Università degli Studi di Salerno
Corso di Ingegneria del Software

Future Forge Gear
Test Incident Report
Versione 1.0

Data: 30/01/2026

Progetto: FutureForgeGear	Versione: 1.0
Documento: Test Incident Report	Data: 30/01/2026

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Francesco Mascolo	0512117352
Francesco Lamanna	0512117166
Luigi Mazza	0512118612

Scritto da:	Francesco Mascolo

Revision History

Data	Versione	Descrizione	Autore
28/01/2026		Specifica Incidente riscontrato durante la fase di testing	Tutto il Team

Indice

1. Introduzione

Questo documento descrive le problematiche riscontrate durante la fase di testing relative al sistema FutureForgeGear. In un caso di test è stata identificata una situazione in cui il sistema ha generato un comportamento non conforme all'oracolo definito

1.1. References

Per stilare questo documento, è stato preso come riferimento il libro di testo "Object- Oriented Software Engineering Using UML, Patterns and Java: Third Edition, di Bernd Bruegge ed Allen H. Dutoit".

Ambiente di Esecuzione

Per il testing sono stati utilizzati i seguenti strumenti:
- JUnit: Per i test unitari delle funzionalità principali.
- Mockito: Per simulare sessioni utente e interazioni tra componenti.
- Selenium IDE: Per simulare l'interazione dell'utente con l'interfaccia web.

Questi strumenti hanno permesso di verificare il corretto funzionamento dei flussi di autenticazione, segnalazione e gestione del sistema.

2. Test Incident Report

Esecuzione Sbagliata: Nel Test TC08 ovvero "Modifica del Prezzo di un prodotto nel magazzino" abbiamo notato che in qualunque modo modificassimo il prezzo il sistema non permetteva la modifica

Test case ID	Test Frame	Esito
TC08.1	PR1	Modifica Effettuata: Il prezzo del prodotto viene aggiornato
TC08.2	PR2	Modifica non Effettuata: Viene visualizzato un messaggio di errore indicando che il prezzo di un prodotto deve essere maggiore di 0

Test Case ID	TC08.1	Test Frame	PR1
INPUT			
Categoria		Valore della scelta	
Nuovo Prezzo		“50”	
Oracolo			
Modifica Effettuata: Il prezzo del prodotto viene aggiornato			

Test Case ID	TC08.2	Test Frame	PR2
INPUT			
Categoria		Valore della scelta	
Nuovo Prezzo		“0”	
Oracolo			
Modifica non Effettuata: Viene visualizzato un messaggio di errore indicando che il prezzo di un prodotto deve essere maggiore di 0			

Esecuzione corretta: Il problema è stato risolto correggendo la gestione del campo prezzo all’interno della ModificaProdottoServlet, sostituendo la conversione numerica errata (da Long) con una conversione coerente con il tipo reale del dato (da Double). In questo modo il sistema accetta prezzi decimali (es. 149.99) e completa correttamente l’aggiornamento del prodotto.

Logica della Servlet (ModificaProdottoServlet)
Durante l’esecuzione del test TC08, la servlet recuperava il valore del prezzo dalla request e tentava di convertirlo tramite:
Long.parseLong(...)
Questa logica causava una NumberFormatException per qualsiasi prezzo contenente decimali (es. "149.99"), generando un HTTP 500 – Internal Server Error e impedendo il salvataggio nel database.

Dopo la correzione, la servlet utilizza:
Double.parseDouble(...)
eventualmente normalizzando il valore (es. sostituendo virgola con punto) per supportare input come 149,99. L'aggiornamento del prezzo viene così eseguito correttamente tramite il metodo di aggiornamento sul catalogo (es. updateProductPrice(id, prezzo)), senza interrompere il flusso di elaborazione.

Causa tecnica dell'errore (conversione errata del tipo numerico)
Il prezzo nel sistema è rappresentato come valore decimale (Double), ma veniva interpretato come intero (Long). Questo disallineamento tra tipo previsto e tipo effettivo portava al fallimento della servlet in fase di parsing e quindi alla mancata persistenza della modifica.

Coerenza tra messaggio e risultato reale (gestione risposta lato client)
In aggiunta, lato client la pagina mostrava comunque un messaggio del tipo:
"Modifica salvata: ..."
anche quando la risposta era un errore, perché lo script concatenava e mostrava il contenuto della risposta HTML di errore restituita dal server (pagina di errore Tomcat/TomEE). Dopo la correzione, lo script verifica response.ok prima di confermare l'avvenuto salvataggio, distinguendo correttamente tra:

- esito positivo (salvataggio completato)
- esito negativo (errore in servlet con messaggio/HTML di errore)

Questo garantisce che il messaggio mostrato all'utente sia coerente con lo stato reale dell'operazione.

Validazione tramite test (TC08 – Modifica prezzo prodotto in magazzino)
Dopo la correzione, il test TC08 è stato rieseguito con prezzi decimali (es. 149.99) e il sistema:

1. non genera più eccezioni in conversione
2. restituisce risposta con esito positivo
3. salva correttamente il nuovo prezzo

L'esecuzione del test ha confermato che la modifica viene persa solo in caso di input non valido, mentre con input corretto il prezzo viene effettivamente aggiornato nel database.