Università degli Studi di Salerno
Corso di Ingegneria del Software

Test Cases Document - HeavyRoute

Versione 1.2

Informazioni Generali

Corso di Laurea:	Informatica
Università:	Università degli Studi di Salerno (UNISA)
Docente:	Chiar.mo Prof. DE LUCIA Andrea
Data:	13/01/2026
Anno Accademico:	2025/2026

Membri Del Gruppo

MANFREDINI Umberto	Matricola 0512119797
MANZO Ugo	Matricola 0512119071 *(Coordinatore)*
ROMANO Pino Fiorello	Matricola 0512120259

Revision History

Data	Versione	Descrizione	Autore
22/12/2025	1.0	Creazione del Test Case Doc.	Pino Fiorello Romano
29/12/2025	1.1	Aggiunta Category Partition	Umberto Manfredini
13/01/2026	1.2	Refactoring Nomi e Nuovi Test Validazione)	Ugo Manzo

Indice

1. Analisi Funzionale: Creazione Richiesta di Trasporto

1.1. Unità Funzionale

La funzionalità oggetto di analisi è la creazione di una nuova richiesta di trasporto (UC3).

- **Input:** Oggetto DTO contenente origine, destinazione, peso, data di ritiro.
- **Condizioni Ambientali:** Validità del Committente (ID).

1.2. Definizione delle Categorie e Scelte

Di seguito sono elencati i parametri identificati e le relative scelte (partitioning), annotate secondo la sintassi TSL (Test Specification Language).

Parametro: Indirizzo Origine
- `Stringa valida non vuota`
- `Stringa vuota o nulla` [**error**]

Parametro: Indirizzo Destinazione
- `Stringa valida non vuota`
- `Stringa vuota o nulla` [**error**]

Parametro: Peso Carico
- `Positivo (> 0)`
- `Zero o Negativo (<= 0)` [**error**]

Parametro: Data Ritiro
- `Data Futura` (Domani o oltre)
- `Data Odierna o Passata` [**error**]

Ambiente: Committente
- `Esistente e Attivo`
- `Non Esistente o Esistente e Non Attivo` [**error**]

2. Test Case 1: Creazione Richiesta Valida

2.1. 1. Test case specification identifier

TC-CORE-01 (Test Frame: Tutte Scelte Valide)

2.2. 2. Test items

- **Unit Under Test (UUT):** Classe `TransportRequestService`.
- **Method:** `createRequest(RequestCreationDTO dto, String username)`.
- **Feature:** Creazione e persistenza di una nuova richiesta (UC3).

2.3. 3. Input specifications

Combinazione Scelte: Origine[Valida] AND Destinazione[Valida] AND Peso[Positivo] AND Data[Futura] AND Committente[Esistente].

Valori Concreti:

- `dto`:

```
{
    "originAddress": "Napoli, Via Roma 1",
    "destinationAddress": "Roma, Via Tiburtina 10",
    "weight": 1500.0,
    "pickupDate": "2026-06-30"
}
```

- `username`: `"hitachi"`.

Stato del Mock:

- `UserRepository` configurato per restituire un oggetto `User` valido.

2.4. 4. Output specifications (Oracle)

Il test è considerato superato se:

1. Il metodo restituisce un oggetto `TransportRequestResponseDTO` non nullo.
2. `dto.getId()` è valorizzato.
3. `dto.getRequestStatus()` è `PENDING`.
4. Viene invocato `save()` sul repository.

2.5. 5. Environmental needs

JUnit 5, Mockito.

2.6. 6. Special procedural requirements

Configurare i mock (`when(...).thenReturn(...)`) prima dell'esecuzione.

2.7. 7. Intercase dependencies

Nessuna.

3. Test Case 2: Validazione Input (Error Cases)

3.1. Test Case 2.A: Peso Negativo

3.1.1. 1. Test case specification identifier

TC-CORE-02.A

3.1.2. 2. Test items

- **Unit Under Test (UUT):** `TransportRequestController`.
- **Feature:** Validazione degli input (Bean Validation).

3.1.3. 3. Input specifications

- **Combinazione:** Origine[Valida] AND Peso[Zero o Negativo].
- **Payload:**

```
{ "originAddress": "Napoli", "destinationAddress": "Roma",
  "weight": -50.0, "pickupDate": "2026-06-30" }
```

3.1.4. 4. Output specifications (Oracle)

1. **Status Code:** `400 Bad Request`.
2. **Body:** Contiene errore sul campo `weight`.
3. **Isolamento:** Il Service **NON** deve essere invocato.

3.2. Test Case 2.B: Data Passata

3.2.1. 1. Test case specification identifier

TC-CORE-02.B

3.2.2. 2. Test items

- **Unit Under Test (UUT):** `TransportRequestController`.

3.2.3. 3. Input specifications

- **Combinazione:** Origine[Valida] AND Data[Passata].
- **Payload:**

```
{ "originAddress": "Napoli", "destinationAddress": "Roma",
  "weight": 100.0, "pickupDate": "2020-01-01" }
```

3.2.4. 4. Output specifications (Oracle)

1. **Status Code:** `400 Bad Request`.
2. **Body:** Contiene errore sul campo `pickupDate`.

4. Analisi Funzionale: Approvazione Richiesta

4.1. Unità Funzionale

La funzionalità testata è l'approvazione di una richiesta da parte del PL e la generazione del viaggio (UC4).

- **Input:** `requestId` (Long).
- **Stato Sistema:** Presenza e stato della richiesta nel database.

4.2. Definizione delle Categorie

Parametro: Esistenza Richiesta
- `Esiste nel DB`
- `Non esiste` [**error**]

Parametro: Stato Corrente Richiesta
- `PENDING` (Stato valido per approvazione)
- `APPROVED` [**error**]
- `REJECTED` [**error**]

5. Test Case 3: Approvazione Valida

5.1. 1. Test case specification identifier

TC-CORE-03 (Test Frame: Richiesta Esistente + Stato Valido)

5.2. 2. Test items

- **Unit Under Test:** `TripService.approveRequest`.

5.3. 3. Input specifications

- **Argomento:** `requestId = 10L`.
- **Precondizione Mock:** Il repository restituisce una richiesta con ID 10 e stato `PENDING`.

5.4. 4. Output specifications (Oracle)

1. **Ritorno:** Oggetto `Trip` non nullo.
2. **Stato Trip:** Lo stato del nuovo viaggio deve essere `IN_PLANNING` (non ancora validato).
3. **Side Effect:** Lo stato della richiesta è aggiornato a `APPROVED`.
4. **Persistenza:** `tripRepository.save()` invocato.

5.5. 5. Environmental needs

JUnit 5, Mockito.

5.6. 6. Special procedural requirements

Verifica transazionalità.

5.7. 7. Intercase dependencies

Nessuna.

6. Analisi Funzionale: Assegnazione Risorse

6.1. Unità Funzionale

Funzionalità di associazione di risorse (Autista e Veicolo) a un viaggio pianificato (UC4).

6.2. Definizione delle Categorie

Parametro: Stato Viaggio
- `IN_PLANNING` (Valido)
- `Altro stato` [**error**]

Ambiente: Disponibilità Risorse
- `Autista FREE e Veicolo AVAILABLE` (Valido)
- `Autista BUSY o Veicolo BUSY` [**error**]

7. Test Case 4: Assegnazione Valida

7.1. 1. Test case specification identifier

TC-CORE-04 (Test Frame: Tutte Risorse Disponibili)

7.2. 2. Test items

- **Unit Under Test:** `TripService.planTrip`.

7.3. 3. Input specifications

- **Argomenti:** `tripId=50L, driverId=200L, vehicleId=300L`.
- **Precondizione Mock:**
  - Viaggio 50 esiste ed è `IN_PLANNING`.
  - `Driver` è `FREE`, `Vehicle` è `AVAILABLE`.

7.4. 4. Output specifications (Oracle)

1. Nessuna eccezione.
2. **Stato Finale:** Il viaggio transita nello stato **`WAITING_VALIDATION`** (in attesa del TC).
3. **Stato Risorse:** Autista `ASSIGNED`, Veicolo `IN_USE`.
4. `tripRepository.save()` invocato.

7.5. 5. Environmental needs

JUnit 5, Mockito.

7.6. 6. Special procedural requirements

Nessuno.

7.7. 7. Intercase dependencies

TC-CORE-03.

8. Test Case 5: Gestione Conflitto Risorse

8.1. 1. Test case specification identifier

TC-CORE-05 (Test Frame: Veicolo Non Disponibile [error])

8.2. 2. Test items

- **Unit Under Test:** `TripService.planTrip`.

8.3. 3. Input specifications

- **Argomenti:** Stessi di TC4.
- **Precondizione Mock:** Il veicolo recuperato ha stato `MAINTENANCE` o `IN_USE`.

8.4. 4. Output specifications (Oracle)

1. **Eccezione:** Viene lanciata `BusinessRuleException` (o `ResourceNotAvailableException`).
2. **Persistenza:** `tripRepository.save()` **non** viene invocato.

9. Test Case 6: Accesso Non Autorizzato

9.1. 1. Test case specification identifier

TC-CORE-06

9.2. 2. Test items

- **Unit Under Test:** `TripManagementController`.
- **Feature:** RBAC (Role Based Access Control).

9.3. 3. Input specifications

- **Context:** Utente autenticato con ruolo `CUSTOMER`.
- **Request:** `POST /api/trips/10/approve` (Endpoint riservato a `LOGISTIC_PLANNER`).

9.4. 4. Output specifications (Oracle)

1. **Status:** `403 Forbidden`.
2. **Service:** Nessuna invocazione a `TripService`.

9.5. 5. Environmental needs

Spring Security Test, MockMvc, `@WithMockUser(roles="CUSTOMER")`.

10. Analisi Funzionale: Validazione Rotta (Traffic Coordinator)

10.1. Unità Funzionale

Processo decisionale del Traffic Coordinator sulla rotta calcolata per un viaggio pianificato.

10.2. Definizione delle Categorie

Parametro: Decisione

- `Approved = TRUE` (Conferma viaggio)
- `Approved = FALSE` (Rifiuta e cancella)

11. Test Case 7: Approvazione Rotta (TC)

11.1. 1. Test case specification identifier

TC-CORE-07

11.2. 2. Test items

- **Unit Under Test:** `TripService.validateRoute`.

11.3. 3. Input specifications

- **Input:** `tripId=50L`, `approved=true`.
- **Stato Iniziale:** Viaggio in `WAITING_VALIDATION`.

11.4. 4. Output specifications (Oracle)

1. **Stato Viaggio:** Aggiornato a **`CONFIRMED`**.
2. **Stato Richiesta:** Aggiornato a **`PLANNED`**.
3. `tripRepository.save()` invocato.

12. Test Case 8: Rifiuto Rotta (TC)

12.1. 1. Test case specification identifier

TC-CORE-08

12.2. 2. Test items

- **Unit Under Test:** `TripService.validateRoute`.

12.3. 3. Input specifications

- **Input:** `tripId=50L`, `approved=false`, `feedback="Altezza ponte insufficiente"`.

12.4. 4. Output specifications (Oracle)

1. **Azione:** Il viaggio viene cancellato (per permettere ripianificazione).
2. **Persistenza:** `tripRepository.delete()` invocato.
3. **Notifica:** Inviata notifica al Planner.