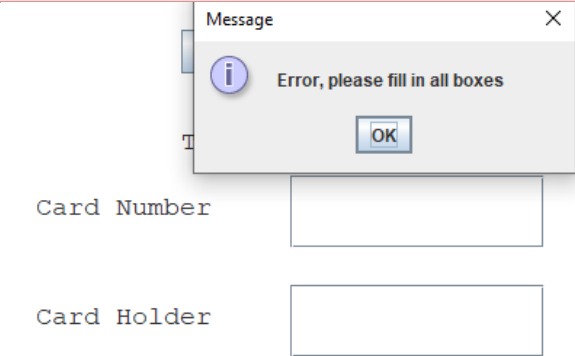
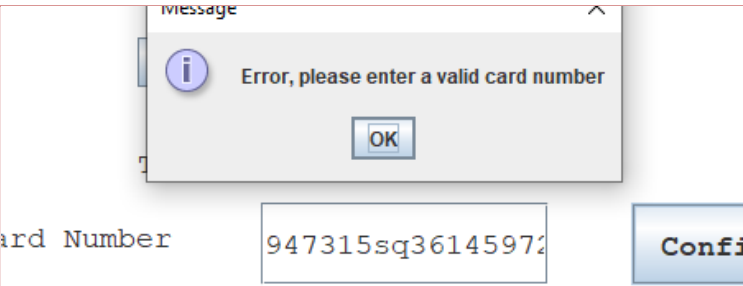
12D – for this test I entered a card **number that was not of the correct length**. Once I had pressed the enter button for each set of data error message was displayed saying the card number is not of a correct length and to **enter a valid card number**. I decided to also output the **length of the phone number entered each time on the command line** to confirm that this validation was successful.

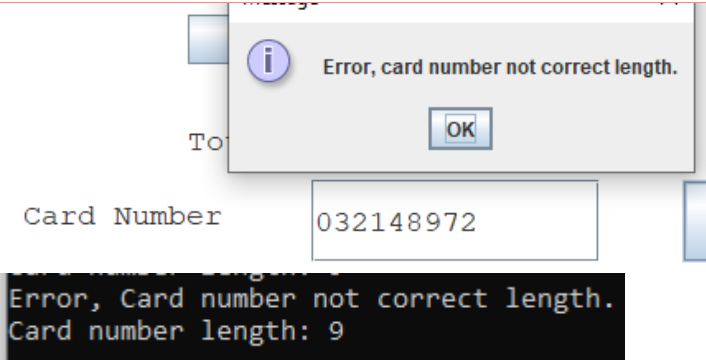
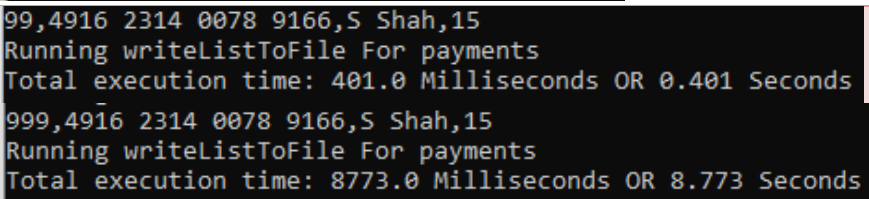
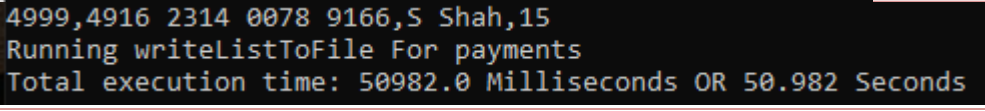
12E – for this test I used the method I created to make 100 payments at once. When making these payments **at once it too approx. 0.4 seconds to do so**, which is a relatively efficient amount of time. When I ran the method to make **1000 payments it took approx. 8.7 seconds** to do so. Although both were successful in making the payments the difference in executions

speeds is quite large. This **could be a potential refinement of the system as the system may crash or slowdown** if there is a large amount of payments being made at once.

12F – for this final performance test I **made 5000 payments at once** to see how the system would handle this much data. **When I did this, it took approx. 50 seconds to added them all**, which is **not an ideal time and very inefficient**. The system also stopped responding during this process which although the process works, it is not an effective way of doing so. This help me identify a clear limitation of my system and **shows that my current approach to writing files is not as effective in a larger scale system**.

Results

Test ID	Evidence
12A	
12B	
12C	

12D	 <p>The screenshot shows a Windows-style error dialog box with a blue 'i' icon. The text inside the dialog reads: "Error, card number not correct length." Below the text is an "OK" button. In the background, a text input field labeled "Card Number" contains the value "032148972". Below the input field, a black console window displays the following text: "Error, Card number not correct length. Card number length: 9".</p>
12E	 <p>The screenshot shows a black console window with white text. It displays two successful payment processing runs. The first run shows a card number "99,4916 2314 0078 9166" for "S Shah,15", followed by "Running writeListToFile For payments" and "Total execution time: 401.0 Milliseconds OR 0.401 Seconds". The second run shows a card number "999,4916 2314 0078 9166" for "S Shah,15", followed by "Running writeListToFile For payments" and "Total execution time: 8773.0 Milliseconds OR 8.773 Seconds".</p>
12F	 <p>The screenshot shows a black console window with white text. It displays a failed payment processing run. The first line shows a card number "4999,4916 2314 0078 9166" for "S Shah,15". The second line shows "Running writeListToFile For payments". The third line shows "Total execution time: 50982.0 Milliseconds OR 50.982 Seconds".</p>

Objective # & Description

14: delete bookings

Discussion of Expected Outcomes

To test this objective, I will be running functionality and validation tests, to see how well the system is able to delete bookings from the system and whether it will prevent any accidental deletion of records.

- **Functionality** – for this test I will be deleting a booking from the system by selecting a record from the table displayed on the view bookings screen. Once selecting the booking, **I will select the delete booking button to run the process. I expect a pop-up to be displayed confirming my decision to which the booking will be deleted and removed from both the table and the booking file.** There will also be an output confirming the booking has been deleted both on the screen and command line.
- **Validation (deleting without selecting a booking)**- this form of validation will be check whether a booking has been selected by the user. If not, **an error message will be displayed** saying that the user should select a booking to cancel. I will also be commenting on **whether this form of output to the user is effective** and user friendly and whether it is a useful output **for all types of users.**
- **Validation (accidental deleting booking)** – for this test I will be deleting a booking **‘by accident’** and seeing how the system will respond to this. I expect there to be **a confirm pop-up** which will allow me to **stop the deleting of the booking.** I will also be commenting **on if this output is enough to prevent** accidental booking and whether this could be **refined in the future.**

Tests Planned

Test ID	Test Type	Purpose	Actual Data Used	Pass / Fail
14A	Functional	Booking is deleted effectively	Button pressed	P
14B	Validation	Can identify unselected booking	Button pressed	P
14C	Validation	Can show a confirmation for deleting booking	Button pressed	F

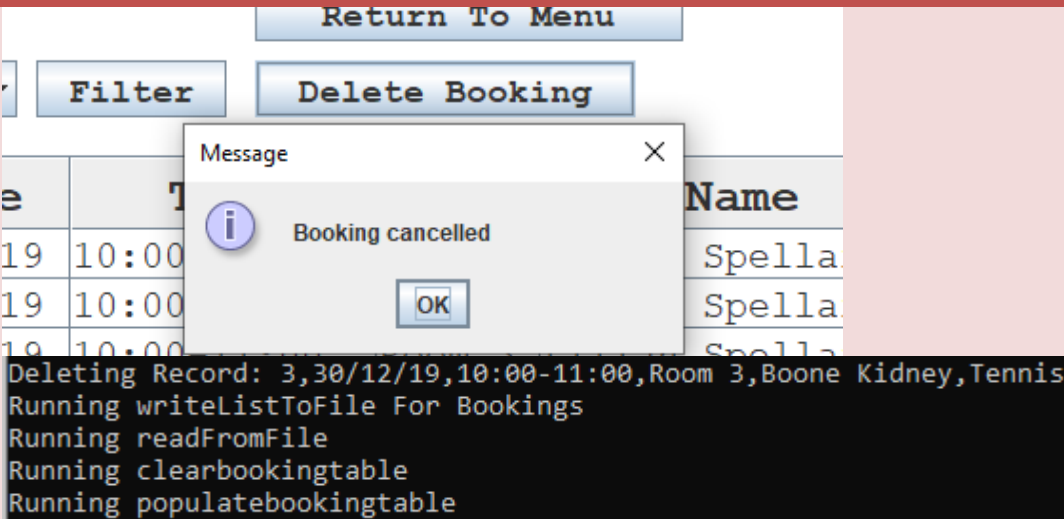
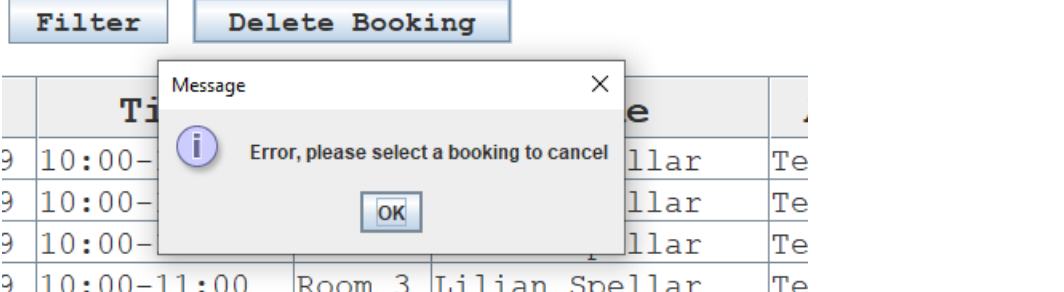
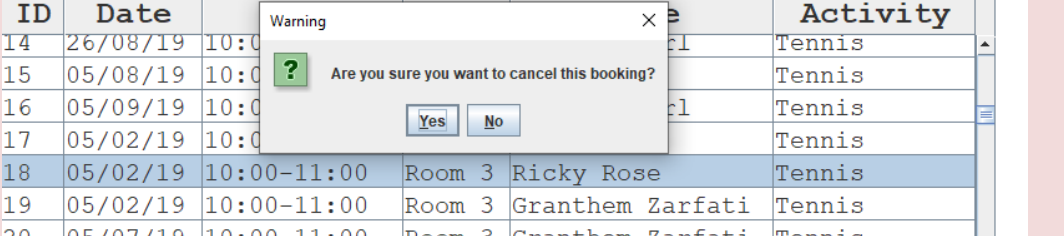
Test Commentary and Refinements Needed

14A – for this test I **deleted a booking in normal conditions by selecting a booking** from the table and pressing the delete button. Once I did this, I was asked to confirm the deletion and another pop-up was displayed **confirming the booking had been cancelled after doing also.** To provide further evidence I displayed the record that was deleted onto the command line. This also shows all other methods being run such as writing to the text file and **repopulating the table with the booking data that doesn't include the deleted index.** The process occurred smoothly with the table being updated very quickly without causing any delays on the system.

14B - for this test I decided to try and run the delete booking process without selecting a record from the table. **As I had my source code open**, I was able to see that if a record was not selected, an **integer variable which holds the value of the index of the record was set to '-1'** which **prompted an error message** to be displayed saying that a record should be selected. This output was simple and **effective but also informs the user on how to delete a booking if they were unsure, making it intuitive.**

14C – **for this test I 'accidentally' pressed the button to delete** after selecting an index within the table. This prompted a **warning message** to be displayed via a pop-up which asked whether I **wanted to cancel the booking I had selected**. As this was done accidentally, I simply pressed the 'no' option and the no other output was produced **allowing me to continue using the system**. This worked **as I was able to prevent myself from deleting** a booking by accident. However, a **possible refinement of this design would be to allow deletes to be undone as novice users may press** the wrong option on the confirmation pop-up causing the booking to be deleted.

Results

Test ID	Evidence
14A	 <p>Deleting Record: 3,30/12/19,10:00-11:00,Room 3,Boone Kidney,Tennis Running writeListToFile For Bookings Running readFromFile Running clearbookingtable Running populatebookingtable</p>
14B	
14C	

Objective # & Description

15: Search booking by name

Discussion of Expected Outcomes

I will be running various functional, performance and validation tests to extensively test this search and see how well it can handle different inputs from the user, without causing any unexpected behaviour which could cause any further problems.

- **Functional test** – for this test I will be searching for an actual name associated with a booking that has been stored within the booking text file. For **this I will expect the correct booking record to be displayed onto a large text area**, that will appear at the bottom of the screen, clearly showing the search results to the user.
- **Validation (empty field)** – for this test I will be running the search by name without entering data into the required text field. The system should respond to this incorrect input and show an error message accordingly, which will be able to show **basic validation for this search, reducing the chance of an incorrect search value being used**.
- **Validation** – for this test I will be searching for a name that **is not an attribute of any of the record stored in the booking file**. With this test I will be able to see whether my conditional statements are working effectively, and I will also be observing **this test alongside my code to ensure that the correct outputs are made**.

Tests Planned

Test ID	Test Type	Purpose	Actual Data Used	Pass / Fail
15A	Functional	Shows that the search functions as intended	Bob Smith	P
15B	Validation	Identify null data input	Null data	P
15C	Validation	Indemnify incorrect data	Tim Smith	P

Test Commentary and Refinements Needed

15A (initial test) - For this test I used a name associated with a **booking that exists within the booking file**, and when I clicked the search button, **a text area that was displayed with the search results**, which showed the booking record related to the name that was searched for. **This worked effectively as intended with the full record being displayed once the search button was pressed which was also clear output to the user making the process user friendly**.

15B – for this test I ran the **search without entering any data** once I tried to search **a pop up was shown informing me to enter a search value into the text field**. This pop up was simple and effective as it identified the error and displayed the suitable message to the user.

15C – for this test I entered a name that doesn't exist in any of the booking's records within the text file. **Once I tried to search with this data the search results displayed a message saying the booking was not found using the name that I had provided**. This therefore showed the search worked effectively but it also responded **relatively quick making it an efficient process too**.

Results

Test ID	Evidence
15A	<div><div>Name<input type="text" value="Bob Smith"/></div><div>Search By Name</div><div>Search Results: 0,05/05/19,10:00-11:00,Room 3,Bob Smith,Tennis 6,03/06/19,10:00-11:00,Room 3,Bob Smith,Tennis 21,05/09/19,10:00-11:00,Room 3,Bob Smith,Tennis</div></div>
15B	<div><div>ID<input type="text"/></div><div>Search By ID</div><div>Date<input type="text"/></div><div>Search By Date</div><div>Name<input type="text"/></div><div>Search By Name</div><div><div>Message</div><div>Error, please enter a search value</div><div>OK</div></div></div>
15C	<div><div>Name<input type="text" value="Tim Smith"/></div><div>Search By Name</div><div>Results:Bookings with Name: Tim Smith not found</div></div>

Objective # & Description

16: search bookings using date

Discussion of Expected Outcomes

for this objective I will be running a functional, validation and performance tests to evaluate this search and how well it can handle different scenarios, without causing any further issues both to for the user or behind the scenes.

- **Functional test** – this will test whether the process can **work effectively under typical correct conditions**, and if it will return the correct value that has been searched for by the user. For this I will expect the **correct booking locations** to be found with the booking of that location being stored into a **string value which will then be displayed onto the screen via a text area pop-up**.
- **Validation** – for this test I will be using **an incorrect date format as the search input**. This will show whether the process is able to identify an incorrect format that may **prompt a malfunctioning system or an incorrect output**. The data entered will **not include '/' but instead in the format "dd-mm-yy"** I expect the system to display an error message in response to any incorrect data with the message specifying to the user, how to provide the correct data.
- **Performance (search with 500 & 1000 bookings)** – similarly to the previous search I will be measuring how well the search performs in terms of time when there are different number of booking records. This will allow me to see the efficiency of this search and whether it is able to cope with **larger amounts of data at a relatively similar speed in comparison to smaller data amounts**. As the search is linear, I expect the search to perform slower when searching **1000 bookings than for 500 as there is less comparisons in the process**.

Tests Planned

Test ID	Test Type	Purpose	Actual Data Used	Pass / Fail
16A	Functional	Shows that the search functions as intended	07/11/19	P
16B	Validation	Identify incorrect format	07-11-19	P
16C	Performance	Search with 500 and 1000 bookings	-	P

Test Commentary and Refinements Needed

16A – for this test I **used typical data when searching for a booking**, and when I clicked the search button, **a text area below the text fields was shown** with a title search result. In these results was the record of all **the bookings that were of the date value** that I tried to search with. This shows that the search for the bookings using its date is effective as it can display **data relevant to the user** for them to find the booking from the results. This is also **user friendly as the text area is large and clear** as it pops up following the user input, therefore makes the system intuitive by providing the user with necessary feedback. However once

possible refinement of this approach would be the way in which the data is displayed, as it is shown how it is written in the text file the data is stored in. **Instead a table could've potentially been used to display this data**, similarly to the way the view bookings page is designed. This would **make the user familiar with the design** making it more user friendly as well as providing headings for the record of data, which may be **important if a user is looking for a specific detail** of the booking such as its time.

16B – for this test I decided to **enter a date in a different format to the** data stored in the text file. When I tried to search using this input, **a pop-up error message** was displayed saying that I should enter **a valid date as well as displaying** the format in which it should be entered. This response from the system is effective as it informs the user that there is an error and specifies how to correct it. This output makes the **system more user friendly especially towards novice users** who may be unfamiliar on how to enter data in the correct format. However, **a possible refinement of this would be to have a calendar on display for** the user to select the date that they are searching for. **This visual input would allow novice users** to be able to use the system more effectively which would make the system **more intuitive**.

16C – for this test I decided to run the search with different amounts of **bookings 500 & 1000**. To create these many bookings, I used the same method I had created for an earlier objective, which would also randomly create a date for each booking when run. Once I searched for a date the search took **approx. 0.007 seconds to complete**. After this I ran the search once there were 1000 bookings stored in the text file. However, when I ran this the execution time for the search **was 0.008 seconds which is almost** the same speed as searching through 500 bookings. This method was effective as it produced the relevant output and was **completed very efficiently, especially for a linear search**. Although the searches did not behave as expected **they did not fail to work effectively** whilst maintain their efficiency at the same time.

Results

Test ID

16A

Evidence

Date

07/11/19

Search By Date

Name

Search By Name

Search Results:

6, 07/11/19, 10:00-11:00, Room 3, Shak Shah, Tennis

26, 07/11/19, 10:00-11:00, Room 3, Shak Shah, Tennis

28, 07/11/19, 10:00-11:00, Room 3, Shak Shah, Tennis

31, 07/11/19, 10:00-11:00, Room 3, Shak Shah, Tennis

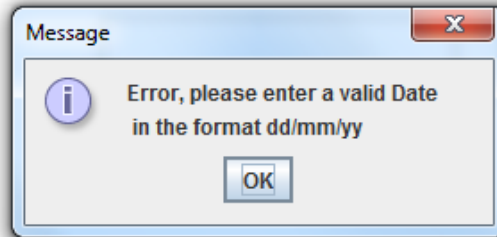
16B

Date

07-11-19

Search By Date

Name



Search By Name

16C

Date

04/06/19

Search By Date

Name

Search By Name

Search Results:

2,04/06/19,10:00-11:00,Room 3,Shak Shah,Tennis
5,04/06/19,10:00-11:00,Room 3,Shak Shah,Tennis
19,04/06/19,10:00-11:00,Room 3,Shak Shah,Tennis
22,04/06/19,10:00-11:00,Room 3,Shak Shah,Tennis
27,04/06/19,10:00-11:00,Room 3,Shak Shah,Tennis

Searching for Booking with Date: 04/06/19
04/06/19

Total execution time: 7.0 Milliseconds OR 0.007 Seconds

Date

01/07/19

Search By Date

Name

Search By Name

805,01/07/19,10:00-11:00,Room 3,Shak Shah,Tennis
895,01/07/19,10:00-11:00,Room 3,Shak Shah,Tennis
905,01/07/19,10:00-11:00,Room 3,Shak Shah,Tennis
910,01/07/19,10:00-11:00,Room 3,Shak Shah,Tennis
999,01/07/19,10:00-11:00,Room 3,Shak Shah,Tennis

Searching for Booking with Date: 01/07/19
01/07/19

Total execution time: 8.0 Milliseconds OR 0.008 Seconds

Objective # & Description

17: Search booking by booking ID

Discussion of Expected Outcomes

I will be running various functional, performance and validation tests to extensively test this search and see how well it can handle different inputs from the user, without causing any unexpected behaviour which could cause any further problems.

- **Functional test** – for this test I will be searching for **an actual ID associated** with a booking that has been stored within the booking text file. For this I will expect the correct booking locations to be found with the location of the booking being **stored into a temporary string variable** which will show all the locations of the records that have the name that was searched as part of the record. **These locations will then be displayed to the user via a pop-up.**
- **Validation (empty field)** – for this test I will be running the **search by ID without** entering data into the required text field. The system should respond to this incorrect input and **show an error message accordingly**, which will be able to show basic validation for this search, **reducing the chance of an incorrect search value being used.**
- **Validation** – for this test I will be searching for an ID that does not exist. This test will be able to effectively show that the correct data is being identified during the search and whether it can identify data that doesn't exist.
- **Validation (String ID)** – for this test I will be **entering an invalid booking ID**, instead using a string as the search value. I expect this to show an error message, as this data would not be a valid data therefore cannot be searched, **therefore the user would be asked to enter a valid ID.**

Tests Planned

Test ID	Test Type	Purpose	Actual Data Used	Pass / Fail
17A	Functional	Shows that the search functions as intended	3	P
17B	Validation	Identify null data input	-	P
17C	Validation	Identify incorrect data	102	P
17D	Validation	Identify a string instead of an integer input	s	P

Test Commentary and Refinements Needed

17A (initial test) - For this test I used a **booking ID found within the file** when searching for the booking related to the ID, and when I **clicked the search button, a pop-up was displayed informing me of the location of the booking under that ID.** This worked well in finding the location of the bookings **yet was not useful to the me as the user**, I would still have to find

the data I was looking for on another screen. Although this was successful in finding the location of the data, it did not successfully find the data the user was searching for therefore had limited functionality which required a refinement in the way it displayed the data to the user.

17A (refinement of search output) – to refine how the search results were **displayed to the user I decided to create a text area that would** be displayed instead of a pop-up window. Within this text area I **also displayed the actual booking record related to the ID** that the user is searching for rather than just its location, which not only improves the output to the user by making more intuitive but also provides the relevant data to the user allowing this search to achieve its goal and maintain its effectiveness. **Using this slight refinement, I re-ran the initial functionality test** using the same data which **worked effectively as intended** with the full record being displayed once the search button was pressed.

17B – for this test I ran the search without entering any data once I tried to **search a pop up was shown informing me to enter a search value into the text field**. This pop up was simple and effective as **it identified the error and displayed the suitable message to the user**. This also **occurred smoothly** after the input without **any delays, ensuring the user can continue**, and enter the correct data.

17C – for this test I entered a **booking ID that doesn't exist**. Once I tried to search with this data, the search was performed as this input was still valid, although the search results displayed a message saying the booking was not found. This **therefore showed the search worked effectively** but it also responded relatively **quick making it an efficient process too**.

17D – for this test **I entered a string value into the search field** and tried to run the search. Once I pressed the search button a pop-up was displayed saying that the data entered was not valid and **prompted me to enter a valid ID**. This validation was effective as it was able to identify the incorrect data type. **As I ran this alongside my source code, I was able to see how the input was converted to an integer** and if the data was unable to do so as it was not an integer, a **Boolean variable I created became false**. A further if statement was then run to check if the Boolean value was false and if it was **would prompt the error message to the user**.

Results

Test ID	Evidence
17A	<div><div><div><div>ID</div><div>3</div></div><div><div>Search By ID</div></div></div><div><div><div>Date</div><div></div></div><div><div>Search By Date</div></div></div><div><div><div>Name</div><div></div></div><div><div>Search By Name</div></div></div></div> <div><div><div>Message</div><div>×</div></div><div><div><div>i</div></div><div>Booking ID: 3 is present at location 3</div><div><div>OK</div></div></div></div> <div><div><div>ID</div><div>3</div></div><div><div>Search By ID</div></div></div> <div><div><div>Date</div><div></div></div><div><div>Search By Date</div></div></div> <div><div><div>Name</div><div></div></div><div><div>Search By Name</div></div></div>


Search Results:

3,30/12/19,10:00-11:00,Room 3,Shak Shah,Tennis

17D

ID

Message

 Error, please enter a suitable search Booking ID

```
boolean integerValue = true;
try
{
    idSearchVal = Integer.parseInt(bookingIDText);
}
catch (NumberFormatException exc)
{
    System.out.println("Error, please enter a search value");
    exc.printStackTrace();
    allTabs.setSelectedIndex(STAFFSEARCHBOOKINGPAGE);
    integerValue = false;
}

if (bookingIDText.isEmpty()==true )
{
    System.out.println("Error, please enter a search value");
    JOptionPane.showMessageDialog(null, ("Error, please enter a search value"));
}
else if (integerValue == false)
{
    System.out.println("Error, please enter a suitable search value");
    JOptionPane.showMessageDialog(null, ("Error, please enter a suitable search Booking ID"));
}
```

8.32 Beta Testing

Test users

- Sonia Parmer (A-level computer science student)
- Usaamah Lakada (A-level computer science student)
- Uwais Patel (general user)
- Mudassir Shah (general user)

Comments on the user friendliness (touch, look and feel) of the system

- Logos show professionalism
- Font is distracting
- Welcome message for login? – unnecessary
- Colour scheme simplistic but appealing
- Large buttons along with large text, prevents any confusion.
- The logos are the same as the actual gym, it looks very realistic.
- The screens are well designed and look like professional websites.

Comments on the system navigation

- Each button moved to the page correctly and each page included the heading on what the page was.
- Navigation was effective and smooth
- Fast button navigation – goes to tab fast making it efficient.
- Consistent size and font of booking making it easy to identify the menu buttons to navigate to the menu.
- It was easy to tell which page does what because of the headings and buttons.

Functionality

- Staff login – the username is case sensitive therefore requires the data to be exactly as stored in the text file. This may not be ideal for all users and may cause more frequent errors.
- Staff login – User login details not removed after successfully logging in -so when a user logs out the next user can easily login using those details, also a security risk.
- Search booking – text area viewing the correct data not wide enough, is annoying as the user must scroll across to see the full results.
- Most validation works effectively although could add more – such as email only requires an '@'
- Staff Make Payments – A£25 displayed why is there an 'A'
- Edit bookings? Could include more features.
- I can search for a booking but not a payment. The bookings in the table do not show how much it cost. Also, could have made a search using its activity as the search.

Efficiency (how fast do things work)

- The add bookings works nice and quick and shows the popups very clearly on the screen when the booking is confirmed.
- The search results are shown very quick when using all the searches
- The filters on the bookings are done very quick and fast, straight after pressing the button.

Other suggestions and improvements

- Could include help popup to understand how to use the system for new users
- The add bookings page buttons and boxes could have been bigger, like the rest of the system.
- There could be some help on how to enter certain data – e.g. how to enter the date properly when searching for a booking.
- The process to filter bookings is not very intuitive as you must select filter then press a button – should happen in one input only.
- The delete button is smaller than the menu button which looks odd.
- The inputs do not validate for extra spaces when entering data, therefore searches may have to match exact data including these spaces. This could cause errors or incorrect data to be shown especially for beginners on the system.

8.4 Objective Summary

Objective #	Objective Description	Achieved?
1	Security – verify staff logins	YES
2	Read from a text file – staff logins	YES
3	Security - Verify customer logins	YES
4	Read from a text file – customer logins	YES
5	Read from a text file – booking file	YES
6	View bookings – display array data into a table	YES
7	Sort/Filter bookings by date	YES
8	Sort/Filter bookings by name	YES
9	Add bookings to the list	YES
10	Write to a file – booking database	YES
11	Write to a file – customer info	YES
12	Write to a file – payments database	YES
13	Calculations to decide price	YES
14	Delete bookings from the database	YES
15	Search for a booking by name	YES
16	Search for a booking by date	YES
17	Search for a booking by booking ID	YES