Università degli Studi di Salerno
Corso di Ingegneria del Software

EduCat
Test Case Specification Document
Versione 1.1

Data: 30/12/2025

Coordinatore del progetto:

Nome	Matricola
Isabella Dima	0512119719

Partecipanti:

Nome	Matricola
Isabella Dima	0512119719
Anna Chiara Esposito	0512119143

Scritto da:	Anna Chiara Esposito

Revision History

Data	Versione	Descrizione	Autore
21/12/2025	1.0	Test Case Specification	Tutto il team
30/12/2025	1.1	Creazione Test Case mancanti e aggiunta valori in Input	Anna Chiara Esposito Isabella Dima

Indice

1. Introduction

Questo documento specifica i casi di test per il sottosistema “Gestione Lezione”, focalizzandosi sulla funzionalità critica di Prenotazione e Acquisto Lezione. I test coprono sia il flusso corretto che i flussi alternativi di errore, verificando l'integrazione tra l'interfaccia utente, la logica di business e la persistenza dei dati.

2. TC_LEZ_01: Prenotazione con successo

2.1. Test case specification identifier

- TC_LEZ_01: Prenotazione con successo

2.2. Test items

- it.unisa.educat.view.GestioneLezioneGUI (Pagina di Checkout)
- it.unisa.educat.control.PrenotazioneServlet (Controller)
- it.unisa.educat.business.GestioneLezioneManager (Metodo prenotaLezione)
- it.unisa.educat.storage.GestioneLezioneDAO

2.3. Input Specifications

- Precondizione: l’utente è loggato

seleziona la lezione e inserisce i dati di pagamento

ID Lezione

Categoria	**Input e Precondizioni**	**Oracolo**
empty	“” (stringa vuota)	Errore: “Selezionare una lezione”
ID non corretto (inesistente)	“001” (non presente nel Database)	Errore: “Lezione non trovata”
ID corretto (esistente)	“001” (presente nel Database)	[property Match]

Time Slot lezione

Categoria	**Input e Precondizioni**	**Oracolo**
Inizio lezione < current Time	“14 maggio 2021 15:00” (se oggi=15/05/2021 10:00)	Errore: “Data di inizio passata”
Inizio lezione = current Time	“15 maggio 2021 10:00” (se oggi=15/05/2021 10:00)	Errore: “La data di inizio non può essere ora”

Inizio lezione > current Time	“16 maggio 2021 15:00” (se oggi=15/05/2021 10:00)	Lezione trovata.
Fine lezione < current Time	“14 maggio 2021 16:00” (se oggi=15/05/2021 10:00)	Valido [if Match]
Fine lezione < Inizio lezione	Inizio=16:00, Fine=15:00	Errore: “La lezione non può finire prima della data d’inizio”
Time Slot Libero	Slot libero per l'utente	Valido [if Match]
Time Slot Occupato	Slot già prenotato (dallo stesso o altro utente)	Errore: “Time Slot occupato”

Numero Carta

Categoria	**Input e Precondizioni**	**Oracolo**
empty	““	Errore: “Inserire numero carta”
Numero Cifre = 16	“4532015112830366”	[property CifreCartaCorrette]
0 < Numero Cifre < 16	“123456789012345” (15 cifre)	Errore: “Numero carta non valido”
Numero Cifre > 16	“12345678901234567” (17 cifre)	Errore: “Numero carta non valido”
Numero Carta presente nel Database	“4916338506082832” (non nel DB)	Errore: “Carta non riconosciuta”
Numero Carta non presente nel Database	“4532015112830366” (nel DB)	[property NumeroCartaCorretto]

CVV

Categoria	**Input e Precondizioni**	**Oracolo**

empty	“” (stringa vuota)	Errore: “Inserire CVV”
Numero Cifre < 3	“12” (2 cifre)	Errore: “CVV non valido”
Numero Cifre > 3	“1234” (4 cifre)	Errore: “CVV non valido”
Numero Cifre = 3	“123”	[property CifreCVVCorrette]
Non corrispondente alla carta	Carta=4532..., CVV=“456” (mentre sul DB è “123”)	Errore: “CVV non corretto”
Corrispondente alla carta	Carta=4532..., CVV=“123” (come sul DB)	[property CVVCorretto]

Data di Scadenza

Categoria	**Input e Precondizioni**	**Oracolo**
empty	“”	Errore: “Inserire data scadenza”
< current Date	“01/2023” (se oggi=15/03/2024)	Errore: “Carta scaduta”
> current Date	“12/2025” (se oggi=15/03/2024)	Valido
Non corrispondente alla carta	Carta scade 12/2025, Input=06/2024	Errore: “Data scadenza non corretta”
Corrispondente alla carta	Carta scade 12/2025, Input=12/2025	[property DataScadenzaCorretta]

Titolare Carta

Categoria	**Input e Precondizioni**	**Oracolo**
empty	“” (stringa vuota)	Errore: “Inserire titolare

		carta”
Contiene numeri	“Mario123”	Errore: “Nome non valido”
Contiene caratteri speciali	“Luca@Bianchi”	Errore: “Nome non valido”
Solo lettere e spazi	“MARIO ROSSI”	Valido (formato corretto)
Non corrispondente alla carta	Carta intestata a “MARIO ROSSI”, Input=“LUCA VERDI”	Errore: “Intestatario non corretto”
Corrispondente alla carta	Carta intestata a “MARIO ROSSI”, Input=“MARIO ROSSI”	[property TitolareCartaCorretto]

2.4 Output Specifications

- Output Utente: Reindirizzamento alla pagina “Le mie lezioni” con messaggio di successo “Prenotazione confermata”.
- Stato Sistema: Nel Database, tabella Prenotazione, inserita nuova riga con IDStudente=101, IDLezione=50, stato=“attiva”.

2.5 Environmental Needs

- Server: Apache Tomcat 10.1 in esecuzione.
- Database: MySQL 8.0 con dataset di test caricato (l'ID Lezione 50 deve esistere ed essere libero).
- Browser/Driver: Google Chrome / Selenium WebDriver per test di sistema; JUnit 5 per test di integrazione.

2.6 Special procedural requirements

Nessuno. Il test può essere eseguito in qualsiasi momento purché il database sia nello stato startup previsto.

2.7 Intercase dependencies

Dipende dal Test Case TC_AUTH_01 (Login Studente): l'utente deve avere una sessione attiva per poter procedere al checkout.

3. TC_LEZ_02: Tentativo di prenotazione su slot già occupato

3.1. Test case specification identifier

- TC_LEZ_02**:** Tentativo di prenotazione su slot già occupato

3.2. Test items

Verifica la gestione delle eccezioni di business nel Manager e la loro presentazione all'utente.

- it.unisa.educat.business.GestioneLezioneManager
- Eccezione attesa: SlotOccupatoException (come definito nell'Object Design).

3.3. Input Specifications

- Precondizione: l'utente è loggato
- Lezione selezionata (id) e dati di pagamento

ID Lezione

Categoria	**Input e Precondizioni**	**Oracolo**
empty	"" (stringa vuota)	Errore: "Selezionare una lezione"
ID non corretto (inesistente)	"001" (non presente nel Database)	Errore: "Lezione non trovata"
ID corretto (esistente)	"001" (presente nel Database)	[property Match]

Time Slot lezione

Categoria	**Input e Precondizioni**	**Oracolo**
Inizio lezione < current Time	"14 maggio 2021 15:00" (se oggi=15/05/2021 10:00)	Errore: "Data di inizio passata"
Inizio lezione = current Time	"15 maggio 2021 10:00" (se oggi=15/05/2021 10:00)	Errore: "La data di inizio non può essere ora"
Inizio lezione > current Time	"16 maggio 2021 15:00" (se oggi=15/05/2021 10:00)	Lezione trovata.
Fine lezione < current Time	"14 maggio 2021 16:00" (se oggi=15/05/2021 10:00)	Valido [if Match]
Fine lezione < Inizio lezione	Inizio=16:00, Fine=15:00	Errore: "La lezione non può finire prima della data d'inizio"
Time Slot Libero	Slot libero per l'utente	Valido [if Match]

Time Slot Occupato	Slot già prenotato (dallo stesso o altro utente)	Errore: “Time Slot occupato”

3.4. Output Specifications
- Output Utente: La pagina di checkout viene ricaricata mostrando un messaggio di errore: *“Errore: Lo slot selezionato non è più disponibile.”*
- Stato Sistema: Nessuna nuova riga inserita nella tabella Prenotazione. Lo stato del database rimane invariato.

3.5. Environmental Needs
- Database: MySQL 8.0 con dataset di test caricato (l'ID Lezione 50 deve esistere ed essere libero).
- Browser/Driver: Google Chrome / Selenium WebDriver per test di sistema; JUnit 5 per test di integrazione.
- Vincolo Dati: Richiede che il database contenga già una prenotazione attiva per la Lezione ID xx (condizione creata dall'esecuzione di TC_LEZ_01).

3.6. Special procedural requirements
Nessuno. Il test può essere eseguito in qualsiasi momento purché il database sia nello stato startup previsto.

3.7. Intercase dependencies
Dipende da TC_LEZ_01 Questo test deve essere eseguito **dopo** che il test TC_LEZ_01 ha occupato lo slot con successo, oppure richiede una pre-condizione manuale sul DB.

4. TC_LEZ_03: Prenotazione con dati di pagamento errati

4.1. Test case specification identifier
- TC_LEZ_03**:** Prenotazione con dati di pagamento errati

4.2. Test items
Verifica i controlli di validazione input nel Controller/Servlet prima di invocare il Manager
- it.unisa.educat.control.PrenotazioneServlet

4.3. Input Specifications
- Precondizione: l’utente è loggato
- Lezione selezionata e dati di pagamento errati

ID Lezione

Categoria	**Input e Precondizioni**	**Oracolo**
empty	“” (stringa vuota)	Errore: “Selezionare una

		lezione”
ID non corretto (inesistente)	“001” (non presente nel Database)	Errore: “Lezione non trovata”
ID corretto (esistente)	“001” (presente nel Database)	[property Match]

Numero Carta

Categoria	**Input e Precondizioni**	**Oracolo**
empty	““	Errore: “Inserire numero carta”
Numero Cifre = 16	“4532015112830366”	[property CifreCartaCorrette]
0 < Numero Cifre < 16	“123456789012345” (15 cifre)	Errore: “Numero carta non valido”
Numero Cifre > 16	“12345678901234567” (17 cifre)	Errore: “Numero carta non valido”
Numero Carta presente nel Database	“4916338506082832” (non nel DB)	Errore: “Carta non riconosciuta”
Numero Carta non presente nel Database	“4532015112830366” (nel DB)	[property NumeroCartaCorretto]

CVV

Categoria	**Input e Precondizioni**	**Oracolo**
empty	““ (stringa vuota)	Errore: “Inserire CVV”
Numero Cifre < 3	“12” (2 cifre)	Errore: “CVV non valido”
Numero Cifre > 3	“1234” (4 cifre)	Errore: “CVV non valido”

Numero Cifre = 3	“123”	[property CifreCVVCorrette]
Non corrispondente alla carta	Carta=4532..., CVV=“456” (mentre sul DB è “123”)	Errore: “CVV non corretto”
Corrispondente alla carta	Carta=4532..., CVV=“123” (come sul DB)	[property CVVCorretto]

Data di Scadenza

Categoria	**Input e Precondizioni**	**Oracolo**
empty	““	Errore: “Inserire data scadenza”
< current Date	“01/2023” (se oggi=15/03/2024)	Errore: “Carta scaduta”
> current Date	“12/2025” (se oggi=15/03/2024)	Valido
Non corrispondente alla carta	Carta scade 12/2025, Input=06/2024	Errore: “Data scadenza non corretta”
Corrispondente alla carta	Carta scade 12/2025, Input=12/2025	[property DataScadenzaCorretta]

Titolare Carta

Categoria	**Input e Precondizioni**	**Oracolo**
empty	““ (stringa vuota)	Errore: “Inserire titolare carta”
Contiene numeri	“Mario123”	Errore: “Nome non valido”
Contiene caratteri speciali	“Luca@Bianchi”	Errore: “Nome non valido”
Solo lettere e spazi	“MARIO ROSSI”	Valido (formato corretto)

Non corrispondente alla carta	Carta intestata a “MARIO ROSSI”, Input=“LUCA VERDI”	Errore: “Intestatario non corretto”
Corrispondente alla carta	Carta intestata a “MARIO ROSSI”, Input=“MARIO ROSSI”	[property TitolareCartaCorretto]

4.4. Output Specifications

- Output Utente: Messaggio di errore inline nel form: *“Dati di pagamento non validi. Controllare numero carta e CVV.”*
- Stato Sistema: Il metodo GestioneLezioneManager.prenotaLezione() non viene invocato. Nessuna modifica al DB.

4.5. Environmental Needs

- Apache Tomcat 10.1
- JUnit per invocare la Servlet in ambiente mock.

4.6. Special procedural requirements

Nessuno. Il test può essere eseguito in qualsiasi momento purché il database sia nello stato startup previsto.

4.7. Intercase dependencies

Nessuna dipendenza specifica da altri test case funzionali.

5. TC_LEZ_04: Verifica eccezione “SlotOccupatoException”

5.1. Test case specification identifier

- TC_LEZ_04**:** Verifica eccezione “SlotOccupatoException”

5.2. Test items

Verifica la logica condizionale del rimborso nel Manager.

- it.unisa.educat.business.GestioneLezioneManager (Metodo annullaLezione)

5.3. Input Specifications

- Data Lezione: 20/01/2026
- Data Corrente (Simulata dal sistema): 18/01/2026 (Differenza > 24 ore).
- Azione: Click su “Annulla Lezione”.

5.4. Output Specifications

- Output Utente: Messaggio “Lezione annullata con successo. Il rimborso è stato emesso.”
- Stato Sistema:
  - Tabella Prenotazione: stato aggiornato a “ annullata”.
  - Tabella Pagamento: stato aggiornato a “rimborsato”

5.5. Environmental Needs
Necessità di poter manipolare la data di sistema o di utilizzare librerie come Mockito per mockare LocalDate.now() durante il test di unità, al fine di soddisfare la condizione if (dataLezione - dataCorrente > 24h).

5.6. Special procedural requirements
Richiede l'intervento del tester (o dello script di test) per impostare la data corrente simulata prima dell'esecuzione.

5.7. Intercase dependencies
Richiede una prenotazione attiva esistente (creata da TC_LEZIONE_01 o inserita manualmente).

6. TC_LEZ_05: Rollback della transazione Database in caso di fallimento pagamento

6.1. Test case specification identifier
TC_LEZ_05: Rollback della transazione Database in caso di fallimento pagamento

6.2. Test items
Verifica la robustezza del metodo prenotaLezione() nel gestire errori durante la fase di pagamento, garantendo l'atomicità della transazione (ACID).
- it.unisa.educat.business.GestioneLezioneManager
- it.unisa.educat.storage.GestioneLezioneDAO

6.3. Input Specifications
- Precondizione: Lo slot lezione è libero.
- Input: Dati di prenotazione validi ed il servizio di pagamento è configurato per lanciare un'eccezione (es: PaymentFailedException) al momento dell'invocazione.

ID Lezione

Categoria	Input e Precondizioni	Oracolo
empty	“” (stringa vuota)	Errore: “Selezionare una lezione”
ID non corretto (inesistente)	“001” (non presente nel Database)	Errore: “Lezione non trovata”
ID corretto (esistente)	“001” (presente nel Database)	[property Match]

Time Slot lezione

Categoria	Input e Precondizioni	Oracolo
Inizio lezione <	“14 maggio 2021 15:00” (se	Errore: “Data di inizio passata”

current Time	oggi=15/05/2021 10:00)	
Inizio lezione = current Time	“15 maggio 2021 10:00” (se oggi=15/05/2021 10:00)	Errore: “La data di inizio non può essere ora”
Inizio lezione > current Time	“16 maggio 2021 15:00” (se oggi=15/05/2021 10:00)	Lezione trovata.
Fine lezione < current Time	“14 maggio 2021 16:00” (se oggi=15/05/2021 10:00)	Valido [if Match]
Fine lezione < Inizio lezione	Inizio=16:00, Fine=15:00	Errore: “La lezione non può finire prima della data d’inizio”
Time Slot Libero	Slot libero per l'utente	Valido [if Match]
Time Slot Occupato	Slot già prenotato (dallo stesso o altro utente)	Errore: “Time Slot occupato”

6.4. Output Specifications

- Output Atteso: Il sistema deve catturare l'eccezione di pagamento e invocare il metodo di rollback sul DAO. Inoltre mostra un messaggio di errore all’utente (“Errore durante il pagamento, riprova”).
- Stato Sistema: La tabella Prenotazione non deve contenere la nuova riga relativa alla prenotazione fallita (“riga orfana”). Il database deve tornare allo stato precedente al tentativo di acquisto.

6.5. Environmental Needs

Utilizzo di Mockito per simulare il fallimento del metodo processPayment() senza dover interagire con un gateway bancario reale.

6.6. Special procedural requirements

Verificare i log di sistema per accertarsi che l’errore sia stato tracciato correttamente prima del rollback.

6.7. Intercase dependencies

Nessuna dipendenza specifica da altri test case funzionali.

7. TC_LEZ_06: Gestione Concorrenza (Race Condition)

7.1. Test case specification identifier

- TC_LEZ_06: Gestione Concorrenza (Race Condition)

7.2. Test items

Verifica i meccanismi di sincronizzazione del GestioneLezioneManager per prevenire il problema

della doppia prenotazione

7.3. Input Specifications

- Precondizione: Lezione ID 17 è libera
- Input: Due utenti invocano simultaneamente il metodo prenotaLezione() per lo stesso ID 17

Concorrenza

Categoria	**Input e Precondizioni**	**Oracolo**
Single-thread	1 utente, 1 richiesta alla volta	[property Single] • Prenotazione sempre riuscita se dati validi • Nessun conflitto di lock
Multi-thread	N utenti simultanei (N>1)	[property Multi] • Solo 1 prenotazione riuscita per slot • N-1 ricevono errore “Slot occupato” • Integrità dati preservata

Timing

Categoria	**Input e Precondizioni**	**Oracolo**
Sequenziale	Richieste in sequenza: • Utente A prenota • Utente B tenta stessa lezione	[property Seq] • Utente A: successo • Utente B: errore “Slot occupato” • Tempo tra richieste: > 0
Simultaneo/Overlap	Richieste nello stesso istante: • Utente A e B cliccano “Prenota” entro 1ms	[property Concurrent] • 1 successo, 1 errore (ordine casuale ma deterministico) • Race condition gestita • Nessuna doppia prenotazione

7.4. Output Specifications

- Output Atteso: solo uno dei due utenti deve completare l’operazione con successo. L’altro deve ricevere un errore (es: SlotOccupatoException)
- Stato Sistema: nel database deve essere presente una sola prenotazione per la lezione di ID 17. Non devono esistere prenotazioni duplicate o stati inconsistenti.

7.5. Environmental Needs

Framework di test che supporti il mutithreading per eseguire esecuzioni parallele.

7.6. Special procedural requirements
Il test deve essere ripetuto più volte per assicurarsi che la race condition venga effettivamente sollecitata e che il risultato sia deterministico anche sotto stress.

7.7. Intercase dependencies
Nessuna dipendenza specifica da altri test case funzionali.

8. TC_LEZ_07: Copertura del path “Annullamento Lezione” con rimborsi
8.1. Test case specification identifier
TC_LEZ_07: Copertura del path “Annullamento Lezione” con rimborsi

8.2. Test items
Verifica la logica condizionale temporale (24 ore) per l’emissione del rimporso
it.unisa.educat.business.GestioneLezioneManager (Metodo annullaLezione)

8.3. Input Specifications
- Scenario A (Rimborso): Data lezione impostata a Now() + 24 ore e 1 minuto.
- Scenario B (No Rimborso): Data lezione impostata a Now() + 23 ore e 59 minuti.

Prenotazione

Categoria	**Input e Precondizioni**	**Oracolo**
Empty	ID Prenotazione = “123” La prenotazione “123” non è presente nel DB	Errore: “Prenotazione non disponibile”
Svolta	ID Prenotazione = “123” La prenotazione “123” è presente nel DB. Stato prenotazione = “SVOLTA”	Errore: “Impossibile annullare lezione già svolta”
Attiva	ID Prenotazione = “123” La prenotazione “123” è presente nel DB. Stato prenotazione = “ATTIVA”	[property Attiva]

Tempo allo svolgimento

Categoria	**Input e Precondizioni**	**Oracolo**
<=24h	Prenotazione con data inizio “16/05/2021 15:00” (se oggi=16/05/2021 10:00)	Errore: “Rimborso non erogabile”
>24h	Prenotazione con data inizio “16/05/2021 15:00” (se oggi=14/05/2021 10:00)	Rimborso erogato [if Attiva]

8.4. Output Specifications

- Scenario A: La prenotazione passa a stato “annulata” e viene invocato il servizio di rimborso.
- Scenario B: La prenotazione passa a stato “annulata” ma non viene invocato il servizio di rimborso (controllare nel RAD)

8.5. Environmental Needs

Necessità di simulare la data di sistema (LocalDate.now() o Clock) per simulare con precisione la differenza temporale senza attendere ore reali.

8.6. Special procedural requirements

Il test deve essere strutturato in due step distinti (o due metodi di test separati nel codice JUnit) per coprire entrambi i rami condizionali (if e else) reimpostando il mock temporale tra un'esecuzione e l'altra.

8.7. Intercase dependencies

Richiede una prenotazione precedentemente creata e attiva (dipendenza da TC_LEZ_01).