Università degli Studi di Salerno

Corso di Ingegneria del Software

Auto Shop

Unit Test Plan Versione 1.0

Data: 12/12/2017

Cognome Nome	Matricola
Corrado Mancino Alfredo	0512102506
Carbè Daniele	0512102326
Caloia Gennaro	0512102332

Indice

1. Introduzione
Lo scopo di questo documento è quello di analizzare e gestire lo sviluppo e le attività di testing riguardanti il software Auto shop. Questa sessione di lavoro deve verificare il corretto funzionamento del sistema in diversi casi, studiati appositamente per mettere alla prova ogni singola funzionalità e caratteristica del sistema, al fine di ottenere un corretto funzionamento. I risultati di questi test saranno utilizzati per capire dove bisognerà intervenire, e quindi correggere eventuali errori o apportare modifiche per il miglioramento dei vari sottosistemi. Il processo verrà iterato fino a che non si otterranno i risultati attesi in accordo con i tempi di sviluppo previsti.

2. Relazione con gli altri documenti
Per verificare il corretto funzionamento del sistema sono stati predisposti dei test cases basati sulle funzionalità individuate sia nel documento inerente gli scenari che su quello di analisi dei requisiti (RAD).
Il documento a cui facciamo riferimento è quindi:
RAD 2.0;

3. Panoramica del sistema
Il sistema ha come obiettivo la creazione di sito web che permette la richiesta di preventivo per un' automobile scelta e l'acquisto di pezzi di ricambio.
Il suo sviluppo è stato concepito in maniera modulare e pertanto è stato possibile dividerlo in sottosistemi (si veda System Design Document); questo approccio apporta vantaggi dal punto di vista della manutenzione e sarà fondamentale nella fase di testing, in quanto si potrà individuare un errore nel singolo sottosistema effettuando un numero limitato di test.
Una volta testati individualmente tutti i sottosistemi andremo a testare la loro integrazione utilizzando un approccio di tipo "Bottom- up". Il sottosistema a livello gerarchico più basso sarà testato individualmente. Successivamente, sarà testata la sua integrazione con la componente a livello di gerarchia più alto. Questo procedimento sarà iterato finché non avremo integrato tutti i sottosistemi per verificare che l'intero sistema funzioni correttamente.

4. Funzionalità da testare e non

Le componenti prese in considerazione nella fase di testing rappresentano le funzionalità core del sistema, ovvero:

- ***Gestione Account***

Questa funzionalità permette ad un utente di effettuare l'accesso e di fare il logout dal sistema; permette agli admin, invece, di autenticarsi ed accedere alle funzionalità a loro consentite.

➢ Saranno testate le funzionalità di login.

- ***Inserimento ricambio***

Tale sezione consente di inserire un nuovo pezzo di ricambio nel sito web. Questo task viene effettuato dall'operatore magazzino

➢ Sarà testata la funzionalità di inserimento delle informazioni relative ai pezzi di ricambio da registrare nel database del sito web.

- ***Inserimento dipendente***

Tale sezione consente ad un Amministratore di inserire un nuovo dipendente in un database

➢ Sarà testata la funzionalità relative alla registrazione di un dipendente fatta da un amministratore.

.

5. Pass/Fail criteria

Il testing ha successo se l'output osservato è diverso dall'output atteso: ciò significa che la fase di testing avrà successo se individuerà un fallimento. In tal caso questa verrà analizzata e, se legata ad un fault, si procederà alla sua correzione. Sarà infine iterata la fase di testing per verificare che la modifica non abbia impattato su altri componenti del sistema. Al contrario, il testing fallirà se l'output osservato sarà uguale all'oracolo.

6. Approccio

Nella sessione di testing del sistema verrà utilizzato un approccio di tipo “BLACK BOX”, che
prevede che i test vengano effettuati in maniera da non scendere nei dettagli del codice, ma basandosi sulle specifiche delle funzionalità da testare.
L’approccio alla fase di testing si compone di tre fasi:

- Testing di unità, che controlla i singoli componenti (classi, metodi)
- Testing di integrazione, che va a testare l'integrazione dei vari sottosistemi
- Testing di sistema: test funzionale, che andrà a verificare la funzionalità dell’intero sistema assemblato.

6.1 Testing di unità

Con il testing di unità verrà effettuato un controllo delle varie classi e metodi del sistema, quindi saranno ricercate le condizioni di fallimento andando ad evidenziare gli errori. Il testing di unità, sarà eseguito dal team di sviluppo attraverso l’implementazione di classi di test utilizzando il framework JUnit. In particolare, per ogni classe che esegue operazioni complesse sarà sviluppata la relativa classe JUnit.

6.2 Testing di integrazione

Con il testing di integrazione si effettuerà un controllo sull’integrazione delle varie componenti del sistema. Si adotterà una strategia di tipo “Bottom- up”. Per effettuare questi test di integrazione, spesso saranno necessari l’utilizzo di driver dato che tale strategia va ad integrare passo passo i sottosistemi partendo dal layer che si trova più in basso nella scala gerarchica. La lista dei casi di test verrà fuori dall’applicazione del metodo del Category Partition.

6.3 Testing di sistema

Con il testing di sistema verrà effettuato un controllo della correttezza dell’intero sistema. E’ da considerare il testing più critico, in quanto può risultare molto complesso andare alla ricerca di eventuali errori, essendo impegnati tutti i sottosistemi. Questo test sarà effettuato utilizzando il framework Selenium, che mette a disposizione strumenti per il controllo di sistemi web- based.

7. Test cases

Per sviluppare i test cases sarà utilizzato il metodo del Category Partition. Questo metodo consiste nell'identificare per ogni funzionalità da testare dei parametri; per ogni parametro verranno individuate delle categorie, le quali poi saranno suddivise in scelte. Alle scelte verrà assegnato un valore.
I test cases verranno definiti nel documento di Test Cases Specification (TCS).

8.2 Gestione Autenticazione

8.2.1 Login

Parametro -> Email **Formato fe:** (?:[a-z0-9!#$%&'*+/=?^_`{\|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?|[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\]) **Categorie**	**Scelte**
Lunghezza Email	1: lunghezza < 5 [errore]
formato email	1: rispetta il formato [if **lunghezza<5**] e rispetta il formato fe 2: non rispetta il formato [if **lunghezza<5**] o non rispetta il formato fe [errore]

8.2 Gestione Dipendente
8.4.1 Inserimento Dipendente

Parametro -> Nome **Formato** **[A-Za-z]** **Categorie**	**Scelte**
Lunghezza ln	1: lunghezza < 2 [errore] 2: lunghezza 2-25 [property **lunghezzaLNok**, lunghezza nome da 2 a 25] 3: lunghezza > 25 [errore]
formato fn	1: rispetta il formato [if **lunghezzaLNok**] [property **formatoFNok,** rispetta il formato [A-Za-z]] 2: non rispetta il formato [if **lunghezzaLNok**] [errore]

Parametro -> Cognome **Formato** **[A-Za-z]** **Categorie**	**Scelte**
Lunghezza lc	1: lunghezza < 2 [errore] 2: lunghezza 2-30 [property **lunghezzaLCok**, lunghezza cognome da 2 a 30] 3: lunghezza > 30 [errore]
formato fc	1: rispetta il formato [if **lunghezzaLCok**] [property **formatoFCok**, rispetta il formato [A-Za-z]] 2: non rispetta il formato [if **lunghezzaLCok**] [errore]

Parametro -> Username **Formato** **[A-Za-z]**	

Categorie	**Scelte**
Lunghezza lus	1: lunghezza < 2 [errore] 2: lunghezza 2-16 [property **lunghezzaLUsok**, lunghezza cognome da 2 a 16] 3: lunghezza > 16 [errore]
formato fus	1: rispecchia il formato [if **lunghezzaLUsok**][property **formatoFUsok**,rispecchia il formato [A-Za-z] 2: non rispetta il formato [if **lunghezzaLUsok**][errore]

Parametro -> Password **Formato** **[A-Za-z]** **[0-9]** **Categorie**	

	Scelte
Lunghezza lus	1: lunghezza <8 [errore] 2: lunghezza 8-16[property **lunghezzaLNTok**, lunghezza Password tra 8 e 16] 3: lunghezza > 16 [errore]
formato fus	1: rispecchia il formato [if **lunghezzaLUsok**][property **formatoFUsok**,rispecchia il formato [A-Za-z] 2: non rispetta il formato [if **lunghezzaLUsok**][errore]