TestCaseID	Component	Priority	Description/Test Summary	Pre-requisites	Test Steps	Expected Result	Actual Result	Status	Test Executed By

UNIT-001	Price Calculator (discounts & rounding)	High	Checks that discounts and rounding are applied correctly for different ticket types	Pricing rules loaded;
				base price = $12;
				weekday = Wednesday
					Call price() for
					REGULAR
					SENIOR
					STUDENT
					VETERAN
						REGULAR = $12.00
						SENIOR = $9.00
						STUDENT = $9.00
						VETERAN = $9.00
							REGULAR=$12;
							SENIOR=$9;
							STUDENT=$9;
							VETERAN=$9
							(rounded to 2 decimals)
								Pass	Shawyan

FUNC-001	Payment Webhook Idempotency	High	Makes sure duplicate payment webhooks don't cause duplicate orders or tickets	Order O123 created in PENDING; sandbox processor ready	Send the same "/payments/hook" payload twice	Order set to PAID only once;
						one payment record;
						one audit log entry;
						no duplicate tickets
							Order was set to PAID once;
							one payment record stored;
							one audit log created;
							no duplicate tickets found
								Pass	Shawyan

SYS-001	Refund Ticket Invalidation at Gate	High	Verifies that refunded tickets cannot be used at the gate	Paid order O987 exists with ticket T555; admin 2FA enabled; gate scanner test endpoint active	Refund order O987; scan QR code for ticket T555	Order and payment marked REFUNDED; audit log entry created; gate scan rejects QR as INVALID/REVOKED	Order and payment marked as REFUNDED; audit log entry stored; gate scan rejected QR as INVALID/REVOKED	Pass	Shawyan
UNIT-002	Order History	High	Display order history of an account	The database contains the account and an order history to provide	Call view_orders()	A list of the orders for the account	The list of the orders for the account	Pass	Samuel
FUNC-002	Account login	High	Verify the account login process functions properly	An account is fully set up	1. Enter a validusername and password 2. Enter a valid username and invalid password 3. Enter an invalid username	A successful login followed by 2 unsuccessful logins	A successful login attempt followed by 2 unsuccessful logins	Pass	Samuel
SYS-002	Ticket purchasing	High	Go through the ticket purchasing process to ensure it works	the ticket and qr code creation is finished, and the payment processors are implemented	Go to the website, select a movie and showing, then confirm payment has gone through	A ticket ID will be created and a QR code will be generated	A ticket ID was createdand a QR code was generated	Pass	Samuel
FUNC-003	Database retrival	High	Check that the sales data is properly retrieved from the database	The database needs to be set up and have data to be retrieved	Call get_sales_data(10/01/25, 10/23/25)	The data from the database will be retrieved and displayed	The data was properly retrieved and displayed	Pass	Samuel
UNIT-003	Showtime get_ticket_count	High	Verify that get_ticket_count returns the number of true entries in the seats array	Three instances of Showtime must be initialized with different sample seat data	Call get_ticket_count on the three instances of Showtime and check what is returned	The function returns the expected number of tickets for each set of sample data supplied	The function returned the expected number of tickets for each sample dataset	Pass	Gregory Larson
FUNC-004	Showtime check_seat_availability	High	Verify that check_seat_availability under Showtime class correctly changes the seat flag for the specified seat	A testing database must exist to pull data from for validation	1: Create an instance of Showtime with data from the testing database. 2: Call check_seat_availability, with this list of parameters: valid index with a False flag, valid index with a True flag, invalid index, incorrect data type(s).	The database has the valid index flags changed, and the invalid parameters change nothing and throw errors.	The database had changed its valid index flags to their complements. The invalid parameters did nothing and threw errors.	Pass	Gregory Larson
SYS-003	Editing a Showtime via Admin Page	High	Verify that the database is properly modified when an administrator edits an existing Showtime on the database.	A testing database must exist to pull data from for validation. In addition, the Admin Page must be verified and login functionality works as intended.	1: On the Admin Page, make a single modification and confirm it. 2: Check the database to confirm modifications have been pushed. 3: Repeat for each possible modification individually. 4: Repeat for multiple modifications in a single confirmation.	The database recieves each modification and updates itself upon each confirmation, including multiple modifications in a single confirmation.	The database did recieve each modification and updated near-instantly. Upon recieving a confirmation for multiple modifications at once, only the leftmost modifications was passed to the database.	Partial Failure	Gregory Larson