Laurea Triennale in Informatica

Università di Salerno

Corso di Ingegneria del Software

Prof. Andrea De Lucia

Progetto EasyDrive

Test Plan

Versione 0.1

Data: 21/12/2025

Coordinatore del progetto:

Nome	Matricola
Mario Mascheri	0512120157
Claudio Brizio	0512119716

Partecipanti:

Nome	Matricola
Brizio Claudio	0512119716
Cannella Vincenzo	0512119065
Coscia Matteo	0512121210
Mascheri Mario	0512120157

Scritto da: Vincenzo Cannella

Cronologia delle Revisioni

Data	**Versione**	**Descrizione**	**Autore**
21/12/2025	0.1	Prima stesura e bozza del Test Plan	Vincenzo Cannella

Indice

1 Introduzione **3**

1.1 Obiettivi del Test 4

1.2 Estensione del Test (Scope) 4

2 Relazione con altri documenti **5**

2.1 Schema di Corrispondenza (Naming Scheme) 5

3 Panoramica del sistema **6**

3.1 Granularità dei Componenti 6

3.2 Dipendenze tra i Componenti 6

4 Funzionalità da testare / non testare **7**

4.1 Funzionalità da testare 7

4.2 Funzionalità non soggette a test 8

5 Criteri di Superamento/Fallimento **8**

5.1 Criteri di Successo del Test (Identificazione Fallimenti) 9

5.2 Criteri di Accettazione del Sistema 9

5.3 Gestione degli Incidenti 9

6 Approccio **10**

6.1 Strategia di Testing Generale 10

6.2 Strategia di Integrazione 10

6.3 Dipendenze tra i Test 11

6.4 Tecniche di Selezione dei Casi di Test 11

7 Sospensione e Ripresa **11**

7.1 Criteri di Sospensione 11

7.2 Criteri di Ripresa 12

8 Materiali di Test **12**

8.1 Risorse Hardware 13

8.2 Risorse Software 13

8.3 Strumenti di Testing e Supporto 13

9 Compiti (Tasks) **14**

9.1 Ruoli e Responsabilità 14

9.2 Matrice delle Attività 15

10 Pianificazione dei Tempi (Schedule) **15**

10.1 Test Cases 16

10.2 Funzionalità: Registrazione Utente 16

10.3 Funzionalità: Login (Autenticazione) 17

10.4 Funzionalità: Prenotazione Noleggio 18

10.5 Funzionalità: Pagamento 19

10.6 Funzionalità: Ricerca Veicoli nel Catalogo 20

11 Test Frames **22**

11.1 Test Frames - Registrazione Utente 22

11.2 Test Frames - Login (Autenticazione) 22

11.3 Test Frames - Prenotazione Noleggio 22

11.4 Test Frames - Pagamento 22

11.5 Test Frames - Ricerca Veicoli 23

1 Introduzione

L'obiettivo di questa sezione è definire gli scopi e l'ampiezza delle attività di verifica per il sistema Easy Drive. Il presente documento fornisce il framework operativo che i manager e i tester utilizzeranno per pianificare ed eseguire i test necessari in modo tempestivo ed efficiente sotto il profilo dei costi.

1.1 Obiettivi del Test

Le attività di testing mirano a garantire che il software soddisfi i requisiti specificati e sia privo di difetti critici prima del rilascio. Gli obiettivi principali includono:

- **Verifica Funzionale**: confermare che ogni sottosistema sia codificato correttamente e svolga le funzionalità previste, come la gestione dei veicoli (RF1), le prenotazioni (RF2), la ricerca avanzata (RF3), la gestione vendite (RF4) e lo storico operazioni (RF5).
- **Verifica dei Requisiti Globali**: determinare se il sistema rispetti i Requisiti Non Funzionali (RNF), con particolare attenzione alle prestazioni (risposta in meno di 2 secondi) e alla disponibilità (99% del tempo).
- **Identificazione dei Fallimenti**: esercitare il software con casi di test specifici per individuare deviazioni tra il comportamento osservato e quello specificato (failure) causate da bug algoritmici o meccanici.
- **Affidabilità e Sicurezza**: validare i meccanismi di protezione dei dati personali tramite HTTPS e cifratura, garantendo l'integrità delle informazioni sensibili.

1.2 Estensione del Test (Scope)

Il piano di test copre l'intero sistema Easy Drive, scomposizione inclusa:

- **Test di Componente (Unit Testing)**: verifica dei singoli package funzionali, tra cui `it.unisa.easydrive.catalog`, `it.unisa.easydrive.user`, `it.unisa.easydrive.booking`, `it.unisa.easydrive.sales`, `it.unisa.easydrive.payment`, `it.unisa.easydrive.staff` e `it.unisa.easydrive.core`.
- **Test di Integrazione**: verifica delle interfacce e delle interazioni tra i sottosistemi sopra citati per assicurare il corretto flusso di dati.
- **Test di Sistema**: valutazione dell'intero sistema integrato nel suo ambiente target (architettura client-server basata su PHP, Apache e MySQL).
- **Test di Accettazione**: sessioni di test finali condotte per dimostrare che il sistema risponde alle esigenze del cliente e sia pronto per l'uso operativo.

2 Relazione con altri documenti

Il presente Piano di Test è strettamente integrato con gli altri documenti prodotti durante il ciclo di vita del software, fungendo da strumento di verifica per le specifiche definite nelle fasi precedenti. Di seguito viene illustrata la relazione tra i test e la documentazione di progetto:

Requirements Analysis Document (RAD): il RAD costituisce la base primaria per il Functional Testing. Ogni test case funzionale è progettato per validare i requisiti estratti dagli scenari e dai casi d'uso definiti nel RAD, assicurando che il sistema soddisfi le necessità dell'utente finale.

System Design Document (SDD): l'SDD viene utilizzato come riferimento per l'Integration Testing. I test di integrazione verificano che le interazioni tra i sottosistemi (es. Catalogo, Prenotazioni, Pagamenti) avvengano secondo l'architettura client-server e il mapping hardware/software stabiliti.

Object Design Document (ODD): l'ODD guida l'esecuzione del Component Testing (Unit Test). I test unitari si concentrano sulla verifica delle interfacce pubbliche delle classi, degli attributi e dei metodi descritti nell'ODD, garantendo che ogni unità di codice sia corretta prima dell'integrazione.

Problem Statement: fornisce il contesto di alto livello per i test di accettazione, permettendo di verificare che l'intento originale di automatizzare la gestione dell'agenzia sia stato raggiunto con successo.

2.1 Schema di Corrispondenza (Naming Scheme)

Per stabilire una tracciabilità univoca tra i requisiti e i test, viene adottato il seguente schema di denominazione per i casi di test:

Prefisso Test	**Riferimento Documento**	**Descrizione**
TC-RF-*x*	RAD (Requisiti Funzionali)	Verifica di un requisito funzionale specifico (es. Gestione Veicoli RF1).
TC-RNF-*x*	RAD (Requisiti Non Funzionali)	Verifica di requisiti come prestazioni (RNF1) o sicurezza (RNF4).
TC-INT-*x*	SDD (Sottosistemi)	Verifica dell'interfaccia tra due o più sottosistemi definiti nel design.
TC-UNIT-*x*	ODD (Package/Classi)	Verifica di una singola classe o metodo all'interno dei package Java/PHP.

3 Panoramica del sistema

Questa sezione descrive gli aspetti strutturali del sistema Easy Drive, identificando i componenti che saranno oggetto di test unitari. La strategia di verifica si basa sulla scomposizione del sistema in sottosistemi isolabili, definiti come gruppi di oggetti o package funzionali.

3.1 Granularità dei Componenti

Ai fini del Unit Testing, il sistema viene suddiviso in package logici che rappresentano le unità minime di test. Ogni componente è progettato per essere testato individualmente dai programmatori per confermare che la logica interna e le funzionalità previste siano codificate correttamente.

I componenti principali identificati sono i seguenti:

- `it.unisa.easydrive.catalog`: gestisce la navigazione, la ricerca e il filtraggio dei veicoli.
- `it.unisa.easydrive.user`: include i moduli per l'autenticazione, la registrazione e la gestione del profilo utente.
- `it.unisa.easydrive.booking`: implementa la logica di business per il controllo della disponibilità e la gestione dei noleggi.
- `it.unisa.easydrive.sales`: gestisce il flusso d'acquisto, gli ordini e la persistenza dello storico delle transazioni.
- `it.unisa.easydrive.payment`: interfaccia il sistema con i servizi di validazione ed esecuzione dei pagamenti.
- `it.unisa.easydrive.staff`: fornisce le funzionalità di manutenzione del catalogo riservate ai gestori del sistema.
- `it.unisa.easydrive.core`: package trasversale contenente le entità condivise (es. Veicolo, Utente) e la logica di connessione al database.

3.2 Dipendenze tra i Componenti

La progettazione dei test deve tenere conto delle relazioni gerarchiche e funzionali tra i componenti per determinare l'ordine di esecuzione e l'eventuale necessità di strumenti di supporto (Stubs e Drivers):

- **Dipendenza dai Dati**: tutti i sottosistemi dello strato applicativo (Application Layer) dipendono dal componente dei dati persistenti (Data Layer) per l'accesso al DBMS MySQL.
- **Trasversalità del Core**: il package core rappresenta la base del sistema; le sue entità sono utilizzate da tutti gli altri componenti, rendendo la sua stabilità critica per l'intero processo di testing.
- **Interazione Client-Server**: poiché il sistema adotta un'architettura client-server, il browser (Client Node) dipende dai servizi esposti dal Web Server tramite protocollo HTTP/HTTPS.

4 Funzionalità da testare / non testare

Questa sezione identifica i requisiti funzionali e le combinazioni di funzionalità che saranno oggetto di verifica, oltre a specificare quali aspetti del sistema sono esclusi dal piano di test e le relative motivazioni.

4.1 Funzionalità da testare

Il focus primario del testing funzionale è validare che il sistema esegua correttamente le operazioni di business descritte nei requisiti. Saranno testate le seguenti funzionalità:

- **Gestione Veicoli (RF1)**: inserimento di nuove automobili, modifica delle informazioni (marca, modello, anno, prezzo, disponibilità) ed eliminazione dal catalogo.
- **Gestione Prenotazioni e Noleggi (RF2)**: creazione di nuove prenotazioni, registrazione dei noleggi e capacità di modifica o annullamento.
- **Calcolo Automatico Costi**: verifica della precisione nel calcolo dei costi di noleggio in base alla durata e al tipo di veicolo.
- **Ricerca Avanzata nel Catalogo (RF3)**: funzionamento corretto dei filtri personalizzati per marca, modello, anno di produzione, fascia di prezzo e disponibilità.
- **Gestione Vendite (RF4)**: inserimento delle informazioni relative ai clienti e gestione della procedura di acquisto.
- **Storico delle Operazioni (RF5)**: tracciamento e consultazione di tutte le transazioni di vendita e noleggio per l'analisi amministrativa.

- **Controllo degli Accessi**: validazione della matrice degli accessi per garantire che Guest, Clienti e Responsabili possano eseguire solo le operazioni autorizzate dal loro ruolo.
- **Combinazioni di Funzionalità**: test integrati su flussi complessi, come la ricerca di un veicolo seguita dalla sua prenotazione e dal relativo pagamento.

4.2 Funzionalità non soggette a test

Le seguenti caratteristiche non saranno verificate internamente dal team di sviluppo per le ragioni specificate:

- **Gestione dell'Infrastruttura Server**: operazioni di avvio (start-up), terminazione (shutdown) e configurazione hardware del server.
  Motivazione: queste attività sono gestite automaticamente dall'ambiente web predefinito (Web Server Apache/Nginx e runtime PHP) e non appartengono alla logica applicativa del software.
- **Logica Interna del Sistema di Pagamento Esterno**: elaborazione effettiva del credito bancario al di fuori dell'invio dei dati e della ricezione dell'esito.
  Motivazione: il sistema si interfaccia con servizi esterni; viene testata solo l'interfaccia di comunicazione e la gestione dei risultati (successo, fallimento, timeout).
- **Compatibilità con Browser Obsoleti**: browser che non supportano i moderni standard HTML, CSS e JavaScript.
  Motivazione: il sistema è progettato per supportare specificamente le versioni correnti di Chrome, Edge, Firefox e Safari.

5 Criteri di Superamento/Fallimento

In questa sezione vengono stabiliti i criteri generici per determinare l'esito dei test. Un test è considerato "di successo" quando riesce a causare un fallimento (failure), rivelando la presenza di un bug. Al contrario, un test che non rileva anomalie conferma la conformità del componente alla specifica, ma non ne garantisce l'assenza totale di difetti.

Un test ha successo (pass) se, dato un input al sistema, l'output ottenuto è diverso dall'output atteso dall'oracolo. Un test fallisce (fail) se, dato un input al sistema, l'output ottenuto è uguale all'output atteso dall'oracolo.

5.1 Criteri di Successo del Test (Identificazione Fallimenti)

Un test individuale è considerato riuscito se si verifica una delle seguenti condizioni:

- **Deviazione dalla Specifica**: il comportamento osservato del sistema differisce dal comportamento specificato nei requisiti funzionali (RF1-RF5).
- **Stato Erroneo**: il sistema entra in uno stato tale per cui l'elaborazione successiva porterà inevitabilmente a un fallimento.
- **Violazione dei Requisiti Non Funzionali (RNF)**: il sistema non rispetta i vincoli di performance, sicurezza o affidabilità.

5.2 Criteri di Accettazione del Sistema

Affinché il sistema o un sottosistema superi la fase di verifica e sia considerato pronto per l'uso, devono essere soddisfatti i seguenti parametri qualitativi:

A. Requisiti Funzionali

RF1-RF5: tutte le operazioni principali (gestione veicoli, prenotazioni, ricerca, vendite e storico) devono operare senza errori algoritmici o meccanici.

Robustezza: il sistema deve gestire correttamente gli input errati o non validi, notificando l'utente senza compromettere l'integrità dei dati.

B. Requisiti Non Funzionali (RNF)

Prestazioni (RNF1): ogni richiesta deve ricevere risposta in un tempo inferiore ai 2 secondi.

Affidabilità (RNF5): il sistema deve garantire una disponibilità del 99%.

Ripristino: in caso di errore critico, il tempo di ripristino dell'operatività non deve superare i 10 minuti.

Sicurezza (RNF4): tutte le comunicazioni sensibili devono avvenire tramite HTTPS e le password devono risultare cifrate nel database.

5.3 Gestione degli Incidenti

Ogni volta che un test risulta "di successo" (ovvero trova un bug), viene generato un Test Incident Report. Questo documento descrive l'incidente per permettere agli sviluppatori di:

- analizzare e dare priorità al fallimento;
- pianificare le modifiche necessarie al sistema;

- eseguire i test di regressione per assicurarsi che la correzione non abbia introdotto nuovi difetti.

6 Approccio

Questa sezione descrive la metodologia generale adottata per il processo di verifica del sistema Easy Drive, dettagliando la strategia di integrazione selezionata e le tecniche di supporto utilizzate.

6.1 Strategia di Testing Generale

Il processo segue un modello iterativo in cui l'implementazione del codice e il testing procedono di pari passo. Le attività sono suddivise in quattro fasi principali condotte dagli sviluppatori e, nella fase finale, dal cliente:

- Unit Testing: verifica dei singoli componenti isolati.
- Integration Testing: test delle interfacce tra i sottosistemi.
- System Testing: verifica dell'intero sistema rispetto ai requisiti funzionali e globali.
- Acceptance Testing: valutazione finale condotta dal cliente per dimostrare che il sistema è pronto all'uso.

6.2 Strategia di Integrazione

Per Easy Drive è stata scelta una strategia di integrazione Bottom-up.
Motivazione: questa scelta è giustificata dall'architettura del sistema, dove i sottosistemi applicativi dipendono fortemente dal package core e dal Data Layer (DBMS MySQL). Iniziando dai componenti di base, è possibile verificare la persistenza dei dati prima di integrare le logiche di business più complesse come le vendite o i noleggi.
Supporto ai Test: poiché i componenti vengono integrati dal basso verso l'alto, si renderà necessario l'uso di Test Drivers. Questi sono programmi parziali che simulano il comportamento dei componenti di livello superiore per esercitare i sottosistemi già pronti (come il catalogo o la gestione utenti).

6.3 Dipendenze tra i Test

L'ordine dei test è dettato dalla gerarchia dei sottosistemi definita nel SDD. Di seguito sono elencate le dipendenze principali:

- I test del sottosistema Pagamenti devono essere integrati con i sottosistemi Vendite e Prenotazioni, poiché questi ultimi dipendono dall'esito della transazione per confermare l'operazione.
- Il sottosistema Gestione Catalogo deve essere verificato prima della Ricerca Avanzata, in quanto la presenza di dati validi inseriti dai responsabili è un prerequisito per i test di filtraggio dei clienti.
- Tutti i test funzionali che richiedono un utente loggato (es. creazione prenotazione) dipendono dal successo dei test del sottosistema Gestione Utenti.

6.4 Tecniche di Selezione dei Casi di Test

Per garantire una copertura ottimale senza eccessivi costi, vengono applicate le seguenti tecniche:

- **Black-box Testing**: utilizzata per il Functional Testing basandosi esclusivamente sui requisiti del RAD.
- **Equivalence Partitioning**: i dati di input (es. fasce di prezzo o date di noleggio) vengono suddivisi in classi di equivalenza per testare solo i valori rappresentativi e i valori limite (boundary values).

7 Sospensione e Ripresa

Questa sezione specifica i criteri necessari per sospendere temporaneamente le attività di test e le condizioni indispensabili per la loro ripresa, al fine di ottimizzare le risorse ed evitare test inutili su componenti instabili.

7.1 Criteri di Sospensione

Le attività di testing verranno immediatamente sospese se si verifica una delle seguenti condizioni critiche:

- **Bug Bloccanti** : identificazione di un difetto che impedisce l'esecuzione di un intero flusso di lavoro fondamentale (es. impossibilità di accedere al sistema o crash del database MySQL durante la ricerca dei veicoli).
- **Instabilità dell'Ambiente**: malfunzionamento critico dell'infrastruttura server (Apache o interprete PHP) che impedisce la comunicazione tra Client e Server.
- **Violazione della Sicurezza**: rilevamento di vulnerabilità che espongono i dati sensibili degli utenti in chiaro o malfunzionamento dei certificati HTTPS.
- **Mancanza di Documentazione Corretta**: se viene rilevato che il codice non corrisponde alle specifiche dell'ODD o dell'SDD, rendendo i test non attendibili.

7.2 Criteri di Ripresa

Le attività di test potranno essere riprese solo dopo che i criteri di sospensione saranno stati risolti e validati. Le condizioni per la ripresa includono:

- **Correzione dei Difetti (Correction)**: gli sviluppatori devono aver applicato le modifiche necessarie per riparare il guasto che ha causato il fallimento.
- **Ripristino dell'Operatività**: in caso di errore hardware o software critico, l'operatività deve essere ripristinata entro un tempo massimo di 10 minuti, come specificato nei requisiti di affidabilità.
- **Esecuzione dei Test di Regressione**: prima di procedere con nuovi test, devono essere ri-eseguiti i test precedenti per assicurarsi che le correzioni non abbiano introdotto nuovi bug (regression testing).
- **Verifica dell'Integrità dei Dati**: conferma che il database sia stato ripristinato tramite backup giornalieri se la sospensione è stata causata da una perdita di dati.

8 Materiali di Test

In questa sezione vengono identificate le risorse hardware, software e gli strumenti di supporto necessari per allestire l'ambiente di prova e rendere eseguibili i casi di test definiti.

8.1 Risorse Hardware

Per garantire la validità dei test di sistema e di accettazione, l'hardware deve rispecchiare l'ambiente target descritto nel Problem Statement:

- **Lato Client (Postazioni Tester)**:
  - Computer desktop o laptop con almeno 8 GB di RAM.
  - Dispositivi mobili (Smartphone/Tablet) con sistemi operativi Android o iOS per testare la responsività della piattaforma.
  - Connessione internet stabile per le prove di latenza e performance.
- **Lato Server (Ambiente di Staging)**:
  - Server dedicato o istanza virtuale configurata per ospitare l'applicazione e il database.

8.2 Risorse Software

L'ambiente di test deve replicare fedelmente lo stack tecnologico scelto per lo sviluppo:

- **Sistemi Operativi**: Windows 10 o superiore, Linux (Ubuntu/CentOS), macOS.
- **Web Server**: Apache o Nginx.
- **Linguaggio di Scripting**: PHP (versione compatibile con il codice sorgente).
- **Database Management System**: MySQL per la gestione della persistenza dei dati.
- **Browser Web**: versioni aggiornate di Chrome, Firefox, Edge e Safari (per la verifica del requisito RNF6 - Portabilità).

8.3 Strumenti di Testing e Supporto

Per l'automatizzazione e la gestione dei report, verranno utilizzati i seguenti tool:

- **PHPUnit**: framework di riferimento per l'esecuzione degli Unit Test e per il calcolo della copertura del codice (Code Coverage).
- **Selenium** : per l'automazione dei test funzionali sulla UI (User Interface) e per i test di sistema.

- **Git / GitHub**: per il controllo di versione degli script di test e per garantire la tracciabilità delle modifiche.

9 Compiti (Tasks)

Questa sezione identifica i compiti specifici necessari per portare a termine il processo di testing e assegna le relative responsabilità ai vari ruoli coinvolti, come previsto dal ciclo di vita del test descritto nella documentazione tecnica.

9.1 Ruoli e Responsabilità

I compiti sono distribuiti tra i membri del team di sviluppo (Mascheri, Brizio, Cannella, Coscia) e il committente secondo il seguente schema:

- **Analista del Test (Test Analyst)**:
  - Identifica i componenti da testare basandosi sui requisiti del RAD e sul design dell'SDD/ODD.
  - Definisce i casi di test e le specifiche di input/output.
- **Tester Professionale / Sviluppatore**:
  - Esegue i test unitari e di integrazione (strategia Bottom-up).
  - Utilizza i driver per simulare i componenti di livello superiore.
  - Registra i risultati nel Test Log.
- **Programmatore (Programmer)**:
  - Responsabile della correzione dei difetti (faults) identificati.
  - Esegue i test di regressione dopo ogni modifica al codice.
- **Specialista della Configurazione (Configuration Management)**:
  - Gestisce l'ambiente di test e garantisce che il codice testato sia sincronizzato su GitHub.
  - Controlla che le versioni del database (MySQL) siano allineate tra sviluppo e test.
- **Cliente / Utente Finale (User)**:

- Partecipa attivamente ai Test di Accettazione (Alpha e Beta test).
- Valida che le funzionalità corrispondano alle aspettative espresse nel Problem Statement.

9.2 Matrice delle Attività

Di seguito sono elencate le attività di testing e i relativi output richiesti:

- **Pianificazione**
  Responsabile: Coordinatore Progetto
  Output: Piano di Test (questo documento)
- **Progettazione Test**
  Responsabile: Analista / Progettista
  Output: Test Case Specification
- **Esecuzione Test**
  Responsabile: Tester / Sviluppatori
  Output: Test Log / Test Execution Report
- **Segnalazione Bug**
  Responsabile: Tester
  Output: Test Incident Report

10 Pianificazione dei Tempi (Schedule)

Le attività di pianificazione del testing saranno avviate al termine della fase di progettazione, in linea con quanto descritto nei capitoli precedenti.

La definizione e la redazione dei casi di test procederanno in parallelo alle attività di sviluppo del software.

L'esecuzione delle attività di test avrà luogo a seguito del completamento dell'implementazione del sistema. Al termine dello sviluppo, l'intera suite di test verrà rieseguita al fine di verificare il corretto funzionamento dell'applicazione e predisporre la reportistica finale.

10.1 Test Cases

Per lo sviluppo dei test cases verrà utilizzato il metodo del *Category Partition*. Questo metodo consiste nell'identificare, per ciascuna funzionalità da testare, un insieme di parametri. Per ogni parametro vengono definite delle categorie, che vengono successivamente suddivise in scelte. A ciascuna scelta viene infine assegnato un valore.

I test cases verranno definiti nel paragrafo *Test Cases Specification*.

10.2 Funzionalità: Registrazione Utente

T.C 1.1

Obiettivo Verificare la corretta creazione di un nuovo account cliente.

Parametri

- email
- password
- conferma_password
- nome
- cognome

Categorie

- **Categoria 1**: Validità del formato email
- **Categoria 2**: Presenza dell'email nel database
- **Categoria 3**: Lunghezza della password
- **Categoria 4**: Corrispondenza tra password e conferma password

Scelte

- **EF1**: Formato email valido (es. user@domain.com)
- **EF2**: Formato email non valido (mancanza di `@` o del dominio)
- **EM1**: Email non presente nel database (nuovo utente)

- **EM2**: Email già presente nel database
- **PL1**: Lunghezza della password valida (≥ 8 caratteri)
- **PL2**: Lunghezza della password non valida (< 8 caratteri)
- **PC1**: Password e conferma password coincidono
- **PC2**: Password e conferma password non

Vincoli associati alle scelte

- **EF1**: Formato email valido [*property formato_valido*]
- **EF2**: Formato email non valido [*error*]
- **EM1**: Email non presente nel database [*if formato_valido*]
- **EM2**: Email già presente nel database [*error*] [*if formato_valido*]
- **PL1**: Lunghezza password valida [*property lunghezza_ok*]
- **PL2**: Lunghezza password non valida [*error*]
- **PC1**: Password e conferma password coincidono [*if lunghezza_ok*]
- **PC2**: Password e conferma password non coincidono [*error*] [*if lunghezza_ok*]

10.3 Funzionalità: Login (Autenticazione)

T.C 1.2

Obiettivo Verificare il corretto accesso al sistema.

Parametri

- email
- password

Categorie

- **Categoria 1**: Presenza dell'email nel database
- **Categoria 2**: Correttezza della password

Scelte

- **UE1**: Email presente nel database
- **UE2**: Email NON presente nel database
- **PW1**: Password corretta per l'utente specificato
- **PW2**: Password NON corretta

Vincoli associati alle scelte

- **UE1**: Email presente nel database [*property email_presente*]
- **UE2**: Email NON presente nel database [*error*]
- **PW1**: Password corretta per l'utente specificato [*if email_presente*]
- **PW2**: Password NON corretta [*error*] [*if email_presente*]

10.4 Funzionalità: Prenotazione Noleggio

T.C 1.3

Obiettivo Verificare la creazione di un ordine di noleggio.

Parametri

- data_inizio
- data_fine
- veicolo

Categorie

- **Categoria 1**: Coerenza temporale delle date (Inizio vs Fine)
- **Categoria 2**: Validità temporale rispetto ad oggi (Futuro)
- **Categoria 3**: Disponibilità del veicolo

Scelte

- **DT1**: Data Inizio < Data Fine

- **DT2**: Data Inizio > Data Fine
- **PST1**: Data Inizio è futura (o oggi)
- **PST2**: Data Inizio è nel passato
- **DISP1**: Veicolo disponibile per tutto il periodo
- **DISP2**: Veicolo occupato nel periodo

Vincoli associati alle scelte

- **DT1**: Data Inizio < Data Fine [*property sequenza_ok*]
- **DT2**: Data Inizio > Data Fine [*error*]
- **PST1**: Data Inizio è futura [*property futuro_ok*]
- **PST2**: Data Inizio è nel passato [*error*]
- **DISP1**: Veicolo disponibile per tutto il periodo [*if sequenza_ok*] [*if futuro_ok*]
- **DISP2**: Veicolo occupato nel periodo [*error*] [*if sequenza_ok*] [*if futuro_ok*]

10.5 Funzionalità: Pagamento

T.C 1.4

Obiettivo Verificare la corretta esecuzione della transazione economica.

Parametri

- numero_carta
- scadenza
- saldo

Categorie

- **Categoria 1**: Validità formale della carta
- **Categoria 2**: Validità della scadenza
- **Categoria 3**: Disponibilità fondi

Scelte

- **CD1**: Carta valida
- **CD2**: Numero carta errato
- **EXP1**: Data scadenza futura
- **EXP2**: Data scadenza passata
- **FND1**: Saldo sufficiente
- **FND2**: Saldo insufficiente

Vincoli associati alle scelte

- **CD1**: Carta valida [*property carta_ok*]
- **CD2**: Numero carta errato [*error*]
- **EXP1**: Data scadenza futura [*property data_ok*]
- **EXP2**: Data scadenza passata [*error*]
- **FND1**: Saldo sufficiente [*if carta_ok*] [*if data_ok*]
- **FND2**: Saldo insufficiente [*error*] [*if carta_ok*] [*if data_ok*]

10.6 Funzionalità: Ricerca Veicoli nel Catalogo

T.C 1.5

Obiettivo Verificare il corretto funzionamento dei filtri di ricerca per individuare un veicolo nel catalogo.

Parametri

- categoria (es. SUV, CityCar)
- marca
- prezzo_min
- prezzo_max

Oggetti dell'ambiente

- Database Catalogo Veicoli

Categorie

- **Categoria 1**: Validità della categoria selezionata
- **Categoria 2**: Validità della marca selezionata
- **Categoria 3**: Consistenza dell'intervallo di prezzo
- **Categoria 4**: Presenza di risultati corrispondenti nel database

Scelte

- **CT1**: Categoria valida ed esistente nel sistema (es. SUV)
- **CT2**: Categoria non valida o non esistente [error]
- **MK1**: Marca valida (es. Fiat, BMW)
- **MK2**: Marca contenente caratteri non validi o inesistente [error]
- **PR1**: Prezzo Min inferiore o uguale al Prezzo Max
- **PR2**: Prezzo Min superiore al Prezzo Max [error]
- **PR3**: Valori negativi nel prezzo [error]
- **DB1**: Esiste almeno un veicolo con le caratteristiche cercate
- **DB2**: Non esiste nessun veicolo con le caratteristiche cercate

Vincoli associati alle scelte

- **CT1**: Categoria valida [*property categoria_ok*]
- **CT2**: Categoria non valida [*error*]
- **MK1**: Marca valida [*property marca_ok*]
- **MK2**: Marca non valida [*error*]
- **PR1**: Prezzo consistente [*property prezzo_ok*]
- **PR2**: Prezzo Min > Max [*error*]
- **PR3**: Prezzo negativo [*error*]
- **DB1**: Veicoli trovati [*if categoria_ok*] [*if marca_ok*] [*if prezzo_ok*]
- **DB2**: Nessun veicolo trovato [*if categoria_ok*] [*if marca_ok*] [*if prezzo_ok*]

11 Test Frames

Di seguito sono riportati i frame di test generati dalla combinazione delle scelte identificate con il metodo Category Partition.

11.1 Test Frames - Registrazione Utente

- EF1, EM1, PL1, PC1 → **Corretto**
- EF2, -, -, - → **Errore**
- EF1, EM2, -, - → **Errore**
- EF1, EM1, PL2, - → **Errore**
- EF1, EM1, PL1, PC2 → **Errore**

11.2 Test Frames - Login (Autenticazione)

- UE1, PW1 → **Corretto**
- UE1, PW2 → **Errore**
- UE2, - → **Errore**

11.3 Test Frames - Prenotazione Noleggio

- DT1, PST1, DISP1 → **Corretto**
- DT1, PST1, DISP2 → **Errore**
- DT2, -, - → **Errore**
- -, PST2, - → **Errore**

11.4 Test Frames - Pagamento

- CD1, EXP1, FND1 → **Corretto**
- CD1, EXP1, FND2 → **Errore**
- CD2, -, - → **Errore**

- CD1, EXP2, - → **Errore**

11.5 Test Frames - Ricerca Veicoli

- CT1, MK1, PR1, DB1 → **Corretto**
- CT1, MK1, PR1, DB2 → **Corretto** (Nessun risultato)
- CT2, -, -, - → **Errore**
- -, -, PR2, - → **Errore**
- -, -, PR3, - → **Errore**