Laurea Triennale in Informatica - Università degli Studi di Salerno

Corso di *Ingegneria del Software* – Prof. C.Gravino

Test Summary Report

Report.it

Riferimento	ReVampAscent_TSR_V1.0
Versione	1.0
Data	05/01/2026
Destinatario	C.Gravino

Laurea Triennale in Informatica - Università degli Studi di Salerno
Corso di *Ingegneria del Software* - Prof.C.Gravino

Presentato da	Antonio Aliberti, Vincenzo Martucci, Raffaella Di Pasquale

Team composition

Nome	Ruolo nel progetto	Acronimo	Informazioni di contatto
Antonio Aliberti	Team Member	AA	a.aliberti56@studenti.unisa.it
Vincenzo Martucci	Team Member	VM	v.martucci5@studenti.unisa.it
Raffaella Di Pasquale	Team Member	RDP	r.dipasquale5@studenti.unisa.it

Sommario

Revision History

Data	Versione	Descrizione	Autori
06/01/2026	0.1	Prima stesura	[TEAM MEMBER]
06/01/2026	1.0	Revisione finale documento	[TEAM MEMBER]

1. Introduzione

1.1 Scopo del Sistema

Il sistema ReVamp Ascent è progettato per consentire la compravendita di beni d'arredamento e complementi per la casa attraverso una piattaforma e-commerce moderna, sicura e centralizzata.

Lo scopo principale è facilitare l'incontro tra domanda e offerta mediante un'infrastruttura digitale che automatizza i processi di ricerca, selezione, ordine, pagamento e gestione post-vendita, garantendo un'esperienza d'acquisto fluida e personalizzata per l'utente finale. Il documento di Test Plan ha l'obiettivo di descrivere ed analizzare le attività di Testing per la piattaforma ReVampAscent. Il fine è quello di garantire che ogni aspetto funzioni in modo corretto.

All'interno del documento sono riportate le strategie di testing adottate, quali funzionalità saranno testate e gli strumenti scelti per la rilevazione degli errori, con lo scopo di presentare al cliente finale una piattaforma priva di malfunzionamenti.
Sono state pianificate attività di testing per le seguenti gestioni:

- Gestione Identità e Accessi
- Gestione Catalogo
- Gestione Recensioni
- Amministrazione

1.2 Scopo del documento

Il seguente documento riporta e descrive le attività di Testing effettuate per garantire il corretto funzionamento della piattaforma ReVampAscent.

All'interno del documento saranno riportate le funzionalità testate, con i relativi strumenti e strategie utilizzati.

1.3 Riferimenti

Di seguito vengono elencate le relazioni tra il presente documento e gli altri documenti di testing.

Relazione con il Test Plan

Il Test Summary Report fa riferimento alle attività di testing specificate nel Test Plan.

Relazione con il Test Case Specification

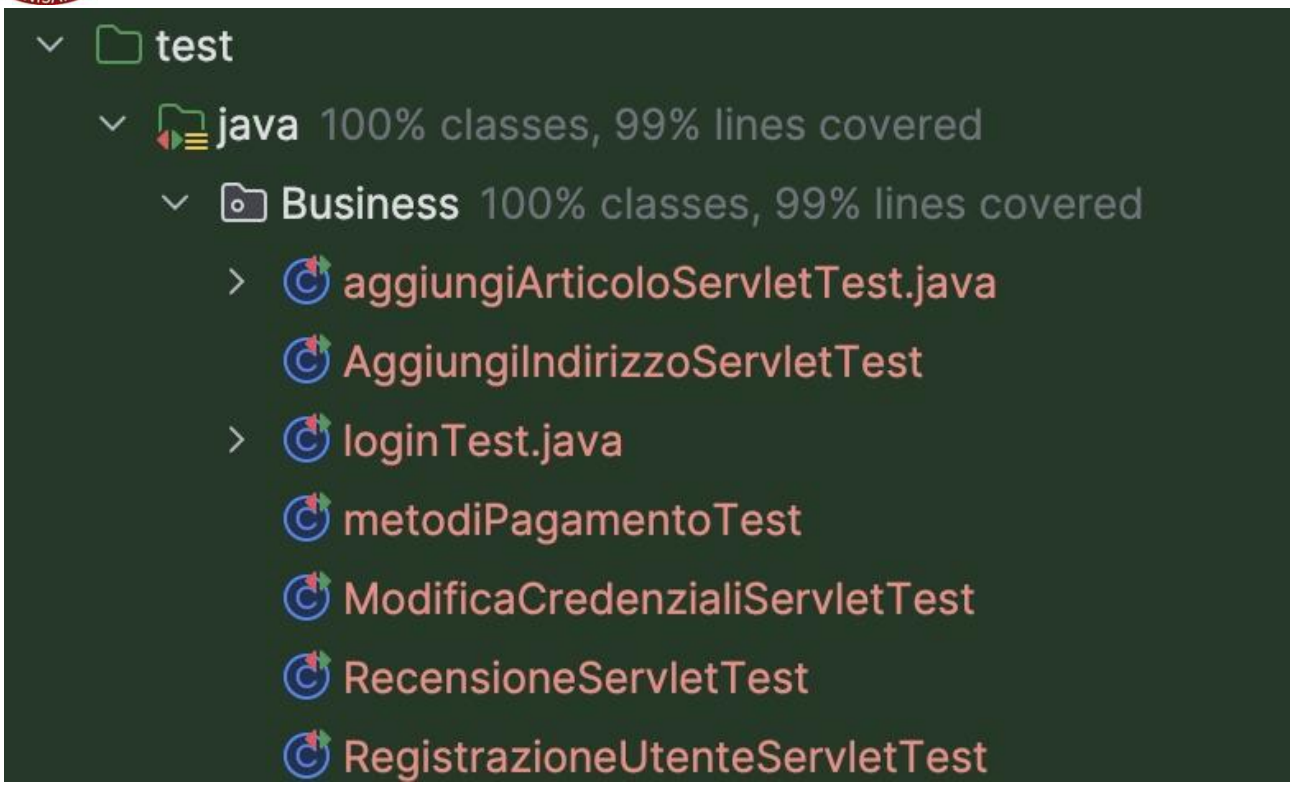
Il Test Summary Report contiene il sunto dell'esecuzione dei test di sistema specificati nel Test Case Specification.

Relazione con il Test Incident Report

Il Test Summary Report contiene il sunto dei risultati sull'esecuzione specificati nel Test Incident Report.

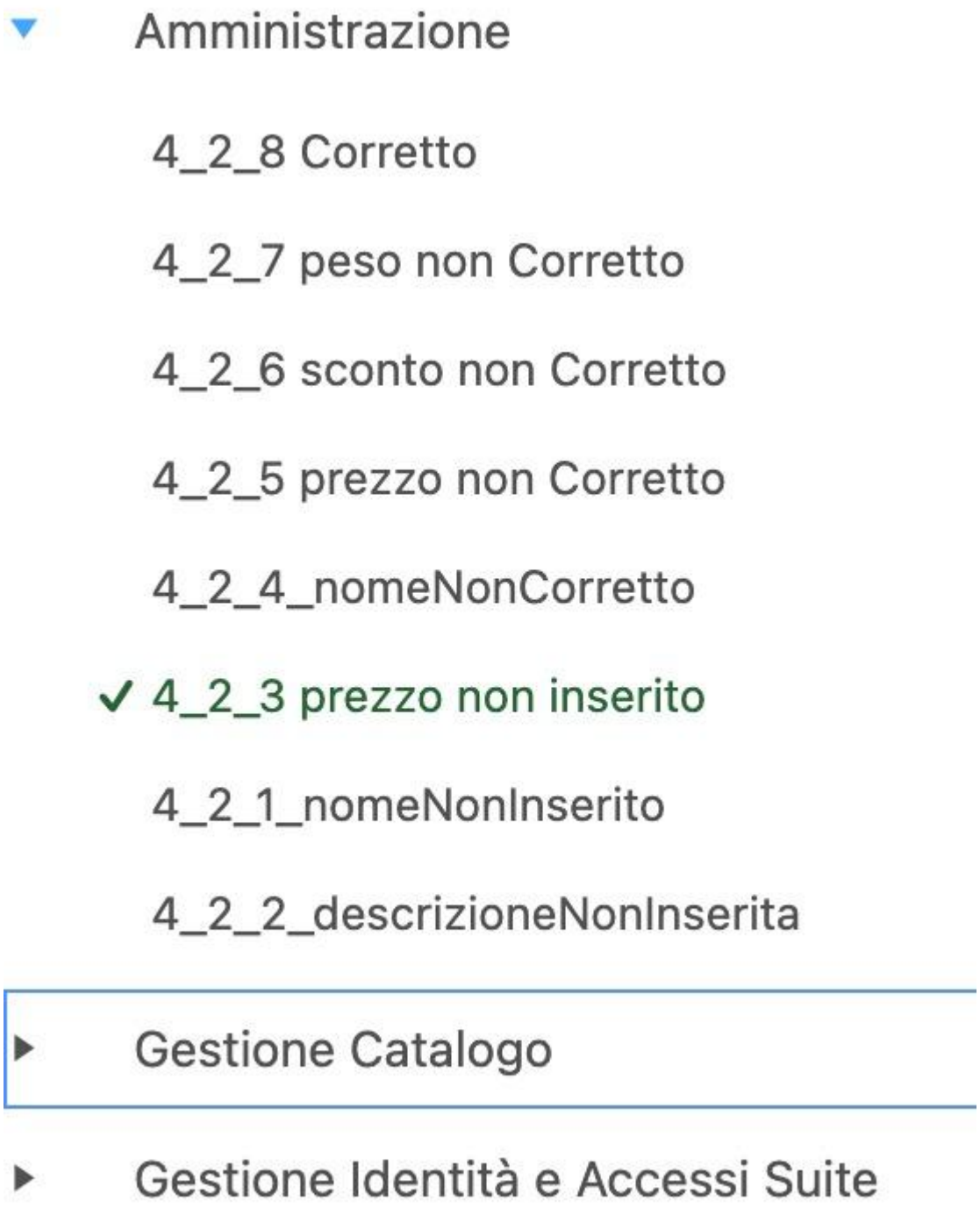
2. Testing di unità

I test di unità sono stati svolti al termine dell'implementazione dai Team Member, utilizzando le librerie **JUnit Test**, **Mockito** e **Jacoco** per le Branch Coverage. In caso di errori, lo sviluppatore ha provveduto alla correzione necessaria. Per il controllo front-end è stato utilizzato Selenium IDE.




```
@Test
void TC_1_5_3_cittaNull_parametriMancanti() throws Exception {
    when(req.getParameter( s: "via")).thenReturn( t: "Via Nola");
    when(req.getParameter( s: "citta")).thenReturn( t: null);
    when(req.getParameter( s: "cap")).thenReturn( t: "80049");
    when(req.getParameter( s: "provincia")).thenReturn( t: "RM");
    when(req.getParameter( s: "paese")).thenReturn( t: "Italia");
    when(req.getParameter( s: "preferito")).thenReturn( t: null

    new AggiungiIndirizzoServlet().doPost(req, res);

    assertEquals( expected: "{\"success\":false, \"message\":\"Parametri mancanti\"}", outContent.toString().trim());
    verify(res, never()).sendRedirect(anyString());
}
```

```java
@Test
void TC_1_2_4_usernameNonRegistrato_loginKO() throws Exception {
    HttpServletRequest req = mock(HttpServletRequest.class);
    HttpServletResponse res = mock(HttpServletResponse.class);
    RequestDispatcher dispatcher = mock(RequestDispatcher.class);

    when(req.getParameter( s: "username")).thenReturn( t: "Ssjsjsj21");
    when(req.getParameter( s: "password")).thenReturn( t: "michi12345");
    when(req.getRequestDispatcher( s: "login.jsp")).thenReturn(dispatcher);

    try (MockedConstruction<ClienteDAO> clienteCtor =
                 mockConstruction(ClienteDAO.class, (dao, ctx) -> {
                     when(dao.checkLogin( nome_utente: "Ssjsjsj21",  pass: "michi12345")).thenReturn( t: null);
                 });
         MockedConstruction<AdminDAO> adminCtor =
                 mockConstruction(AdminDAO.class, (dao, ctx) -> {
                     when(dao.doLogin( username: "Ssjsjsj21",  password: "michi12345")).thenReturn( t: null);
                 })) {

        new login().doPost(req, res);

        ArgumentCaptor<Object> captor = ArgumentCaptor.forClass(Object.class);

        verify(req).setAttribute(eq( value: "erroreLogin"), captor.capture());


        assertEquals( expected: "Nome utente o password errati", captor.getValue());
    }
```


Amministrazione
4_2_8 Corretto
4_2_7 peso non Corretto
4_2_6 sconto non Corretto
4_2_5 prezzo non Corretto
4_2_4_nomeNonCorretto
4_2_3 prezzo non inserito
4_2_1_nomeNonInserito
4_2_2_descrizioneNonInserita
Gestione Catalogo
Gestione Identità e Accessi Suite

	Command	Target	Value
2	✓ *click*	id=textboxRicerca	
3	✓ *type*	id=textboxRicerca	sedia
4	✓ *click*	css=.fa	
5	✓ *assert element present*	xpath=//ul[@id='listaProdottiModale']//li//a	
6	✓ *assert element not present*	xpath=//ul[@id='listaProdottiModale']//li[contains(normalize-space(.),'Nessun prodotto trovato')]	

3. Testing di sistema

Il testing di sistema è stato effettuato utilizzando manualmente l'applicazione, testandone ogni funzionalità con input differenti, per ogni tipo di utente.

Di seguito sono riportati i risultati dei test:

Esecuzione	Numero di Successi	Numero di Fallimenti
06/01/2026	83	8
06/01/2026	91	0

4. Coverage Dei Test

Branche Coverage	Line Coverage
54%	86%