Università degli Studi di Salerno
Corso di Ingegneria del Software

CoinVerter
Test Case Specification
Versione 1.2

Data: 01/02/2025

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Pastore Alfredo	0512108925
Perna Alessandro	0512118963

Scritto da:	Pastore Alfredo

Revision History

Data	Versione	Descrizione	Autore
16/12/2024	1.0	Prima versione del documento	Pastore Alfredo
28/12/2024	1.1	Aggiunti i test di sistema	Pastore Alfredo
01/02/2025	1.2	Aggiunti i test di unità	Pastore Alfredo

Indice

1. Test d'unità

1.1. ModifyProductServletTest(ModifyProductServlet.java)

1.1.1. Redirezione al form di modifica prodotto tramite doPost

Input: Id: l'id del prodotto da modificare
Oracolo: il prodotto con l'id corretto viene caricato in sessione
L'utente viene reindirizzato alla pagina di modifica prodotto

1.1.2. Test doGet

Input: request: l'insieme dei valori necessari all'avvio della servlet
Response: L'insieme dei valori d'uscita dalla servlet
Oracolo: la funzione doPost viene richiamata e riceve i valori di request e response

1.2. CarrelloTest(Carrello.java)

1.2.1. Aggiunta prodotto a carrello

Input: Prodotto: il prodotto da aggiungere, mock di un prodotto generico,
Quantità: la quantità di prodotto da aggiungere, valore intero casuale tra 1 e 1000
Oracolo: Prodotto aggiunto correttamente al carrello

1.2.2. Aggiunta prodotto a carrello quantità errata

Input: Prodotto: il prodotto da aggiungere, mock di un prodotto generico,
Quantità: la quantità di prodotto da aggiungere, 0
Oracolo: Prodotto non aggiunto al carrello

1.2.3. Aggiunta prodotto a carrello quantità errata

Input: Prodotto: il prodotto da aggiungere, valore null,
Quantità: la quantità di prodotto da aggiungere, valore intero casuale tra 1 e 1000
Oracolo: Prodotto non aggiunto al carrello

1.2.4. Rimozione prodotto da carrello

Input: Prodotto: il prodotto da aggiungere, mock di un prodotto generico,
Quantità: la quantità di prodotto da aggiungere, valore intero casuale tra 1 e 1000
Oracolo: Prodotto rimosso dal carrello
Requisiti: il test deve aggiungere il prodotto al carrello perché il test funzioni

1.2.5. Rimozione prodotto non presente da carrello

Input: Prodotto: il prodotto da aggiungere, mock di un prodotto generico,
Quantità: la quantità di prodotto da aggiungere, valore intero casuale tra 1 e 1000
Oracolo: Prodotto non rimosso dal carrello
Requisiti: il prodotto che stiamo cercando di rimuovere non deve essere presente nel carrello perché il test funzioni

1.3.CartServletTest(CartServlet.java)

1.3.1. Aggiunta prodotto a carrello tramite doPost

Input: ID: L'id del prodotto da aggiungere, 1
action: L'azione da effettuare, add
quantity: la quantità da aggiungere al carrello, 1
Oracolo: prodotto aggiunto al carrello,
carrello aggiornato nella sessione

1.3.2. Rimozione prodotto a carrello tramite doPost

Input: ID: L'id del prodotto da aggiungere, 1
action: L'azione da effettuare, remove
quantity: la quantità da aggiungere al carrello, 1
Oracolo: prodotto rimosso dal carrello,
carrello aggiornato nella sessione

1.3.3. Test doPost

Input: request: l'insieme dei valori necessari all'avvio della servlet
Response: L'insieme dei valori d'uscita dalla servlet
Oracolo: la funzione doGet viene richiamata e riceve i valori di request e response

1.4.CheckoutTest(Checkout.java)

1.4.1. Test acquisto tramite doGet

Input: User: utente che sta effettuando l'acquisto, mock di un utente generico
Oracolo: il carrello viene svuotato,
Gli acquisti vengono aggiunti al portafoglio dell'utente in sessione
L'utente viene reindirizzato alla pagina principale del sito
Viene richiamata la funzione doBuy() per il completamento dell'acquisto

1.4.2. Test doPost

Input: request: l'insieme dei valori necessari all'avvio della servlet
Response: L'insieme dei valori d'uscita dalla servlet
Oracolo: la funzione doGet viene richiamata e riceve i valori di request e response

1.5.OrderDaoDataSourceTest(OrderDaoDataSource.java)

1.5.1. Test funzione DoRetrieveAll

Input: nessuno
Oracolo: Tutti gli ordini vengono estratti dal database

1.5.2. Test funzione DoRetrieveByDateFilter

Input: data1: la data di inizio filtro, 1-1-1
data2: la data di fine filtro, 2-2-2

Oracolo: Tutti gli ordini effettuati tra data1 e data2 vengono estratti dal database

1.5.3. Test funzione DoRetrieveByNameFilter

Input: nome: il nome della persona che ha effettuato l'ordine, nome
Oracolo: Tutti gli ordini effettuati dalla persona scelta vengono estratti dal database

1.6.OrdiniServletTest(OrdiniServlet.java)

1.6.1. Recupero di tutti gli ordini tramite DoGet

Input: user: l'utente loggato al momento dell'operazione
Start: valore d'inizio del filtro, valore null
End: valore di fine del filtro, valore null
Oracolo: tutti gli ordini vengono caricati nella sessione

1.6.2. Recupero degli ordini filtrati per data tramite DoGet

Input: user: l'utente loggato al momento dell'operazione
Start: valore d'inizio del filtro, 1-1-1
End: valore di fine del filtro, 2-2-2
Oracolo: tutti gli ordini tra start e end vengono caricati nella sessione

1.6.3. Recupero degli ordini filtrati per nome tramite DoGet

Input: user: l'utente loggato al momento dell'operazione
Nome: nome dell'utente di cui si vuole recuperare gli ordini
Oracolo: tutti gli ordini effettuati dall'utente richiesto vengono caricati nella sessione

1.6.4. Test doPost

Input: request: l'insieme dei valori necessari all'avvio della servlet
Response: L'insieme dei valori d'uscita dalla servlet
Oracolo: la funzione doGet viene richiamata e riceve i valori di request e response

1.7.ManageProductServletTest(ManageProductServlet.java)

1.7.1. accesso non autorizzato

Input: user: l'utente che sta accedendo alla pagina
Oracolo: L'utente viene reindirizzato alla pagina principale del sito
Requisiti: L'utente non deve essere un admin per far funzionare il test

1.7.2. accesso con utente nullo

Input: user: l'utente che sta accedendo alla pagina, valore nullo
Oracolo: L'utente viene reindirizzato alla pagina principale del sito

1.7.3. ridirezione alla pagina di aggiunta prodotto

Input: user: l'utente che sta accedendo alla pagina
Attività:l'attività da svolgere, valore nullo
Oracolo: l'admin viene portato alla pagina di aggiunta prodotto

1.7.4. Modifica prodotto con campi vuoti

Input: user: l'utente che sta accedendo alla pagina
Attività:l'attività da svolgere, modify
Id: id del prodotto da modificare, ""
Valore: valore del prodotto da modificare, ""
Prezzo: prezzo del prodotto da modificare, ""
Tipo: tipo del prodotto da modificare, ""
Nome: nome del prodotto da modificare; ""
Foto: path alla foto dell'oggetto da modificare ""
Oracolo: i campi del prodotto vengono non aggiornati ed i valori rimangono invariati

1.7.5. Modifica prodotto

Input: user: l'utente che sta accedendo alla pagina
Attività: l'attività da svolgere, modify
Id: id del prodotto da modificare, 1
Valore: valore del prodotto da modificare, 1
Prezzo: prezzo del prodotto da modificare, 1
Tipo: tipo del prodotto da modificare, 1
Nome: nome del prodotto da modificare; nome
Foto: path alla foto dell'oggetto da modificare
Oracolo: i campi del prodotto vengono aggiornati correttamente

1.7.6. Aggiunta prodotto

Input: user: l'utente che sta accedendo alla pagina
Attività: l'attività da svolgere, add
Id: id del prodotto da aggiungere, 1
Valore: valore del prodotto da aggiungere, 1
Prezzo: prezzo del prodotto da aggiungere, 1
Tipo: tipo del prodotto da aggiungere, 1
Nome: nome del prodotto da aggiungere; nome
Foto: path alla foto dell'oggetto da aggiungere
Oracolo: il prodotto viene creato correttamente utilizzando i valori dei campi

1.7.7. Rimozione prodotto

Input: user: l'utente che sta accedendo alla pagina
Attività: l'attività da svolgere, remove
Id: id del prodotto da rimuovere, 1
Oracolo: il prodotto viene rimosso correttamente

1.7.8. Test doGet

Input: request: l'insieme dei valori necessari all'avvio della servlet
Response: L'insieme dei valori d'uscita dalla servlet
Oracolo: la funzione doPost viene richiamata e riceve i valori di request e response

1.8. ProductAdminServletTest(ProductAdminServlet.java)

1.8.1. Recupero di tutti i prodotti tramite doPost

Input: nessuno
Oracolo: tutti i prodotti vengono caricati in sessione
L'utente viene reindirizzato alla pagina di gestione prodotti

1.8.2. Test doGet

Input: request: l'insieme dei valori necessari all'avvio della servlet
Response: L'insieme dei valori d'uscita dalla servlet
Oracolo: la funzione doPost viene richiamata e riceve i valori di request e response

1.9.ShopServletTest(ShopServlet.java)

1.9.1. Recupero tutti i prodotti

Input: nessuno
Oracolo: tutti i prodotti vengono caricati e mostrati nella pagina di shop
Utente reindirizzato nella pagina dello shop

1.9.2. Recupero prodotti filtrati per nome

Input: filter: la parola da utilizzare per filtrare i prodotti
Oracolo: I prodotti che corrispondono al filtro vengono caricati e mostrati nella pagina di shop
Utente reindirizzato nella pagina dello shop

1.9.3. Recupero prodotti filtrati per categoria

Input: filter: la categoria da utilizzare per filtrare i prodotti
Oracolo: I prodotti che corrispondono al filtro vengono caricati e mostrati nella pagina di shop
Utente reindirizzato nella pagina dello shop

1.9.4. Test doGet

Input: request: l'insieme dei valori necessari all'avvio della servlet
Response: L'insieme dei valori d'uscita dalla servlet
Oracolo: la funzione doPost viene richiamata e riceve i valori di request e response

1.10. AccountAdminServletTest(AccountAdminServlet.java)

1.10.1. Gestione account

Input: nessuno
Oracolo: tutti gli account vengono caricati e mostrati nella sezione admin del sito
L'utente viene reindirizzato alla pagina di gestione

1.10.2. Test doGet

Input: request: l'insieme dei valori necessari all'avvio della servlet
Response: L'insieme dei valori d'uscita dalla servlet
Oracolo: la funzione doPost viene richiamata e riceve i valori di request e response

1.11. LoginTest(Login.java)

1.11.1. Autenticazione corretta tramite doPost

Input: email: la mail dell'utente che sta provando ad autenticarsi
Pwd: la password dell'utente che sta provando ad autenticarsi
Oracolo: l'utente si autentica correttamente
L'utente viene reindirizzato alla pagina principale del sito
Requisiti: L'utente deve essere presente all'interno del sistema

1.11.2. Autenticazione incorretta con mail vuota tramite doPost

Input: email: la mail dell'utente che sta provando ad autenticarsi,""
Pwd: la password dell'utente che sta provando ad autenticarsi
Oracolo: l'utente si non si autentica

1.11.3. Autenticazione incorretta con password vuota tramite doPost

Input: email: la mail dell'utente che sta provando ad autenticarsi,""
Pwd: la password dell'utente che sta provando ad autenticarsi
Oracolo: l'utente si non si autentica

1.11.4. Test doGet

Input: request: l'insieme dei valori necessari all'avvio della servlet
Response: L'insieme dei valori d'uscita dalla servlet
Oracolo: la funzione doPost viene richiamata e riceve i valori di request e response

1.12. LogoutTest(logout.java)

1.12.1. Deautenticazione tramite doGet

Input: nessuno
Oracolo: L'utente viene deautenticato correttamente
L'utente viene portato alla pagina principale del sito

1.12.2. Test doPost

Input: request: l'insieme dei valori necessari all'avvio della servlet
Response: L'insieme dei valori d'uscita dalla servlet
Oracolo: la funzione doGet viene richiamata e riceve i valori di request e response

1.13. ManageAccountServletTest(ManageAccountServlet.java)

1.13.1. Rimozione stato admin di un account utente

Input: activity: l'attività da svolgere, modify
Email: la mail legata all'account da modificare
Oracolo: Vengono rimossi i privilegi di admin dall'account
Requisiti: L'account deve essere di un admin perché il test funzioni

1.13.2. Elevazione a admin di un account utente

Input: activity: l'attività da svolgere, modify
Email: la mail legata all'account da modificare
Oracolo: Vengono forniti i privilegi di admin dall'account
Requisiti: L'account non deve essere di un admin perché il test funzioni

1.13.3. Rimozione di un account dal sistema

Input: activity: l'attività da svolgere, remove
Email: la mail legata all'account da rimuovere
Oracolo: L'account legato alla mail viene rimosso
Requisiti: L'account deve essere presente nel sistema perché test funzioni

1.13.4. Test doGet

Input: request: l'insieme dei valori necessari all'avvio della servlet
Response: L'insieme dei valori d'uscita dalla servlet
Oracolo: la funzione doPost viene richiamata e riceve i valori di request e response

1.14. RegisterServletTest(RegisterServlet.java)

1.14.1. Registrazione tramite doPost

Input: Name: nome dell'account da creare
Surname: cognome dell'account da creare
Email: email da collegare all'account
Pwd: password dell'account da creare
Oracolo: L'account viene creato utilizzando i parametri di input
L'utente viene reindirizzato alla pagina di login

1.14.2. Test doGet

Input: request: l'insieme dei valori necessari all'avvio della servlet
Response: L'insieme dei valori d'uscita dalla servlet

Oracolo: la funzione doPost viene richiamata e riceve i valori di request e response

1.15. UserTest(User.java)

1.15.1. Aggiunta al portafoglio vuoto di una valuta nuova

Input: valuta: nome della valuta da aggiungere, moneta
Valore: valore della valuta da aggiungere, 1.23
Oracolo: la valuta viene aggiunta al portafoglio correttamente

1.15.2. Aggiunta al portafoglio di una valuta nuova

Input: valuta: nome della valuta da aggiungere, moneta
Valore: valore della valuta da aggiungere, 1.23
Oracolo: la valuta viene aggiunta al portafoglio correttamente

2 Test di sistema

2.1 Register

Input: Nome: nome dell'utente che sta creando l'account, Pinco
Cognome: cognome dell'utente che sta creando l'account, Pallino
Mail: mail dell'utente che sta creando l'account, pincopallino@email.com
Password: nome dell'utente che sta creando l'account, testpas$
Oracolo: La registrazione va a buon fine
L'utente viene reindirizzato alla pagina principale del sito

2.2 Login con credenziali errate

Input: Nome: nome dell'utente che sta effettuando l'accesso
Cognome: cognome dell'utente che sta effettuando l'accesso, Pallino
Mail: mail dell'utente che sta effettuando l'accesso, pincopallino@email.com
Password: nome dell'utente che sta effettuando l'accesso, testpas
Oracolo: L'accesso fallisce
Requisiti: L'utente deve essere registrato perché il test funzioni

2.3 Login con credenziali corrette

Input: Nome: nome dell'utente che sta effettuando l'accesso
Cognome: cognome dell'utente che sta effettuando l'accesso, Pallino
Mail: mail dell'utente che sta effettuando l'accesso, pincopallino@email.com
Password: nome dell'utente che sta effettuando l'accesso, testpas$
Oracolo: L'accesso avviene con successo
L'utente viene reindirizzato alla pagina principale del sito
Requisiti: L'utente deve essere registrato perché il test funzioni

2.4 Logout

Input: click dell'utente sul tasto "logout"
Oracolo: L'utente viene scollegato con successo
Requisiti: l'utente deve essere autenticato perché il test funzioni

2.5 Acquisto

Input: l'utente effettua il login
Si sposta sulla pagina dello shop
Seleziona il primo prodotto
Seleziona 50 unità e le aggiunge al carrello
Si sposta nel carrello ed effettua il checkout
Inserisce i dati della carta ed effettua il pagamento
Oracolo: l'acquisto viene effettuato con successo
L'utente viene reindirizzato alla pagina principale

2.6 Rimozione da carrello

Input: l'utente effettua il login
Si sposta sulla pagina dello shop
Seleziona il primo prodotto
Seleziona 5 unità e le aggiunge al carrello
Si sposta nel carrello e rimuove gli oggetti
Oracolo: Il carrello viene svuotato con successo

2.7 Modifica prodotto

Input: l'utente effettua il login con un account admin
Si sposta nella sezione di gestione prodotti
E seleziona la funzione di modifica del primo prodotto
Una volta aperto il form modifica il valore "2" e il prezzo "1"
Conferma le modifiche
Oracolo: Il prodotto viene modificato con successo

2.8 Elimina prodotto

Input: l'utente effettua il login con un account admin
Si sposta nella sezione di gestione prodotti
E seleziona la funzione elimina del primo prodotto
Oracolo: Il prodotto viene cancellato dal sistema