Università degli Studi di Salerno
Corso di Ingegneria del Software

SoundShelf
Test Execution Report
Versione 1.0

Data: 09/01/2025

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Marco Della Greca	0512116959
Luca Mastino	0512108160
Giuseppe Caiazzo	0512104992
Michele Quaglia	0512118399

Scritto da:	Intero Team

Revision History

Data	Versione	Descrizione	Autore
09/01/2025	1.0	Stesura Test Execution Report	Della Greca Marco, Mastino Luca

Indice

1. INTRODUZIONE

1.1. Scopo del documento

Questo **Test Execution Report** ha l'obiettivo di documentare in dettaglio l'esecuzione dei test effettuati sul software SoundShelf, evidenziando i risultati ottenuti e le eventuali discrepanze rispetto alle aspettative.

Il report include tutte le informazioni necessarie per valutare l'efficacia dei test e per prendere decisioni informate riguardo alle modifiche necessarie al software o al processo di testing.

1.2. Obiettivi del test

Gli obiettivi principali di questo testing sono i seguenti:

- **Verifica della funzionalità**: Assicurarsi che il software soddisfi i requisiti funzionali e operi come previsto.
- **Identificazione degli errori**: Rilevare bug e malfunzionamenti all'interno del sistema.
- **Verifica della performance**: Garantire che il sistema risponda ai requisiti di performance in termini di velocità, stabilità e reattività.
- **Valutazione della robustezza**: Testare la capacità del sistema di gestire scenari imprevisti, errori e carichi di lavoro intensi.

1.3. Contesto e Requisiti

Il software testato è stato progettato per soddisfare specifici requisiti di funzionalità, usabilità, performance e sicurezza. La fase di testing si è concentrata su scenari che replicano l'ambiente di produzione, con l'obiettivo di validare il corretto funzionamento del sistema in condizioni reali.
I requisiti del sistema includono:

- **Funzionalità**: Il sistema deve garantire l'esecuzione corretta di operazioni di gestione degli ordini, interazione con il database, e gestione delle richieste di supporto.
- **Sicurezza**: Il software deve proteggere i dati sensibili degli utenti e prevenire accessi non autorizzati.
- **Usabilità**: L'interfaccia utente deve essere intuitiva e facile da navigare.
- **Performance**: Il sistema deve rispondere velocemente anche sotto carico elevato.

1.4. Ambito del test

I test descritti in questo report si concentrano principalmente sulle seguenti aree del sistema:

- **Test di funzionalità**: Verifica della corretta esecuzione delle operazioni principali del sistema.
- **Test di integrazione**: Verifica dell'integrazione tra moduli diversi, in particolare per quanto riguarda la gestione degli ordini e la gestione delle richieste di supporto.
- **Test di regressione**: Assicurarsi che le modifiche al sistema non abbiano introdotto errori in funzionalità precedentemente verificate.
- **Test di performance**: Simulazione di scenari di carico per testare la reattività del sistema sotto stress.

1.5.Strategia di test

La strategia di test adottata per il progetto è di tipo **Bottom up**. Questo approccio è stato scelto per permettere il testing preliminare di tutti quanti i sottosistemi totalmente indipendenti da altri, prima di passare per livelli a quelli superiori.

2. TEST ITEM TRANSMITTIAL REPORT

2.1.Identificazione dei Software Item

Ogni sottosistema testato è descritto di seguito, con le relative informazioni di identificazione e le specifiche pertinenti.

2.1.1. Sottosistema Utente

Nome del Sottosistema: Gestione Utente
ID: UT-001
Descrizione: Gestisce la registrazione, l'autenticazione, la gestione delle informazioni dell'utente e la visualizzazione dei dati relativi agli utenti.
Versione: 1.0
Responsabile del Test: Intero Team

2.1.2. Sottosistema Rimborso

Nome del Sottosistema: Gestione Rimborso
ID: RM-002
Descrizione: Gestisce la gestione dei rimborsi per i singoli articoli e l'elaborazione delle richieste.
Versione: 1.0
Responsabile del Test: Intero Team

2.1.3. Sottosistema Prodotto

Nome del Sottosistema: Gestione Catalogo Prodotti
ID: PD-003
Descrizione: Gestisce il catalogo di prodotti (vinili, dischi), inclusi l'aggiunta, la modifica e l'eliminazione di prodotti.
Versione: 1.0
Responsabile del Test: Intero Team

2.1.4. Sottosistema Recensione

Nome del Sottosistema: Gestione Recensioni
ID: RC-004
Descrizione: Permette agli utenti di scrivere, leggere e moderare recensioni sui prodotti acquistati.
Versione: 1.0
Responsabile del Test: Intero Team

2.1.5. Sottosistema Ordine

Nome del Sottosistema: Gestione Ordini
ID: OD-005
Descrizione: Gestisce la creazione, la modifica e il tracciamento degli stati degli ordini.
Versione: 1.0
Responsabile del Test: Intero Team

2.1.6. Sottosistema Supporto

Nome del Sottosistema: Gestione Supporto
ID: SP-006
Descrizione: Gestisce le richieste di supporto e le risposte da parte del team di supporto.
Versione: 1.0
Responsabile del Test: Intero Team

2.2. Riferimenti

Tutti i sottosistemi sono stati definiti nel documento di System Design.
Tutti i piani di test sono stati definiti nei Test Plan e nel documento dei casi di Test.

3. TEST LOG

3.1. TC1

Test ID	Descrizione
Caso di Test	TC1 - Verifica della corretta funzionalità del login dell'utente, includendo i seguenti scenari: 1. Login con credenziali corrette. 2. Login con password errata o non trovata. 3. Login con e-mail errata o non trovata.
Modulo/Componente Testato	Login Control
Responsabile del Test	Intero team
Data del Test	04/01/2025
Ambiente di Test	Ambiente di sviluppo, Eclipse IDE
Prerequisiti	L'utente deve essere registrato nel sistema.
Strumenti di Test	JUnit 5, Mockito, Servlet Test Environment, Database simulato con mock.

Test	Descrizione del test	Risultato Atteso	Risultato Effettivo	Stato del Test	Deviazioni Riscontrate
testLogin Success	Verifica che l'utente venga autenticato correttamente e rediretto alla home page se le credenziali sono corrette.	Autenticazione corretta dell'utente con e-mail e password. La sessione è aggiornata con l'utente autenticato. L'utente viene rediretto alla home.	Autenticazione corretta dell'utente. Sessione aggiornata correttamente. Redirezione alla home effettuata.	Passed	Nessuna deviazione riscontrata.
testLogin WrongPassword	Verifica che l'utente venga notificato con un errore quando la password inserita non corrisponde.	Il sistema deve restituire un messaggio di errore "Email o password non validi.". L'utente è	Messaggio di errore restituito correttamente. Redirezione alla pagina di errore eseguita	Passed	Nessuna deviazione riscontrata.

		reindirizzato alla pagina di errore.	correttamente.		
testLoginEmailNotFound	Verifica che l'utente venga notificato con un errore quando l'email non esiste nel sistema.	Il sistema deve restituire un messaggio di errore "Email o password non validi.". L'utente è reindirizzato alla pagina di errore.	Messaggio di errore restituito correttamente. Redirezione alla pagina di errore eseguita correttamente.	Passed	Nessuna deviazione riscontrata.

3.2. TC2

Test ID	Descrizione
Caso di Test	TC2 - Verifica della corretta funzionalità dell'acquisto dell'utente, includendo i seguenti scenari: 1. Utente non autenticato con carrello non vuoto. 2. Utente autenticato con carrello non vuoto e nuovo indirizzo di spedizione valido. 4. Utente autenticato con carrello non vuoto e un indirizzo di spedizione salvato valido. 3. Utente autenticato con prodotto non disponibile.
Modulo/Componente Testato	Acquisto Control
Responsabile del Test	Intero team
Data del Test	04/01/2025
Ambiente di Test	Ambiente di sviluppo, Eclipse IDE
Prerequisiti	L'utente deve essere registrato nel sistema.
Strumenti di Test	JUnit 5, Mockito, Servlet Test Environment, Database simulato con mock.

Test	Descrizione del test	Risultato Atteso	Risultato Effettivo	Stato del Test	Deviazioni Riscontrate
testUnauthenticatedUser	Verifica che un utente non autenticato venga notificato con un errore quando tenta di procedere con l'acquisto con il carrello non vuoto.	Il sistema deve restituire un messaggio di errore "Per procedere con l'acquisto è necessario essere autenticati." e reindirizzare alla pagina di errore.	Messaggio di errore restituito correttamente. Redirezione alla pagina di errore eseguita correttamente.	Passed	Nessuna deviazione riscontrata.
testAuthenticatedUserWithNewAddress	Verifica che un utente autenticato, con un carrello non vuoto e un nuovo indirizzo di spedizione valido, possa completare l'acquisto e ricevere la conferma dell'ordine.	Il sistema deve aggiungere l'ordine nel database e redirigere correttamente l'utente alla pagina di conferma acquisto.	Acquisto completato correttamente. Ordine aggiunto nel database, redirezione alla pagina di conferma eseguita correttamente.	Passed	Nessuna deviazione riscontrata.
testProductNotAvailable	Verifica che l'utente venga notificato con un errore quando tenta di acquistare un prodotto non disponibile in quantità sufficiente.	Il sistema deve restituire un messaggio di errore "Il prodotto [Nome Prodotto] non è disponibile nella quantità richiesta."	Messaggio di errore restituito correttamente. Indicazione della quantità non disponibile eseguita correttamente.	Passed	Nessuna deviazione riscontrata.
testShippingAddressValidation	Verifica che l'utente venga notificato con un errore quando tenta di acquistare un prodotto avendo inserito un nuovo indirizzo di spedizione non valido.	Il sistema deve restituire un messaggio di errore " L'indirizzo di spedizione non è valido. Assicurati che contenga almeno 10 caratteri."	Messaggio di errore restituito correttamente. Indicazione della quantità non disponibile eseguita correttamente.	Passed	Nessuna deviazione riscontrata.

3.3. TC3

Test ID	Descrizione
Caso di Test	TC3 - Verifica della corretta funzionalità del funzionamento del Carrello, in particolare sugli scenari di: 1. Aggiunta di prodotti. 2. Modifica di quantità di un prodotto con quantità valida. 3. Modifica di quantità di un prodotto con quantità invalida. 4. Rimozione di prodotti.
Modulo/Componente Testato	Carrello Control
Responsabile del Test	Intero team
Data del Test	04/01/2025
Ambiente di Test	Ambiente di sviluppo, Eclipse IDE
Prerequisiti	L'utente deve essere registrato nel sistema.
Strumenti di Test	JUnit 5, Mockito, Servlet Test Environment, Database simulato con mock.

Test	Descrizione del test	Risultato Atteso	Risultato Effettivo	Stato del Test	Deviazioni Riscontrate
testInvalidQuantityUpdate	Verifica il comportamento del carrello quando l'utente tenta di aggiornare la quantità a un valore non valido (-1).	Il sistema non permette il cambio di quantità, con un errore "Quantità non valida" e la modifica non viene effetuata.	Messaggio di errore restituito correttamente.	Passed	Nessuna deviazione riscontrata.
testValidQuantityUpdate	Verifica che l'utente possa aggiornare correttamente la quantità di un prodotto a un valore valido (3).	La quantità del prodotto nel carrello viene aggiornata correttamente, e l'utente viene rediretto alla pagina del carrello.	Quantità aggiornata correttamente. Redirezione alla pagina del carrello eseguita correttamente.	Passed	Nessuna deviazione riscontrata.

testAddProductToCart	Verifica che l'utente possa aggiungere un prodotto al carrello.	Il prodotto viene aggiunto correttamente al carrello, e l'utente viene rediretto alla pagina del carrello.	Prodotto aggiunto correttamente. Redirezione alla pagina del carrello eseguita correttamente.	Passed	Nessuna deviazione riscontrata.
testRemoveProductFromCart	Verifica che l'utente possa rimuovere un prodotto dal carrello.	Il prodotto viene rimosso correttamente dal carrello, e l'utente viene rediretto alla pagina del carrello.	Prodotto rimosso correttamente. Redirezione alla pagina del carrello eseguita correttamente.	Passed	Nessuna deviazione riscontrata.

3.4. TC4

Test ID	Descrizione
Caso di Test	TC4 - Verifica della corretta funzionalità del funzionamento del processo di Recensioni, con i seguenti scenari 1. Aggiunta di una recensione. 2. Aggiunta di una recensione con descrizione mancante. 3. Aggiunta di una recensione con voto mancante. 4. Aggiunta di una recensione con descrizione invalida. 5. Aggiunta di una recensione con voto invalido.
Modulo/Componente Testato	Recensione Control
Responsabile del Test	Intero team
Data del Test	04/01/2025
Ambiente di Test	Ambiente di sviluppo, Eclipse IDE
Prerequisiti	L'utente deve essere registrato nel sistema.
Strumenti di Test	JUnit 5, Mockito, Servlet Test Environment, Database simulato con mock.

Test	Descrizione del test	Risultato Atteso	Risultato Effettivo	Stato del Test	Deviazioni Riscontrate
testValidReview	Verifica che una recensione valida venga processata correttamente.	Recensione inviata correttamente e richiesta inoltrata alla vista di successo.	Recensione inviata correttamente e richiesta inoltrata alla vista di successo.	Passed	Nessuna deviazione riscontrata.
testMissingVoto	Verifica che il sistema segnali l'assenza del voto.	Il sistema visualizza il messaggio di errore "Compilare tutti i campi obbligatori".	Messaggio di errore "Compilare tutti i campi obbligatori" visualizzato correttamente,	Passed	Nessuna deviazione riscontrata.
testInvalidVoto	Verifica che il sistema segnali un voto fuori dall'intervallo valido (0-5).	Il sistema visualizza il messaggio di errore "Voto non valido. Inserire un valore tra 0 e 5".	Messaggio di errore "Voto non valido. Inserire un valore tra 0 e 5" visualizzato correttamente.	Passed	Nessuna deviazione riscontrata.
testMissingDescrizione	Verifica che il sistema segnali l'assenza della descrizione.	Il sistema visualizza il messaggio di errore "Compilare tutti i campi obbligatori".	Messaggio di errore "Compilare tutti i campi obbligatori" visualizzato correttamente.	Passed	Nessuna deviazione riscontrata.
testInvalidDescrizione	Verifica che il sistema segnali una descrizione non valida.	Il sistema visualizza un messaggio di errore "Descrizione non valida. Inserire una descrizione adeguata"	Messaggio di errore "Descrizione non valida. Inserire una descrizione adeguata" visualizzato correttamente.	Passed.	Nessuna deviazione riscontrata.

3.5. TC5

Test ID	Descrizione
Caso di Test	TC5 - Verifica della corretta funzionalità del funzionamento della ricerca prodotti, con i seguenti scenari 1. Ricerca con parametri validi. 2. Ricerca senza parametri 3. Errore nella ricerca.
Modulo/Componente Testato	Ricerca Control
Responsabile del Test	Intero team
Data del Test	04/01/2025
Ambiente di Test	Ambiente di sviluppo, Eclipse IDE
Prerequisiti	L'utente deve essere registrato nel sistema.
Strumenti di Test	JUnit 5, Mockito, Servlet Test Environment, Database simulato con mock.

Test	Descrizione del test	Risultato Atteso	Risultato Effettivo	Stato del Test	Deviazioni Riscontrate
testwithValid Parameters	Verifica che la ricerca restituisca correttamente i prodotti quando vengono forniti parametri validi.	Il sistema restituisca i prodotti correttamente come JSON e richiesta inoltrata senza errori.	Prodotti restituiti correttamente come JSON e richiesta inoltrata senza errori.	Passed	Nessuna deviazione riscontrata.
testwithNoParameters	Verifica che la ricerca restituisca correttamente tutti i prodotti quando non vengono forniti parametri.	Il sistema restituisce correttamente tutti i prodotti come JSON e richiesta inoltrata senza errori.	Tutti i prodotti restituiti correttamente come JSON e richiesta inoltrata senza errori.	Passed	Nessuna deviazione riscontrata.
testwithError InDAO	Verifica che la ricerca gestisca l'errore correttamente.	Il sistema imposta lo stato della risposta su 500 e messaggio di errore restituito come JSON.	Stato risposta impostato su 500 e messaggio di errore restituito come JSON.	Passed	Nessuna deviazione riscontrata.

3.6. TC6

Test ID	Descrizione
Caso di Test	TC6 - Verifica della corretta funzionalità del funzionamento dell'invio di richieste di rimborso, con i seguenti scenari 1. Utente non loggato. 2. Codice ordine errato. 3. Ordine non trovato. 4. Richiesta compilata correttamente e inviata correttamente. 5. Richiesta compilata con parametri invalidi. 6. Errore di database sul salvataggio della richiesta.
Modulo/Componente Testato	Invia Richiesta Rimborso Control
Responsabile del Test	Intero team
Data del Test	04/01/2025
Ambiente di Test	Ambiente di sviluppo, Eclipse IDE
Prerequisiti	L'utente deve essere registrato nel sistema.
Strumenti di Test	JUnit 5, Mockito, Servlet Test Environment, Database simulato con mock.

Test	Descrizione del test	Risultato Atteso	Risultato Effettivo	Stato del Test	Deviazioni Riscontrate
testNotLoggedIn	Verifica che l'invio gestisca correttamente l'assenza di una sessione utente attiva.	Il sistema mostra il messaggio di errore e reindirizza alla pagina di errore.	Messaggio di errore mostrato e reindirizzamento alla pagina di errore.	Passed	Nessuna deviazione riscontrata.
testInvalidDetailCode	Verifica che l'invio gestisca un codice di dettaglio ordine non valido.	Il sistema mostra il messaggio "Parametri non validi." e reindirizza alla pagina di errore.	Messaggio "Parametri non validi." mostrato e reindirizzamento alla pagina di errore.	Passed	Nessuna deviazione riscontrata.
testDetailNotFound	Verifica che l'invio gestisca la mancata individuazione del dettaglio ordine.	Il sistema mostra il messaggio "Dettagli prodotto non trovati." e reindirizza alla pagina di errore.	Messaggio "Dettagli prodotto non trovati." mostrato e reindirizzamento alla pagina di errore.	Passed	Nessuna deviazione riscontrata.

testSuccess	Verifica che l'invio funzioni correttamente con tutti i parametri validi.	Il sistema invia e salva correttamente la richiesta e reindirizza allo storico ordini.	Richiesta salvata e inviata e reindirizzamento allo storico ordini.	Passed	Nessuna deviazione riscontrata.
testInvalidParameters	Verifica che l'invio gestisca correttamente parametri null o invalidi.	Il sistema visualizza il messaggio di errore "Parametri non validi." e reindirizza alla pagina di errore.	Messaggio "Parametri non validi." mostrato e reindirizzamento alla pagina di errore.	Passed	Nessuna deviazione riscontrata.
testSaveRequestError	Verifica che l'invio gestisca correttamente un errore durante il salvataggio della richiesta di rimborso.	Il sistema visualizza il messaggio "Errore nel salvataggio della richiesta di rimborso." e reindirizza alla pagina di errore.	Messaggio "Errore nel salvataggio della richiesta di rimborso." mostrato e reindirizzamento alla pagina di errore.	Passed	Nessuna deviazione riscontrata.

3.7. TC7

Test ID	Descrizione
Caso di Test	TC7 - Verifica della corretta funzionalità della modifica del prodotto, con i seguenti scenari 1. Prodotto trovato. 2. Prodotto non trovato. 3. ID del prodotto non valido. 4. Modifica prodotto riuscita. 5. Prodotto non trovato durante la modifica. 6. Errore di database durante la modifica del prodotto. 7. Parametri di input non validi durante la modifica del prodotto.
Modulo/Componente Testato	Modifica Prodotto Control
Responsabile del Test	Intero team
Data del Test	04/01/2025
Ambiente di Test	Ambiente di sviluppo, Eclipse IDE
Prerequisiti	L'utente deve essere registrato nel sistema e deve essere il Gestore.

Strumenti di Test	JUnit 5, Mockito, Servlet Test Environment, Database simulato con mock.

Test	Descrizione del test	Risultato Atteso	Risultato Effettivo	Stato del Test	Deviazioni Riscontrate
testProductFound	Verifica che il prodotto venga trovato e restituito correttamente.	Il prodotto viene mostrato nella pagina e il controllo avvia il reindirizzamento.	Il prodotto viene restituito correttamente e il controllo invia il reindirizzamento.	Passed	Nessuna deviazione riscontrata.
testProductNotFound	Verifica che venga gestito il caso in cui il prodotto non esiste.	Il sistema mostra un messaggio di errore "Product not found".	Messaggio "Product not found" visualizzato correttamente e reindirizzamento.	Passed	Nessuna deviazione riscontrata.
testInvalidProductId	Verifica che venga gestito correttamente un ID prodotto non valido.	Il sistema mostra un messaggio di errore "Product code is required".	Messaggio "Product code is required" visualizzato correttamente e reindirizzamento.	Passed	Nessuna deviazione riscontrata.
testProductUpdated	Verifica che la modifica del prodotto venga effettuata correttamente.	Il prodotto viene aggiornato e il sistema invia un redirect allo storico.	Il prodotto viene aggiornato e redirige correttamente allo storico ordini.	Passed	Nessuna deviazione riscontrata.
testProductNotFoundModifica	Verifica che venga gestito il caso in cui il prodotto non esista.	Il sistema visualizza un messaggio di errore "Error updating product".	Messaggio "Error updating product" visualizzato correttamente e reindirizzamento.	Passed	Nessuna deviazione riscontrata.
testSQLException	Verifica che venga gestito un errore di database durante la modifica.	Il sistema visualizza il messaggio "Error updating product" e reindirizza.	Messaggio "Error updating product" visualizzato correttamente e reindirizzamento.	Passed	Nessuna deviazione riscontrata.

testInvalidParameters	Verifica che venga gestito il caso di parametri non validi.	Il sistema visualizza un messaggio di errore "Error updating product".	Messaggio "Error updating product" visualizzato correttamente e reindirizzamento.	Passed	Nessuna deviazione riscontrata.

3.8. TC8

Test ID	Descrizione
Caso di Test	TC8 - Verifica della corretta funzionalità della rimozione del prodotto, con i seguenti scenari 1. Prodotto trovato e rimozione riuscita. 2. Prodotto non trovato. 3. ID del prodotto non valido. 4. ID prodotto mancante. 5. Errore di database durante la rimozione del prodotto.
Modulo/Componente Testato	Rimuovi Prodotto Control
Responsabile del Test	Intero team
Data del Test	04/01/2025
Ambiente di Test	Ambiente di sviluppo, Eclipse IDE
Prerequisiti	L'utente deve essere registrato nel sistema e deve essere il Gestore.
Strumenti di Test	JUnit 5, Mockito, Servlet Test Environment, Database simulato con mock.

Test	Descrizione del test	Risultato Atteso	Risultato Effettivo	Stato del Test	Deviazioni Riscontrate
testProductExists	Verifica che, se il prodotto esiste, venga rimosso dal database e venga effettuato il redirect alla pagina di gestione catalogo.	Il prodotto deve essere rimosso correttamente, e il sistema deve eseguire un redirect alla pagina del catalogo.	Il prodotto è rimosso correttamente e il redirect è stato eseguito correttamente.	Passed	Nessuna deviazione riscontrata.

testProductNotFound	Verifica che se il prodotto non viene trovato nel database, venga visualizzato un messaggio di errore e venga effettuato un forward alla pagina di errore.	Il sistema mostra il messaggio di errore "Product not found" e deve essere fatto un forward alla pagina di errore.	Messaggio di errore impostato correttamente e il forward è stato eseguito correttamente.	Passed	Nessuna deviazione riscontrata.
testSQLExceptionOnDelete	Verifica che, in caso di errore SQL durante la rimozione di un prodotto, venga mostrato un messaggio di errore e venga effettuato un forward alla pagina di errore.	Il sistema solleva un'eccezione SQLException che deve essere gestita, con la visualizzazione del messaggio di errore "Error removing product from the database" e il forward alla pagina di errore.	L'eccezione SQLException è stata gestita correttamente, il messaggio di errore è stato impostato e il forward è stato eseguito.	Passed	Nessuna deviazione riscontrata.
testProductCodeMissing	Verifica che, se il parametro productId è mancante, venga visualizzato un messaggio di errore.	Il messaggio di errore "Product code is required" deve essere visualizzato e il forward alla pagina di errore deve avvenire.	Il messaggio di errore è stato impostato correttamente e il forward è stato eseguito correttamente.	Passed	Nessuna deviazione riscontrata.
testInvalidProductCodeFormat	Verifica che, se il parametro productId ha un formato non valido, venga visualizzato un messaggio di errore.	Il messaggio di errore "Invalid product code format" deve essere visualizzato e il forward alla pagina di errore deve avvenire.	Il messaggio di errore è stato impostato correttamente e il forward è stato eseguito correttamente.	Passed	Nessuna deviazione riscontrata.

3.9. TC9

Test ID	Descrizione
Caso di Test	TC8 - Verifica della corretta funzionalità della rimozione di una recensione, con i seguenti scenari 1. Recensione trovata e rimossa. 2. Recensione non trovata. 3. ID della recensione non valido. 4. ID recensione mancante. 5. Errore di database durante la rimozione della recensione.
Modulo/Componente Testato	Rimuovi Recensione Control
Responsabile del Test	Intero team
Data del Test	04/01/2025
Ambiente di Test	Ambiente di sviluppo, Eclipse IDE
Prerequisiti	L'utente deve essere registrato nel sistema e deve essere il Gestore.
Strumenti di Test	JUnit 5, Mockito, Servlet Test Environment, Database simulato con mock.

Test	Descrizione del test	Risultato Atteso	Risultato Effettivo	Stato del Test	Deviazioni Riscontrate
testReviewExists	Verifica che, se la recensione esiste, venga rimossa dal database e venga effettuato il redirect.	La recensione deve essere rimossa correttamente, e deve avvenire un redirect alla pagina successiva.	La recensione è stata correttamente eliminata e il redirect è avvenuto.	Passed	Nessuna deviazione riscontrata.
testReviewNotFound	Verifica che, se la recensione non viene trovata, venga effettuato un forward alla pagina di errore.	Se la recensione non esiste, deve essere effettuato un forward alla pagina di errore.	Il test ha passato. Il forwarding è stato eseguito correttamente alla pagina di errore.	Passed	Nessuna deviazione riscontrata.
testSQLExceptionOnDelete	Verifica che, in caso di errore SQL durante la rimozione della recensione, venga fatto un forward alla pagina di errore.	In caso di SQLException, deve essere visualizzato il messaggio di errore e il forwarding alla pagina di errore deve avvenire.	L'eccezione SQLException è stata gestita correttamente e il forwarding è stato eseguito.	Passed	Nessuna deviazione riscontrata.

testReviewCodeMissing	Verifica che, se il parametro reviewId è mancante, venga fatto un forward alla pagina di errore.	Il messaggio di errore "Review code is required" deve essere visualizzato e il forward alla pagina di errore deve avvenire.	Il messaggio di errore è stato impostato correttamente e il forward è stato eseguito correttamente.	Passed	Nessuna deviazione riscontrata.
testInvalidReviewCodeFormat	Verifica che, se il parametro reviewId ha un formato non valido, venga fatto un forward alla pagina di errore.	Il messaggio di errore "Invalid review code format" deve essere visualizzato e il forward alla pagina di errore deve avvenire.	Il messaggio di errore è stato impostato correttamente e il forward è stato eseguito correttamente.	Passed	Nessuna deviazione riscontrata.

3.10. TC10

Test ID	Descrizione
Caso di Test	TC8 - Verifica della corretta funzionalità della gestione di una richiesta di supporto, con i seguenti scenari 1. Richiesta trovata e mostrata. 2. Errore di database nell'ottenimento della richiesta. 3. Funzionalità di richiesta informazioni con parametri validi. 4. Funzionalità di richiesta informazioni con parametri invalidi. 5. Funzionalità di aggiornamento stato con parametri validi. 6. Funzionalità di aggiornamento stato con parametri invalidi. 7. Funzionalità di aggiornamento stato con richiesta inesistente. 8. Funzionalità di richiesta informazioni con richiesta inesistente.
Modulo/Componente Testato	Gestione Richieste Supporto Control
Responsabile del Test	Intero team
Data del Test	04/01/2025
Ambiente di Test	Ambiente di sviluppo, Eclipse IDE
Prerequisiti	L'utente deve essere registrato nel sistema e deve essere il Gestore.
Strumenti di Test	JUnit 5, Mockito, Servlet Test Environment, Database simulato con mock.

Test	Descrizione del test	Risultato Atteso	Risultato Effettivo	Stato del Test	Deviazioni Riscontrate
testDoGet	Verifica che tutte le richieste di supporto siano recuperate e impostate correttamente nella richiesta.	La lista delle richieste di supporto dovrebbe essere impostata e la servlet dovrebbe inoltrare la richiesta al JSP di catalogo.	La lista delle richieste è stata impostata e la servlet ha inoltrato correttamente la richiesta al JSP.	Passed	Nessuna deviazione riscontrata.
testDoGetSQLException	Verifica che la servlet gestisca una SQLException correttamente durante il recupero delle richieste.	La servlet dovrebbe impostare un messaggio di errore e inoltrare la richiesta alla pagina di errore.	La servlet ha impostato il messaggio di errore e ha inoltrato la richiesta alla pagina di errore.	Passed	Nessuna deviazione riscontrata.
testDoPost_RichiediInformazioni_Valid	Verifica che la servlet gestisca correttamente la richiesta di aggiungere informazioni a una richiesta valida.	La richiesta di supporto dovrebbe essere aggiornata e la servlet dovrebbe inoltrare la richiesta al JSP di catalogo.	La richiesta è stata aggiornata correttamente e la servlet ha inoltrato la richiesta al JSP.	Passed	Nessuna deviazione riscontrata.
testDoPost_RichiediInformazioni_Invalid	Verifica che la servlet gestisca correttamente dati non validi per l'aggiunta di informazioni.	La servlet dovrebbe impostare un messaggio di errore e inoltrare la richiesta alla pagina di errore.	La servlet ha impostato il messaggio di errore e ha inoltrato la richiesta alla pagina di errore.	Passed	Nessuna deviazione riscontrata.
testDoPost_AggiornaStato_Valid	Verifica che la servlet gestisca correttamente l'aggiornamento dello stato di una richiesta valida.	La richiesta di supporto dovrebbe essere aggiornata e la servlet dovrebbe inoltrare la richiesta al JSP di catalogo.	La richiesta è stata aggiornata correttamente e la servlet ha inoltrato la richiesta al JSP.	Passed	Nessuna deviazione riscontrata.
testDoPost_AggiornaStato_Invalid	Verifica che la servlet gestisca correttamente dati non validi per l'aggiornamento dello stato.	La servlet dovrebbe impostare un messaggio di errore e inoltrare la richiesta alla pagina di errore.	La servlet ha impostato il messaggio di errore e ha inoltrato la richiesta alla pagina di errore.	Passed	Nessuna deviazione riscontrata.

testDoPost_RichiediInformazioni_RichiestaNonEsistente	Verifica che la servlet gestisca il caso in cui la richiesta non esiste durante l'aggiunta di informazioni.	La servlet dovrebbe impostare un messaggio di errore e inoltrare la richiesta alla pagina di errore.	La servlet ha impostato il messaggio di errore e ha inoltrato la richiesta alla pagina di errore.	Passed	Nessuna deviazione riscontrata.
testDoPost_AggiornaStato_RichiestaNonEsistente	Verifica che la servlet gestisca il caso in cui la richiesta non esiste durante l'aggiornamento dello stato.	La servlet dovrebbe impostare un messaggio di errore e inoltrare la richiesta alla pagina di errore.	La servlet ha impostato il messaggio di errore e ha inoltrato la richiesta alla pagina di errore.	Passed	Nessuna deviazione riscontrata.

4. TEST INCIDENT REPORT

4.1. Fallimento Gestione SQLEcxception

TC: 10.
Caso di Test: testDoGetSQLException
Descrizione dell'Incidente: Durante l'esecuzione del test, quando il metodo getAllSupportRequests() nel DAO genera un'eccezione SQLException, il servlet è in grado di rilevare l'errore e reindirizzare l'utente alla pagina di errore. Tuttavia, il messaggio di errore passato alla richiesta è generico ("Errore nel recupero delle richieste di supporto") e non fornisce informazioni sufficienti per il debug. Questo potrebbe impedire agli sviluppatori di individuare con precisione il tipo di errore nel database o la sua causa sottostante, se non accompagnato da log aggiuntivi.
Azione Richiesta:

- Migliorare la gestione degli errori nel servlet per fornire messaggi di errore più dettagliati e specifici, possibilmente includendo l'ID della richiesta o informazioni relative alla connessione al database.
- Implementare una logica di logging che memorizzi l'errore in un log di sistema, permettendo agli sviluppatori di indagare meglio sui problemi di database senza ricorrere al codice sorgente.
- Aggiungere un meccanismo di gestione delle eccezioni per individuare specifici errori di connessione al database (es. SQLTimeoutException, CommunicationsException).

4.2. Fallimento restituzione UtenteRegistrato

TC: 1
Caso di Test: testLoginSuccess
Descrizione dell'Incidente:
Durante l'esecuzione del test testLoginSuccess, si è verificato un incidente quando il metodo authenticate nel DAO ha restituito null invece di un oggetto UtenteRegistrato valido, nonostante i parametri corretti siano stati passati nella richiesta. L'incidente è stato causato da un errore nella logica del mock dell'oggetto userDAO. In particolare, il valore restituito da userDAO.authenticate non è stato configurato correttamente nel mock, portando a un comportamento imprevisto. Questo

ha causato un fallimento nel flusso di autenticazione, impedendo l'impostazione dell'utente nella sessione e la redirezione alla pagina principale. Il test non ha quindi passato la verifica delle azioni eseguite, poiché la sessione non è stata aggiornata e il metodo sendRedirect non è stato invocato. Inoltre, non sono stati registrati log adeguati, quindi non è stato possibile identificare rapidamente la causa del problema senza esaminare il codice di test.

Azione Richiesta:

- Correggere la configurazione del mock di userDAO per assicurarsi che venga restituito l'utente autenticato corretto durante l'esecuzione del test.
- Implementare un controllo aggiuntivo per verificare che i parametri passati alla richiesta siano corretti e che l'utente venga effettivamente autenticato prima di procedere con l'aggiornamento della sessione.
- Aggiungere log appropriati per tracciare la corretta esecuzione dei metodi e facilitare il debug degli errori in futuro.
- Aggiungere test di errore che simuli situazioni in cui l'autenticazione fallisce (utente non trovato, password errata, problemi di connessione al database) per garantire che il sistema gestisca correttamente queste situazioni.

4.3. Errore nella gestione di null

TC: 2

Caso di Test: testUnauthenticatedUser

Descrizione dell'Incidente:

Durante l'esecuzione del test testUnauthenticatedUser, si è verificato un problema relativo alla gestione del caso di un utente non autenticato. Nonostante il mock della sessione fosse stato configurato correttamente con un valore null per l'attributo "user", l'errore di autenticazione non è stato gestito correttamente. In particolare, il metodo doGet non ha impostato il messaggio di errore nell'oggetto request come previsto, né ha inviato il forwarding della richiesta al dispatcher per la pagina di errore. Questo ha causato un malfunzionamento nel flusso del test, poiché la gestione dell'errore non è stata eseguita correttamente, e la pagina di errore non è stata visualizzata all'utente. Inoltre, il test non ha potuto verificare che il carrello non fosse stato considerato, il che ha impedito di assicurarsi che il sistema non procedesse con l'acquisto in assenza di un utente autenticato.

Azione Richiesta:

- Correggere la logica di gestione dell'errore nel metodo doGet in modo che, quando un utente non autenticato cerca di accedere alla pagina di acquisto, venga effettivamente impostato il messaggio di errore nell'oggetto request e il forwarding venga effettuato correttamente.
- Aggiungere controlli di errore per verificare che il flusso di gestione dell'errore avvenga in tutte le condizioni previste (ad esempio, quando l'utente non è autenticato o quando ci sono altri errori nella sessione).
- Verificare che l'oggetto requestDispatcher venga effettivamente invocato per inoltrare la richiesta alla pagina di errore in caso di fallimento.
- Aggiungere un test che verifichi anche i casi in cui il carrello è vuoto, per coprire tutte le situazioni possibili in cui l'utente potrebbe trovarsi durante il flusso di acquisto.

5. TEST SUMMARY REPORT

5.1. Incidenti risolti

Test di gestione dell'errore SQLException (testDoGetSQLException):

- **Descrizione dell'Incidente:** Durante l'esecuzione del test, quando il metodo getAllSupportRequests() nel DAO ha generato una SQLException, il servlet ha rilevato l'errore e reindirizzato l'utente alla pagina di errore. Tuttavia, il messaggio di errore era troppo generico e non forniva informazioni sufficienti per il debug.
- **Azione Risolta:** È stata migliorata la gestione degli errori nel servlet per fornire messaggi di errore più dettagliati, inclusi l'ID della richiesta e informazioni relative alla connessione al database. È stata implementata una logica di logging per memorizzare l'errore in un log di sistema, consentendo agli sviluppatori di indagare meglio sui problemi senza dover esaminare il codice sorgente. È stata anche aggiunta una gestione delle eccezioni per individuare specifici errori di connessione al database, come SQLTimeoutException e CommunicationsException.

Test di login con successo (testLoginSuccess):

- **Descrizione dell'Incidente:** Durante l'esecuzione del test testLoginSuccess, il metodo authenticate nel DAO ha restituito null invece di un oggetto UtenteRegistrato valido, nonostante i parametri corretti. L'incidente è stato causato da un errore nella logica del mock di userDAO, dove il valore restituito non era configurato correttamente.
- **Azione Risolta:** È stata corretta la configurazione del mock di userDAO per restituire l'utente autenticato corretto. È stato implementato un controllo aggiuntivo per verificare che i parametri passati alla richiesta siano corretti e che l'utente venga autenticato prima di procedere con l'aggiornamento della sessione. Inoltre, sono stati aggiunti log appropriati per tracciare la corretta esecuzione dei metodi e facilitare il debug degli errori. Sono stati aggiunti test di errore per simulare situazioni in cui l'autenticazione fallisce (utente non trovato, password errata, problemi di connessione al database).

Test di utente non autenticato (testUnauthenticatedUser):

- **Descrizione dell'Incidente:** Durante l'esecuzione del test testUnauthenticatedUser, nonostante il mock della sessione fosse stato configurato correttamente con un valore null per l'attributo "user", l'errore di autenticazione non è stato gestito correttamente. Il metodo doGet non ha impostato il messaggio di errore nell'oggetto request né ha eseguito il forwarding della richiesta alla pagina di errore.
- **Azione Risolta:** È stata corretta la logica di gestione dell'errore nel metodo doGet, garantendo che, quando un utente non autenticato cerca di accedere alla pagina di acquisto, venga impostato il messaggio di errore e il forwarding venga effettuato correttamente. Sono stati aggiunti controlli di errore per garantire che il flusso di gestione dell'errore avvenga in tutte le condizioni previste. È stato verificato che l'oggetto requestDispatcher venga invocato per inoltrare la richiesta alla pagina di errore. È stato anche aggiunto un test per verificare che il carrello venga trattato correttamente quando l'utente non è autenticato.

Test di aggiunta prodotto al carrello (testAddProductToCart):

- **Descrizione dell'Incidente:** Durante l'esecuzione del test testAddProductToCart, si è verificato un errore relativo alla gestione del carrello. Non è stato verificato correttamente se il carrello fosse già presente nella sessione. Inoltre, la gestione delle eccezioni per il recupero del prodotto tramite productDAO.getProductById(1) non gestiva correttamente eventuali errori di database.
- **Azione Risolta:** È stata implementata una logica di controllo per verificare che il carrello venga recuperato dalla sessione solo se è presente, altrimenti viene inizializzato correttamente. È stata aggiunta una gestione delle eccezioni per la chiamata a productDAO.getProductById(1) per gestire correttamente eventuali errori di database, come prodotti non trovati o problemi di connessione. È stata implementata una gestione degli errori nella risposta del server per garantire che, in caso di problemi durante il recupero del prodotto, l'utente venga reindirizzato a una pagina di errore o visualizzi un messaggio di errore. Il test è stato modificato per includere casi in cui il carrello non fosse presente nella sessione e verificare che venga inizializzato correttamente.

5.2. Valutazione delle tecniche di testing usate

Per SoundShelf sono state adottate due tecniche di testing principali per ottimizzare il processo di verifica e garantire un'elevata qualità del sistema: la **Strategia Bottom up** per il testing di integrazione e la **Category Partition** per la selezione dei casi di test.
Queste metodologie sono state scelte per coprire in modo efficiente vari aspetti del sistema e per massimizzare la capacità di rilevare difetti in fase di sviluppo. Di seguito vengono analizzate nel dettaglio, insieme alle loro limitazioni, in modo da valutare la loro efficacia e identificare eventuali aree di miglioramento.

5.2.1. Strategia Bottom-up

La Strategia Bottom-Up è una metodologia di testing che parte dal testing delle unità o componenti di base per poi proseguire verso i livelli più alti del sistema. In questa tecnica, i moduli più bassi vengono testati per primi, e successivamente vengono integrati con i moduli superiori fino a coprire l'intero sistema in un ambiente integrato.
Vantaggi:

- **Identificazione precoce degli errori nei componenti di base:** Testando prima i moduli di basso livello, è possibile individuare e risolvere errori critici già nelle fasi iniziali del processo di sviluppo.
- **Stabilità dei moduli base:** Garantire il corretto funzionamento dei componenti fondamentali semplifica l'integrazione e il testing successivo.
- **Riduzione della complessità:** Procedendo gradualmente verso i livelli superiori, è più facile isolare e risolvere problemi in modo sistematico.

Limitazioni:

- **Necessità di driver di test:** Nei primi stadi del processo, dove i moduli superiori non sono ancora disponibili, è necessario sviluppare driver di test per simulare i moduli mancanti, aumentando i costi iniziali.
- **Ritardi nell'individuazione di problemi di integrazione:** Poiché i moduli superiori vengono testati solo nelle fasi avanzate, eventuali errori di integrazione potrebbero essere scoperti in ritardo.

- **Dipendenza dall'affidabilità dei moduli base:** Qualsiasi problema nei componenti di basso livello può avere un impatto significativo sui test successivi.

5.2.2. Category Partition

La **Category Partition** è una tecnica di testing che si basa sulla suddivisione delle possibili condizioni di input in categorie equivalenti. Ogni categoria rappresenta un gruppo di dati che, per quanto riguarda il sistema, è trattato in modo simile. L'idea principale di questa tecnica è di ridurre il numero di test necessari attraverso la selezione di un rappresentante per ogni categoria di dati, assicurando al contempo che il comportamento del sistema venga correttamente verificato.

Vantaggi:

- **Ottimizzazione dei Casi di Test**: La Partizione delle Categorie riduce il numero di casi di test necessari, poiché consente di testare gruppi di condizioni simili con un singolo caso, riducendo la complessità e il tempo complessivo del ciclo di test.
- **Flessibilità nell'analisi dei dati**: La tecnica è molto utile in scenari in cui i dati di input variano in modo significativo. Permette di validare la logica del sistema con un numero ridotto di test, pur coprendo un ampio spettro di condizioni di input.
- **Individuazione di Errori nelle Condizioni Limite**: La Partizione delle Categorie aiuta a identificare gli errori legati a condizioni limite, come valori estremi o input non validi, testando situazioni che potrebbero facilmente passare inosservate durante il testing tradizionale.

Limitazioni:

- **Difficoltà nell'Identificazione delle Categorie**: Una delle principali difficoltà nell'applicazione della Partizione delle Categorie è l'identificazione corretta delle categorie equivalenti. Se le categorie non fossero definite in modo preciso, i test potrebbero non coprire tutte le possibili situazioni di errore.
- **Semplicità nelle Categorie di Dati**: La tecnica si adatta meglio a casi in cui i dati di input sono relativamente semplici e ben definiti. Nei sistemi complessi con numerosi fattori variabili (come interazioni utente, condizioni di rete o configurazioni hardware), potrebbe essere difficile applicare correttamente la partizione delle categorie.
- **Mancanza di Verifica dei Comportamenti Complessi**: Anche se la tecnica riduce il numero di test, non garantisce una copertura completa di tutti i possibili flussi del sistema. Potrebbero esserci condizioni di errore o percorsi di esecuzione non esplorati, che rimangono fuori dalla copertura del test.