COS80022 – Software Quality and Testing

Report for Lab 5

Student Name: Arun Ragavendhar Arunachalam Palaniyappan
Student ID: 104837257
Student email: 104837257@student.swin.edu.au
Submission date: 07/04/2024

1. Scheduling test execution

After reading about the given system design plan, it is very clear that the "Customer" is created first with the first priority, under whom one or more "accounts" are created, and under an "account" a "booking" is placed and for a booking a "billing" is generated.
With this type of branching and their respective execution priorities, the below test execution order has been scheduled.

The order of priority while deleting should be from the Inner branch, such that for the same priority index, a booking under an account should be deleted first, then one or more accounts under a customer can be deleted and only then a customer should be deleted.

Another example is that it is better to first amend a booking and then amend a customer when they have the same priority, as from a business point of view, a booking generally involves transactions and money.

And logically, a billing enquiry can be done without any modification or effect on the other test cases and moreover a billing enquiry is made for a booking, so, it is best done before deleting a booking,

After considering technical control flow considerations and logical considerations, the execution order of the test cases is as follows.

Table 1

Execution order	Test Case ID	Description	Priority
1	TC_1045	Login	High
2	TC_1022	Create a Customer	High
3	TC_1033	Create an Account	High
4	TC_1080	Create a Booking	High
5	TC_1036	Amend an Account	High
6	TC_1082	Amend a Booking	Medium
7	TC_1025	Amend a Customer	Medium
8	TC_1055	Billing Enquiry	Low
9	TC_1085	Delete a Booking	Low
10	TC_1038	Delete an Account	Low
11	TC_1028	Delete a Customer	Low

2. Non-functional requirements for the Automated Ticket Issuing system.

All the non-functional requirements have been discussed. Specifically, the Reliability and the response time requirements are explained in depth with a test case example.

Reliability:

- The system should be very reliable, with minimum downtime or faults.
- It should be capable of processing a high number of transactions without crashing or causing severe interruptions

Test Case: Verify a System Recovery After a Simulated Failure

Description: A system failure scenario is to be simulated, such as a server crash, and it is verified whether the ticket-issuing system is able to recover completely without losing transaction data or causing disruptions to the user experience.

Steps:

1. Simulate a server crash or network outage during a transaction process.
2. Monitor system behaviour during the failure to ensure that no data loss or corruption occurs.
3. Restart the system or recover from the failure according to established procedures.
4. Verify that the system resumes normal operation without any noticeable impact on user transactions.
5. Check transaction logs to ensure that all transactions initiated before the failure were successfully completed or appropriately rolled back.

Expected Result: The entire system functionality, should recover smoothly from the simulated failure scenario, ideally within 10 minutes, without any data loss or disruption to user transactions.

.

Response Time:

- The system should respond to user inputs quickly, with little time between picking a game of choice and completing the transaction.
- The average response time for processing transactions should be within a set acceptable range, such as less than 3 seconds.

Test Case: To Measure Transaction Processing Time Under Peak Load

Description: The response time of the ticket-issuing system is to be measured when subjected to peak load conditions to ensure that transactions are processed within an acceptable time frame.

Steps:

1. Simulate peak load conditions by generating a high volume of concurrent transactions.
2. Initiate multiple ticket purchase transactions simultaneously from different user terminals.
3. Record the time taken for each transaction from the moment the user selects a game to the issuance of the ticket.
4. Calculate the average transaction processing time across all simulated transactions.
5. Compare the measured response time against the predefined acceptable range.

Expected Result: The average transaction processing time should fall within the predefined acceptable range, such as under 3 seconds, even under peak load conditions. If the response time exceeds the acceptable range, potential performance bottlenecks and system optimization issues are to be investigated further.

Security:

- The system must protect the security of customer credit card information and personal identifiers.
- It should adhere to appropriate security standards, such as SSL/TLS encryption to prevent unauthorised access or data breaches.

Usability:

- The user interface should be simple to use, especially for those with little technical experience.
- Clear information and prompts should help consumers navigate the ticket purchase procedure.

Performance:

- The system should be able to handle high loads during busy times, such as weekends or holidays.
- Performance testing should be performed to guarantee that the system can sustain acceptable reaction times even under large loads.

Maintainability:

- The system should be built with modular components that can be easily maintained and updated.
- Code should be well-documented and organised to allow for future expansions or alterations.

Compatibility:

- The system should work with a variety of credit card payment processors and hardware devices usually found at theme parks and should be cross platform compatible.