****

**Test Plan**

**Partecipanti del progetto**

| **Nome** | **Matricola** | **E-mail** |
| --- | --- | --- |
| **Ermanno Allocco** | **0512109570** | [**e.allocco1@studenti.unisa.it**](mailto:e.allocco1@studenti.unisa.it) |
| **Melania Leonelli** | **0512110395** | [**m.leonelli3@studenti.unisa.it**](mailto:m.leonelli3@studenti.unisa.it) |
| **Salvatore Scafa** | **0512113101** | [**s.scafa1@studenti.unisa.it**](mailto:s.scafa1@studenti.unisa.it) |

**1.Introduzione**

Nel documento in oggetto verranno definiti gli approcci e le attività di testing in merito alla piattaforma web LunchTIme. Verranno identificate le funzionalità della piattaforma da testare o meno, approcci e strumenti da utilizzare per l’attività di testing.  
L’obiettivo principale del testing è quello di identificare e prevedere quanti più difetti possibili di un’applicazione, questo tramite i suoi malfunzionamenti per evitare che essi possano verificarsi durante il normale funzionamento.  
Per cui, diremo che le attività di testing avranno avuto successo se riusciremo a identificare quanti più possibili failure presenti nel sistema prima che il software venga messo in esercizio.

**2.Documenti correlati**

Il test planning è in stretta relazione con i documenti prodotti fino ad ora; infatti, i test case sono basati sulle funzionalità individuate in questi, particolarmente:

**2.1 Relazione con il documento di raccolta ed analisi dei requisiti (RAD)**

La relazione tra TP e RAD riguarda i requisiti funzionali e non funzionali del sistema, infatti il RAD contiene la descrizione dettagliata delle funzionalità nella quale è indicata anche la priorità.

**2.2 Relazione con il System Design Document (SDD)**

Nell’ SDD è presente l'architettura del sistema (MVC), la struttura dei dati e i servizi dei sottosistemi.

**2.3 Relazione con l’Object Design Document (ODD)**

Nell’ODD sono presenti i package e le classi del sistema

**3.Funzionalità da testare**

Di seguito le funzionalità introdotte nel sistema che saranno sottoposte a test, suddivise per ogni gestione del sistema:

**Gestione autenticazione**

* Registrazione
* Login

**Gestione catalogo**

* Aggiungi Prodotto al Catalogo
* Modifica Prodotto Catalogo

**Gestione carrello**

* Procedere con l’Ordine

**4.Funzionalità da non testare**

* Rimuovere Prodotto dal Carrello
* Aggiungere Prodotto Carrello
* Rimuovi Prodotto Catalogo
* Aggiorna Menu del Giorno

**5.Criteri Pass / Fail testing**

Lo scopo del testing è trovare una failure nel programma, quindi, il testing va a buon fine se l’output osservato è diverso dall’output atteso; una volta riscontrata una failure (nel caso sia presente nel sistema), si interviene per correggerla e si effettua di nuovo il testing per appurare che non abbia prodotto effetti collaterali, producendo altre failure.

**6.Approccio**

L’approccio della fase di testing si compone di tre fasi.  
Si inizia con il testing di unità che ha lo scopo di testare le componenti del sistema singolarmente, poi si passa al testing di integrazione in cui le componenti del sistema verranno combinate e testate come un unico gruppo ed infine, si effettua il testing di sistema che mira a verificare il funzionamento dell’intero sistema  
  
**Testing di unità**  
Per il testing di unità si utilizzerà la tecnica “BLACK-BOX”. Con il “BLACK-BOX” testing ci focalizzeremo sul comportamento dell’input/output delle singole componenti senza tener conto della loro struttura interna. A causa della mancanza di fattibilità di effettuare un test esaustivo per l’ingente quantitativo di dati di input, verrà utilizzata la strategia del “Category Partition”, che consente di decomporre lo spazio di input in categorie per poi partizionare le “categorie” in classi di equivalenza chiamate “scelte”. Al termine saranno specificate le “combinazioni” delle scelte da testare creando delle istanze di casi di test specificando i valori dei dati effettivi per ciascuna scelta e determinare i risultati corrispondenti. Mediante il “Category Partition” otterremo, quindi, un test efficiente e privo di ridondanze.

**Testing di integrazione**  
In questa fase le singole unità vengono combinate e testate come gruppo. Per poter effettuare l’integration test è stata scelta la strategia bottom-up, in quanto consente di poter iniziare l’attività di testing non appena il primo modulo è stato specificato. Questo approccio richiede la costruzione di driver per simulare l’ambiente chiamante. In generale però, può portare alla problematica che i moduli possano essere codificati senza avere una chiara idea di come dovranno essere connessi ad altre parti del sistema. La riusabilità del codice è uno dei principali benefici dell’approccio bottom-up. Nonostante questa strategia di testing di integrazione abbia alcune limitazioni, risulta essere la più semplice e naturale forma con cui eseguire questo tipo di testing  
  
**Testing di sistema**  
Prima della messa in esercizio del sistema verrà effettuata una fase di testing di sistema per dimostrare che i requisiti commissionati dal cliente sono soddisfatti. In questo tipo di testing andremo a testare le funzionalità usate più frequentemente dal cliente e quelle che presentano maggiore possibilità di fallimento. Trattandosi di una applicazione web verrà utilizzato il tool Selenium, che si occuperà di simulare l’interazione con il sistema dal punto di vista dell’utente.

**7.Sospensione**

La fase di testing può essere sospesa se si riscontrasse un difetto nel sistema per essere poi ripresa dopo aver risolto il problema riscontrato.

**8.Materiale per i testing**

Gli strumenti utilizzati sono:

* MySQL
* Selenium IDE
* Web Server Apache Tomcat 9 in locale, il quale gira sul sistema
* Client-web per effettuare richieste al server

**9.Glossario**

* **errore:** una misura della differenza stimata tra il valore osservato o calcolato di una quantità e il suo valore reale
* **testing:** processo o metodo per trovare errori in un’applicazione o un programma software in modo che l’applicazione funzioni in base ai requisiti dell’utente finale
* **tool:** strumento software utilizzato per ottenere un dato risultato o semplificare delle operazioni

***TEST GESTIONE ACCOUNT***

**TC\_REGISTRAZIONE**

| Parametro: NomeUtente | |
| --- | --- |
| Categorie | Scelte |
| Lunghezza NomeUtente - LN | 1. Lunghezza = 0 – campo vuoto [property invalidLNValue] 2. Lunghezza >=1 1 [property validLNValue] |
| Formato NomeUtente - FN | 1. Non rispetta il formato [property invalidFNValue] 2. Rispetta il formato [if validNUValue] [property validFNValue] |

| Parametro: Password | |
| --- | --- |
| Categorie | Scelte |
| Lunghezza Password - LP | 1. Lunghezza =0 – campo vuoto [property invalidLPValue] 2. Lunghezza >=5 e <=8 [property validLNValue] 3. Lunghezza >0 ma <5 o >8 [property invalidLPValue] |

| Parametro: email | |
| --- | --- |
| Categorie | Scelte |
| Lunghezza email - LE | 1. Lunghezza = 0 – campo vuoto [property invalidLEValue] 2. Lunghezza >= 1 [property validLEValue] |
| Formato email - FE | 1. Non rispetta il formato [property invalidLEValue] 2. Rispetta il formato [if validLEValue] [property validFEValue] |
| Esistenza nel database - EXEM | 1. Non esiste nel database [property NotEXEM]  2. esiste nel database [if validFEValue] [property EXEM] |

| Parametro: ruolo | |
| --- | --- |
| Categorie | Scelte |
| selezione ruolo- SC | 1. 0 elementi selezionato [property invalidLEValue] 2. 1 elemento selezionati [property validLEValue] |

| Parametro: codiceTessera | |
| --- | --- |
| Categorie | Scelte |
| Lunghezza codiceTessera- LT | 1. Lunghezza = 0 – campo vuoto [property invalidLEValue] 2. Lunghezza >= 1 [property validLEValue] |

| CODICE | COMBINAZIONE | ESITO |
| --- | --- | --- |
| TC\_RegistrazioneOK | LN2.FN2.EXCA2.LP2.LE2.FE2.EXEM2.SC2.LT2 | “Registrazione avvenuta con successo!” |
| TC\_RegistrazioneFailed1 | LN1.FN2.EXCA2.LP2.LE2.FE2.EXEM2.SC2.LT2 | “NomeUtente campo obbligatorio” |
| TC\_RegistrazioneFailed2 | LN2.FN2.EXCA1.LP2.LE2.FE2.EXEM2.SC2.LT2 | “NomeUtente non rispetta il formato” |
| TC\_RegistrazioneFailed3 | LN2.FN2.EXCA2.LP1.LE2.FE2.EXEM2.SC2.LT2 | “password campo obbligatorio” |
| TC\_RegistrazioneFailed4 | LN2.FN2.EXCA2.LP2.LE1.FE2.EXEM2.SC2.LT2 | “email obbligatoria ” |
| TC\_RegistrazioneFailed5 | LN2.FN2.EXCA2.LP2.LE2.FE1.EXEM2.SC2.LT2 | “email non rispetta il formato” |
| TC\_RegistrazioneFailed6 | LN2.FN2.EXCA2.LP2.LE2.FE2.EXEM1.SC2.LT2 | “email esistente sul database” |
| TC\_RegistrazioneFailed8 | LN2.FN2.EXCA2.LP2.LE2.FE2.EXEM2.SC1.LT2 | “Selezionare un elemento” |
| TC\_RegistrazioneFailed9 | LN2.FN2.EXCA2.LP2.LE2.FE2.EXEM2.SC2.LT1 | “Codice tessera obbligatoria” |
| TC\_RegistrazioneFailed10 | LN2.FN2.EXCA2.LP3.LE2.FE2.EXEM2.SC2.LT1 | “Lunghezza password errata” |

**TC\_LOGIN**

| Parametro: Email | |
| --- | --- |
| Categorie | Scelte |
| Esistenza nel database - EXCA | 1. Non esiste nel database [property NotEXEM]  2. esiste nel database [if validFEValue] [property EXEM] |

| Parametro: Password | |
| --- | --- |
| Categorie | Scelte |
| Associazione all’utente- AS | 1. La password non è associata all’utente in sessione [property invalidAS Value]  2. La password è associata all’utente in sessione [property validASValue] |

| CODICE | COMBINAZIONE | ESITO |
| --- | --- | --- |
| TC\_LoginOK | EXCA2.AS2 | “Login avvenuta con successo!” |
| TC\_LoginFailed1 | EXCA1.AS2 | “email non esistente” |
| TC\_LoginFailed2 | EXCA2.AS1 | “password errata” |

***TEST GESTIONE CATALOGO***

**TC\_AGGIUNGI\_PRODOTTO\_AL\_CATALOGO**

| Parametro: idProdotto | |
| --- | --- |
| Categorie | Scelte |
| Lunghezza idProdotto- LA | 1. Lunghezza = 0 – campo vuoto [property invalidLAValue] 2. Lunghezza >=1 [property validLAValue] |

| Parametro: prezzoAcquisto | |
| --- | --- |
| Categorie | Scelte |
| prezzoAcquisto- LP | 1. prezzoAcquisto non valido – campo vuoto [property invalidLPValue] 2. prezzoAcquisto valido [property validLPValue] |

| Parametro: Descrizione | |
| --- | --- |
| Categorie | Scelte |
| Lunghezza Descrizione- LD | 1. Lunghezza = 0 – campo vuoto [property invalidLDValue] 2. Lunghezza >=1 [property validLDValue] |

| Parametro: Nome | |
| --- | --- |
| Categorie | Scelte |
| Lunghezza Nome- LN | 1. Lunghezza = 0 – campo vuoto [property invalidLDValue] 2. Lunghezza >=1 [property validLDValue] |

| CODICE | COMBINAZIONE | ESITO |
| --- | --- | --- |
| TC\_AggiungiProdottoAlCatalogoOK | LA2.LP2.LD2.LN2 | “Aggiunta prodotto al catalogo avvenuta con successo!” |
| TC\_AggiungiProdottoAlCatalogoFailed1 | LA1.LP2.LD2.LN2 | “id acquisto campo obbligatorio” |
| TC\_AggiungiProdottoAlCatalogoFailed2 | LA2.LP1.LD2.LN2 | “prezzo campo obbligatorio” |
| TC\_AggiungiProdottoAlCatalogoFailed3 | LA2.LP2.LD1.LN2 | “descrizione campo obbligatorio” |
| TC\_AggiungiProdottoAlCatalogoFailed3 | LA2.LP2.LD2.LN1 | “nome campo obbligatorio” |

**TC\_MODIFICA\_PRODOTTO\_DAL\_CATALOGO**

| Parametro: idProdotto | |
| --- | --- |
| Categorie | Scelte |
| Lunghezza idProdotto- LA | 1. Lunghezza = 0 – campo vuoto [property invalidLAValue] 2. Lunghezza >=1 [property validLAValue] |

| Parametro: prezzoAcquisto | |
| --- | --- |
| Categorie | Scelte |
| prezzoAcquisto- LP | 1. prezzoAcquisto non valido – campo vuoto [property invalidLPValue] 2. prezzoAcquisto valido [property validLPValue] |

| Parametro: Descrizione | |
| --- | --- |
| Categorie | Scelte |
| Lunghezza Descrizione- LD | 1. Lunghezza = 0 – campo vuoto [property invalidLDValue] 2. Lunghezza >=1 [property validLDValue] |

| Parametro: Nome | |
| --- | --- |
| Categorie | Scelte |
| Lunghezza Nome- LN | 1. Lunghezza = 0 – campo vuoto [property invalidLDValue] 2. Lunghezza >=1 [property validLDValue] |

| CODICE | COMBINAZIONE | ESITO |
| --- | --- | --- |
| TC\_ModificaiProdottoAlCatalogoOK | LA2.LP2.LD2.LN2 | “Modifica prodotto dal catalogo avvenuta con successo!” |
| TC\_ModificaiProdottoAlCatalogoFailed1 | LA1.LP2.LD2.LN2 | “id acquisto campo obbligatorio” |
| TC\_ModificaiProdottoAlCatalogoFailed2 | LA2.LP1.LD2.LN2 | “prezzo campo obbligatorio” |
| TC\_ModificaiProdottoAlCatalogoFailed3 | LA2.LP2.LD1.LN2 | “descrizione campo obbligatorio” |
| TC\_ModificaiProdottoAlCatalogoFailed3 | LA2.LP2.LD2.LN1 | “nome campo obbligatorio” |

***TEST GESTIONE CARRELLO***

**TC\_PROCEDERE\_CON\_LORDINE**

| Parametro: Saldo | |
| --- | --- |
| Categorie | Scelte |
| Saldo - S | 1. Saldo insufficiente [property invalidSValue] 2. Saldo>=ImportoTotale [property validSValue] |

| Parametro: Saldo | |
| --- | --- |
| Categorie | Scelte |
| stato carrello- SC | 1. carrello non contiene prodotti[property invalidSCValue] 2. carrello contiene prodotti [property validSCValue] |

| CODICE | COMBINAZIONE | ESITO |
| --- | --- | --- |
| TC\_ProcedereConLOrdineOK | S2.SC2 | “Operazione avvenuta con successo!” |
| TC\_ProcedereConLOrdineFailed1 | S1.SC2 | “saldo insufficiente" |