Test Plan: TicketHandler Test Set 1

TicketHandler Test Case Set									
TestCaseId	**Component**	**Priority**	**Description/Test Summary**	**Pre-requisites**	**Test Steps**	**Expected Result**	**Actual Result**	**Status**	**Test Executed By**
PurchaseConfirma	TicketHandler	P3	Test should verify that when a user purchases a ticket, an email confirmation is sent to the email under the user's account detailing the ticket purchase along with a receipt of the purchase.	The client has selected ticket(s), proceeded to checkout, entered payment method, and server has verfied their purchase	1. User is signed into their account. 2. User selects ticket(s) to purchase. 3. User proceeds to checkout. 4. User enters payment method. 5. User's payment method is confirmed.	Purchase is verified and validated by server and receipt is sent to the email under the client's account.	Purchase is verified and validated by server and receipt along with digital copy of tickets are sent to the email under the client's account.	Pass	Carlos Lopez
serverVerify()	TicketHandler, Server	P3	The test should verify that when a user is in the checkout process for buy ticket(s) online, the server confirms the payment method entered by the user, processes and validates the online ticket purchase to the server, and confirms the purchase.	The client has selected their tickets for purchase, has entered their payment method, and confirmed their purchase	1. User is signed into their account. 2. User selects ticket(s) to purchase. 3. User proceeds to checkout. 4. User enters payment method.	The client's ticket purchase is verified by the server by validating the payment method. Returns to PurchaseConfirmation().	The client's ticket purchase is verified by the server by validating the payment method. Returns to PurchaseConfirmation().	Pass	Carlos Lopez

The test set shown above is designed to target both functional and system requirements for the TicketHandler class. The responsibility of this class is to direct the ticket purchasing process given an online customer client and a list of one or more tickets. From the perspective of the client, they are able to select tickets and checkout with their selected ticket(s) and purchase them and expect to receive a purchase confirmation, a copy of their digital tickets, and a receipt of their purchase.

What the test set does is test the PurchaseConfirmation and serverVerify() components of the class. The first test case, involving purchase confirmation, verifies that after the client, being the online customer, has their online purchase verified and is sent a copy of their digital ticket(s) along with a receipt of their purchase to the email either filed under the client's account or the email address used when checking out without an account. Then, the prerequisites for this test case are that the client has proceeded to checkout with their selected movie ticket(s) and has entered a payment method. The program will verify if the payment method is valid and then run a server verify leading to the server confirmation of the purchase. Finally, the test case concludes with the email confirmation being sent to the client detailing the purchase with a copy of the online digital ticket(s) and a receipt. The second test case involves the server verification of ticket purchasing mentioned in the previous test case. This is a system case as the system is dependent on the server to properly function. What this function does is verify and validate the client's payment methods for their ticket purchase and update the remaining available seats/tickets in the database.

Test Plan: TicketHandler Test Set 2

Theatre Test Cases									
TestCaseId	Component	Priority	Description/Test Summary	Pre-requisites	Test Steps	Expected Result	Actual Result	Status	Test Executed By
ShowAvailibleTicketsClient	Ticket Class Theatre Class WebsiteUI OnlineClient	P1	Verify that the website instance allows the client to send a request to view availible tickets, and that the Ticket Handler class contains the information that the client requested	A client is logged in and online, user requests to show movies at a specific theatre location, data is relayed to Ticket Handler	1. Enter the theatre website url (assuming the site is hosted) 2. Initialize login credentials for an Online Client (payment information not required) 3. Select a random theatre and request to see the tickets for a random movie	TicketHandler class fields should match those sent by the client.	Under perfect conditions, the TickerHandler class fields matched the data sent by the client, and the corresponding theatre and movie are properly formatted	Pass	Ryan Bouzan
SendPurchaseTicketClient	Ticket Class Theatre Class clientTicketHandler TickerHandler OnlineClient	P1	Verify that when a ticket is purchased on the client side, the request is sent to the server. The request, which includes the client's ticket selection and puchase information + a puchaseID, should be initiated from the OnlineClient Class, then be passed on to the TickerHandler. This test case does NOT verify if the server updates the records or if the payment method is valid.	A client is logged in and online, and the website is in the state where it is showing availible tickets to purchase at a specific theatre	1. Initialize payment information, login credentials, and any other neccesary personal information such as email and name 2. Create an instance of an online client using the credentials 3. Send purchase request using website UI	TicketHandler class fields should match those sent by the client.	Under perfect conditions, the TickerHandler class fields matched the data sent by the client, and the purchaseID sent from the client matched the one created by the TickerHandler	Pass (unsecure)	Ryan Bouzan
ShowAvailibleTicketsServer	Ticket Class Theatre Class WebsiteUI OnlineClient	P1	Verify that the website instance allows the client to send a request to view availible tickets, and that the Ticket Handler class contains the information that the client requested	A client is logged in and online, user requests to show movies at a specific theatre location, data is relayed to Ticket Handler	1. Enter the theatre website url (assuming the site is hosted) 2. Initialize login credentials for an Online Client (payment information not required) 3. Select a random theatre and request to see the tickets for a random movie	TicketHandler class fields should match those sent by the client.	Under perfect conditions, the TickerHandler class fields matched the data sent by the client, and the corresponding theatre and movie are properly formatted	Pass	Ryan Bouzan
SendPurchaseTicketServer	Ticket Class Theatre Class clientTicketHandler TickerHandler OnlineClient	P1	Verify that when a ticket is purchased on the client side, the request is sent to the server. The request, which includes the client's ticket selection and puchase information + a puchaseID, should be initiated from the OnlineClient Class, then be passed on to the TickerHandler. This test case does NOT verify if the server	A client is logged in and online, and the website is in the state where it is showing availible tickets to purchase at a specific theatre	1. Initialize payment information, login credentials, and any other neccesary personal information such as email and name 2. Create an instance of an online client using the credentials 3. Send purchase request using website UI	TicketHandler class fields should match those sent by the client.	Under perfect conditions, the TickerHandler class fields matched the data sent by the client, and the purchaseID sent from the client matched the one created by the TickerHandler	Pass	Ryan Bouzan

The test set shown above is designed to target all three granularities: unit, functional, and system requirements. This is accomplished by carrying out two pairs of tests: requesting to view available tickets and making a purchase, both of which are common actions that a client would take. At the small scale, these are unit tests because they test specific functionality of the OnlineClient and TicketHandler classes. In terms of functional tests, multiple classes are used, including the Ticket, Theatre, TicketHandler, OnlineClient, and WebsiteUI classes. Each of these has an interconnected role in handling the current instance of OnlineClient. Specifically, all of these classes help to manage data related to theatre information, along with data related to client credentials/information. Finally, the system requirements are thoroughly tested because in order to effectively display the data, the client and server must have a reliable and secure transfer of information. As explained in the Software Design Specification, the client and server do not directly interact and are instead mediated by the TicketHandler class. This helps to ensure that the data and requests passed between the client and server, including sensitive payment information, are both accurate and valid. This kind of interaction is an example of core system functionality, because keeping the server records organized and secure will go

a long way towards maintaining the integrity of the application. The test that is highlighted in yellow is insecure because there is not currently a way to verify the payment information is 100% accurate or that the client did not attempt to exploit the system using tactics like SQL Injection to avoid payment. Since this represents a more critical functionality than other actions, new attributes should be added to either the OnlineClient, TicketHandler, or Server class that guard against possible flaws.

Updates to the Software Design Specification

In order to clarify how the client and server interact, new methods and fields were added to most of the classes that are related to the "OnlineClient" class. The changes are laid out in detail below:

1) **WebsiteUI**
    a) Added the JavascriptElements field to handle physical user input requests
    b) Added cascading methods for location, theaters, movies, and tickets
2) **OnlineClient**
    a) Specified RSA encryption for the client's password, and introduced a session ID
3) **TicketHandler**
    a) Added session ID that will correspond to OnlineClient's session ID
4) **Server**
    a) Added ticket, movie, and order records. These will be implemented using a relational SQL database
    b) Added a client record database, implemented using a non-relational SQL database
5) **Identification** (abstract)
    a) Fixed the access modifiers so that OnlineClient can access them
    b) Reorganized the fields so that it matches the design paradigm