Università degli Studi di Salerno
Corso di Ingegneria del Software

AutoFlow
Test Plan
Versione 1.0

Data: 08/11/2025

Sommario

1. Introduzione

Il presente Test Plan illustra l'approccio scelto per verificare e validare il funzionamento del sistema AutoFlow, descrivendo in dettaglio le tecniche, gli strumenti e le procedure utilizzate durante le fasi di testing. L'obiettivo principale è accertare che la piattaforma rispetti pienamente i requisiti funzionali e non funzionali definiti nei documenti progettuali, garantendo un comportamento stabile, affidabile e adeguato all'uso quotidiano nelle concessionarie automobilistiche.

Il documento fornisce inoltre una visione d'insieme delle attività dei casi di test previsti e delle risorse necessarie per assicurare la qualità del software, con particolare attenzione ai flussi fondamentali del sistema, come la gestione delle proposte, la configurazione dei veicoli, l'amministrazione dello showroom e la generazione dei documenti PDF legati al ciclo di vendita.

2. Relazione con altri documenti

La progettazione e lo sviluppo di AutoFlow si basano sull'analisi dei processi operativi tipici delle concessionarie e sul confronto con soluzioni gestionali già consolidate nel settore automobilistico, che hanno dimostrato efficacia nella gestione centralizzata di showroom, clienti, trattative e documentazione commerciale.

Il presente Test Plan è strettamente collegato ai documenti principali del progetto:

- Problem Statement (PS) – Analizza le criticità presenti nelle concessionarie e definisce gli obiettivi che AutoFlow intende perseguire.
- Requirement Analysis Document (RAD) – Descrive requisiti funzionali, non funzionali, vincoli e casi d'uso che guidano le attività di test.
- System Design Document (SDD) – Illustra l'architettura software del sistema, l'organizzazione dei sottosistemi e le interazioni interne.
- Object Design Document (ODD) – Definisce il modello a oggetti, le classi principali, le interfacce e i trade-off architetturali alla base dell'implementazione.

Oltre alla documentazione progettuale, il lavoro fa riferimento a testi e metodologie consolidate nel campo dell'ingegneria del software, tra cui Object-Oriented Software Engineering Using UML, Patterns, and Java™ di Bruegge & Dutoit.

3. Panoramica sistema

Il sistema AutoFlow è concepito come piattaforma web moderna destinata alla gestione integrata delle attività tipiche delle concessionarie automobilistiche. L'architettura segue un modello three-tier, strutturato in interfaccia utente, logica applicativa e livello di persistenza dei dati. Questa suddivisione permette di isolare chiaramente le responsabilità dei vari componenti, rendendo più semplice individuare il livello su cui concentrare i test unitari e le verifiche di integrazione.

I diversi sottosistemi del progetto — dalla gestione dell'autenticazione al controllo dello showroom, dalla configurazione dei veicoli alla gestione delle proposte e delle fatture — sono progettati secondo principi di modularità e separazione delle responsabilità. Tale organizzazione permette di testare in modo mirato ciascun modulo, verificando sia il corretto funzionamento interno sia la loro interazione con i componenti vicini.

Un ruolo centrale è svolto dal livello dati, il quale deve garantire consistenza, integrità e prestazioni elevate anche in presenza di operazioni concorrenti tipiche delle attività di vendita. I test su questo livello si concentreranno sulla correttezza delle operazioni CRUD, sulla gestione dei vincoli (come unicità di targa/VIN o stato delle proposte) e sulla capacità del sistema di mantenere stabilità sotto carichi maggiori.

La chiara definizione dei confini tra i sottosistemi consente di garantire una copertura completa delle funzionalità fondamentali. Nel caso di AutoFlow, particolare attenzione verrà dedicata ai moduli di autenticazione, gestione utenti, gestione veicoli, configuratore, proposte di vendita e generazione dei documenti PDF, poiché costituiscono l'asse portante dell'intero sistema.

4. Funzionalità da essere testate / non testate

Nel sistema AutoFlow sarà data la massima priorità alla verifica delle funzionalità considerate cruciali per il corretto svolgimento delle attività operative della concessionaria, con particolare attenzione ai processi che richiedono input manuale da parte dell'utente o che influenzano dati sensibili. Ogni interazione rilevante sarà oggetto di controlli approfonditi, al fine di garantire accuratezza dei dati, continuità del servizio e piena aderenza ai requisiti funzionali e non funzionali definiti nella fase di analisi.

Di seguito sono elencate le principali funzionalità che saranno sottoposte a test:

Authentication

- Autenticazione utente (login con credenziali)

User Management

- Modifica dei dati anagrafici del cliente
- Cambio password utente

Proposte di vendita

- Creazione di una nuova proposta a partire da una configurazione veicolo

Fatturazione

- Generazione della fattura in formato PDF a partire da una proposta confermata

5. Criteri di successo/insuccesso

I criteri di successo e insuccesso per il piano di test sono stabiliti in riferimento al rispetto dei requisiti funzionali e non funzionali previsti per il sistema AutoFlow.

Un test è considerato **superato** quando i risultati effettivi coincidono con quelli attesi riportati nei casi di test e quando non emergono difetti critici o anomalie in grado di compromettere le funzionalità principali dell'applicazione.

In caso di esito positivo, tutte le funzionalità coinvolte devono risultare conformi ai requisiti stabiliti; inoltre, i test prestazionali devono mantenersi entro i limiti accettabili definiti e non devono verificarsi errori bloccanti o di alta priorità durante la loro esecuzione.

In caso di **insuccesso**, sono considerate critiche tutte le situazioni in cui i risultati ottenuti divergono da quelli previsti, la presenza di bug bloccanti o difetti di elevata gravità, nonché il mancato rispetto dei vincoli non funzionali, come tempi di risposta superiori alle soglie consentite o errori riscontrati in condizioni di carico elevato.

Ogni test non superato richiederà la localizzazione del difetto, il suo corretto trattamento e una successiva esecuzione di regression testing, così da verificare che le modifiche apportate non introducano nuovi problemi.

6. Approccio

L'approccio al testing adottato per AutoFlow segue una strategia strutturata e basata su metodologie consolidate, così da garantire una copertura completa e accurata delle funzionalità previste. In particolare, verrà utilizzata la tecnica del Category Partition Testing per individuare e organizzare le categorie di input rilevanti, i casi di test associati e le combinazioni significative, assicurando un'analisi sistematica dei requisiti.

Unit Testing

Per i test unitari verranno impiegati i framework JUnit e Mockito, che permettono di isolare le singole componenti del codice e verificarne il comportamento in condizioni controllate. L'utilizzo di mock consente di simulare le dipendenze esterne e garantire che ogni unità venga testata senza interferenze provenienti da altri moduli.

Integration Testing

Il testing di integrazione per AutoFlow adotterà un approccio di tipo Sandwich, che combina i vantaggi delle strategie bottom-up e top-down.

I test verranno eseguiti attraverso JUnit, mentre Mockito sarà utilizzato per simulare i componenti non ancora integrati. L'automazione dei test sarà gestita tramite Maven, che permetterà una gestione ordinata delle dipendenze e dei cicli di testing.

Il processo verrà articolato principalmente in tre fasi:

1. Verifica delle classi Repository, valutando il loro comportamento e l'interazione con il database.
2. Test delle classi Service, assicurandosi che la logica applicativa funzioni correttamente in combinazione con il livello dati.
3. Test dei Controller, utilizzando mock dei servizi sottostanti per controllare la coerenza delle risposte fornite dagli endpoint.

System Testing

Per il test di sistema verrà utilizzato Selenium, strumento che consente l'automazione delle interazioni con l'interfaccia utente e la verifica del comportamento complessivo della piattaforma in scenari realistici.

Attraverso Selenium sarà possibile simulare il flusso dell'utente — ad esempio durante una creazione di proposta o nella generazione di un documento PDF — verificando la corretta integrazione delle funzionalità e la conformità ai requisiti funzionali.

Le attività di testing saranno integrate all'interno di una pipeline di Continuous Integration / Continuous Deployment (CI/CD), realizzata tramite GitHub Actions.

Ad ogni commit o pull request, la pipeline eseguirà automaticamente l'intera suite di test — unitari, di integrazione e di sistema — consentendo di rilevare rapidamente eventuali regressioni o problemi introdotti durante lo sviluppo, garantendo così un feedback tempestivo agli sviluppatori e una maggiore stabilità del codice nel tempo.

7. Sospensione e ripresa

Il processo di testing potrà essere sospeso o ripreso in base a specifiche condizioni, così da garantire un utilizzo corretto delle risorse e assicurare che le verifiche vengano eseguite in un ambiente adeguato.

Criteri di sospensione

La sospensione delle attività di test avverrà nei casi in cui si presentino difetti critici o malfunzionamenti bloccanti che impediscano la prosecuzione delle prove, come errori che compromettono l'avvio del sistema o l'accesso alle funzionalità principali di AutoFlow.

Il testing sarà inoltre interrotto qualora l'ambiente di prova non sia disponibile o non funzioni correttamente, ad esempio per problemi sui server, indisponibilità del database o malfunzionamenti degli strumenti utilizzati per l'automazione.

Infine, le attività potranno essere sospese nel caso in cui vengano superati i limiti di tempo o budget previsti, rendendo necessaria una revisione della pianificazione.

Criteri di ripresa

Le attività di testing riprenderanno una volta che i difetti critici saranno stati individuati, corretti e verificati tramite test di regressione.

Il processo potrà inoltre essere ripreso quando l'ambiente di test sarà stato ripristinato e reso nuovamente operativo, garantendo l'accesso a tutte le risorse necessarie.

La ripresa delle attività avverrà anche al termine di eventuali revisioni della pianificazione o in seguito all'assegnazione di ulteriori risorse necessarie per proseguire con la campagna di testing.

8. Materiale di test (requisiti hardware/software)

Per garantire la qualità, l'affidabilità e la robustezza della piattaforma AutoFlow è essenziale disporre di un ambiente di testing adeguato, sia dal punto di vista hardware che software. Gli strumenti scelti supportano l'intero ciclo di vita del testing, agevolando l'esecuzione di test unitari, di integrazione e di sistema in maniera efficiente e ripetibile. L'utilizzo di framework e tool mirati consente di automatizzare buona parte delle verifiche, ridurre i tempi di sviluppo e individuare con rapidità eventuali anomalie o vulnerabilità. Di seguito sono riportati i requisiti hardware e software necessari per implementare una strategia di testing completa per AutoFlow.

Requisiti Hardware

Postazioni di Sviluppo e Testing:

- CPU: almeno 2 core
- RAM: minimo 8 GB
- Storage: SSD con capienza sufficiente a ospitare database MariaDB e file di log/generati (PDF, report, ecc.)
- Connessione Internet

Requisiti Software

- IDE (es. IntelliJ IDEA o VS Code) utilizzato per lo sviluppo e il testing del backend Java/Spring Boot e del frontend React/TypeScript di AutoFlow.
- Spring Boot: framework Java impiegato per lo sviluppo e l'esecuzione del backend, che fornisce il contesto applicativo e le funzionalità necessarie all'orchestrazione dei servizi di AutoFlow.
- Node.js e NPM: ambiente di esecuzione JavaScript e gestore di pacchetti utilizzati per installare le dipendenze del frontend e per eseguire gli script di build/test dell'interfaccia React/TypeScript.
- MariaDB: DBMS utilizzato per la gestione dei dati persistenti di AutoFlow; è necessario disporre di un'istanza di test dedicata, con un dataset rappresentativo, su cui eseguire le prove funzionali e di integrazione.
- JUnit: framework per l'implementazione e l'esecuzione dei test unitari sul codice Java del backend.
- Mockito: libreria per la creazione di mock e stub, utilizzata per isolare le dipendenze nei test unitari e di integrazione del backend.
- Selenium: strumento per l'automazione dei test sull'interfaccia web, utile per verificare il comportamento del sistema in scenari d'uso realistici dal punto di vista dell'utente finale.
- Postman: utilizzato per definire ed eseguire richieste verso gli endpoint REST di AutoFlow, facilitando il testing delle API e la verifica delle risposte del backend.

9. Agenda per il testing

Dopo il completamento della fase di progettazione, verrà avviata la pianificazione dettagliata delle attività di testing, con l'obiettivo di garantire una copertura completa e accurata delle funzionalità di AutoFlow. In ottica di Continuous Integration (CI), i casi di test saranno sviluppati parallelamente al codice, così da verificare il corretto funzionamento delle funzionalità fin dalle prime fasi di implementazione. Questo approccio permette di individuare e risolvere rapidamente eventuali problemi derivanti dall'integrazione tra i componenti.

Grazie all'adozione della CI, ogni volta che una funzionalità viene completata e integrata nel branch principale, essa verrà sottoposta automaticamente a una serie di test automatizzati — tra cui test unitari e di integrazione — al fine di verificare la coerenza del sistema rispetto ai requisiti specificati. La pipeline di CI provvede all'esecuzione automatica dell'intera suite test ad ogni integrazione, realizzando un regression testing continuo.

Questo garantisce che eventuali modifiche recenti non introducano nuovi difetti o regressioni nel comportamento delle funzionalità già implementate.

10. Glossario

Termine	Definizione
Category Partition Testing	Tecnica utilizzata per individuare e organizzare categorie di input rilevanti e generare combinazioni di test che coprano efficacemente i requisiti.
Unit Testing	Test che verificano il comportamento delle singole unità di codice in isolamento, spesso mediante strumenti come JUnit e Mockito.
JUnit	Framework per l'esecuzione dei test unitari in Java, utilizzato per verificare la correttezza del codice del backend AutoFlow.
Mockito	Libreria che permette di creare mock e stubs per simulare dipendenze e isolare le componenti durante i test unitari.
Integration Testing	Test che verificano l'interazione tra più componenti del sistema, assicurando il corretto funzionamento della logica di business e dell'accesso ai dati.
Sandwich Approach	Strategia di testing ibrida che combina tecniche bottom-up e top-down per testare più livelli dell'architettura in parallelo.
Maven	Strumento per la gestione delle dipendenze e l'automazione del build e dei test nel progetto AutoFlow.
System Testing	Tipologia di test orientata a verificare il comportamento complessivo del sistema in condizioni d'uso realistiche.
Selenium	Strumento utilizzato per automatizzare e validare le interazioni dell'utente con l'interfaccia web di AutoFlow.
Continuous Integration / Continuous Deployment (CI/CD)	Pipeline automatizzata che integra le modifiche al codice e avvia automaticamente la suite di test a ogni aggiornamento.
GitHub Actions	Servizio CI/CD utilizzato per eseguire automaticamente i test di AutoFlow ad ogni commit o pull request.
Postman	Strumento per l'invio di richieste HTTP e la verifica degli endpoint REST del backend durante il testing delle API.
IntelliJ IDEA	IDE utilizzato per lo sviluppo della componente backend Java/Spring Boot e per l'esecuzione dei test.
Spring Boot	Framework Java che fornisce strumenti e configurazioni per lo sviluppo del backend AutoFlow.
NPM	Gestore di pacchetti utilizzato per installare e mantenere le dipendenze del frontend basato su React e TypeScript.

MariaDB	DBMS utilizzato per la gestione dei dati persistenti nel progetto AutoFlow, adottato sia per l'ambiente di sviluppo che per i test.