Laurea Magistrale in informatica-Università di Salerno
Corso di *Gestione dei Progetti Software*- Prof.ssa F. Ferrucci

UniSeats
PRENOTA CON NOI IL TUO POSTO

Test Plan
Progetto UniSeats

Riferimento	
Versione	1.1
Data	18/01/2020
Destinatario	Dipartimento di Informatica dell'Università degli Studi di Salerno
Presentato da	Vincenzo Russo
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
24/12/2020	1.0	Stesura Documento	V. Russo
18/01/2021	1.1	Aggiornamento	V. Russo

Team Composition

Nome	Ruolo	Posizione	Contatti
Filomena Ferrucci	Top Manager	Rappresentante del cliente	fferrucci@unisa.it
Vincenzo Russo	Project Manager	Manager	v.russo66@studenti.unisa.it
Adil El Yousfi	Team Member	Sviluppatore	a.elyousfi@studenti.unisa.it
Alessia Sabia	Team Member	Sviluppatore	a.sabia15@studenti.unisa.it
Matteo Ercolino	Team Member	Sviluppatore	m.ercolino1@studenti.unisa.it
Daniele Salerno	Team Member	Sviluppatore	d.salerno8@studenti.unisa.it
Benedetto Simone	Team Member	Sviluppatore	b.simone@studenti.unisa.it
Simone Silvestri	Team Member	Sviluppatore	s.silvestri15@studenti.unisa.it

Sommario

Revision History 2
Team Composition 3
1. Introduzione 5
1.1. Definizioni, Acronimi e Abbreviazioni 5
1.1.1. Definizioni 5
1.1.2. Acronimi e Abbreviazioni 5
1.1.3. Riferimenti 6
2. Documenti Correlati 7
3. Panoramica del Sistema 7
4. Possibili Rischi 8
5. Funzionalità da Testare 9
6. Funzionalità da Non Testare 9
7. Approccio 9
8. Criteri di Pass/Fail 9
9. Deliverables 10
10. Responsabilità 10
11. Glossario 10

1. Introduzione

In questo documento verranno definiti gli approcci e le attività di testing riguardanti la piattaforma web UniSeats. Saranno identificati gli approcci e gli strumenti utilizzati per l'attività di testing e i relativi elementi che faranno parte di questa fase. L'intento del testing è rilevare anticipatamente fault e failure presenti nel sistema, in modo tale che questi errori non si presentino durante la messa in esercizio del sistema.

Andremo ad analizzare gli elementi con priorità media o alta:

- Gestione Prenotazione;
- Gestione Utente;
- Gestione Accesso;

1.1. Definizioni, Acronimi e Abbreviazioni

1.1.1. Definizioni

- **Branch Coverage:** metodo che richiede che tutti i rami del programma o gli stati condizionali vengano testati almeno una volta durante un processo di test;
- **Failure:** mancata prestazione di un servizio atteso;
- **Fault:** causa di un failure, insieme di informazioni che quando processate generano un
- fallimento;
- **Model-View-Controller:** modello architetturale comunemente usato per lo sviluppo di interfacce utente che dividono un'applicazione in tre parti interconnesse. Ciò permette di separare le rappresentazioni interne di informazioni dai modi in cui le informazioni sono presentate e accettate dall'utente
- **Ottimalità:** Questa caratteristica si verifica se la soluzione trovata dall'algoritmo è la migliore possibile.

1.1.2. Acronimi e Abbreviazioni

- **RAD:** Abbreviazione utilizzata per indicare il Requirement Analysis Document;
- **SDD:** Abbreviazione utilizzata per indicare il System Design Document;
- **ODD:** Abbreviazione utilizzata per indicare il Object Design Document;
- **SOW:** Abbreviazione utilizzata per indicare lo Statement Of Work;
- **BC:** Abbreviazione utilizzata per indicare il Business Case;

- **TP:** Abbreviazione utilizzata per indicare il Test Plan;
- **STC:** Abbreviazione utilizzata per indicare il System Test Case;
- **MVC:** Abbreviazione utilizzata per indicare l'architettura Model-View-Controller;
- **REST:** Abbreviazione utilizzata per indicare il REpresentional State Transfer;
- **DB:** Abbreviazione utilizzata per indicare il Database;
- **API:** Abbreviazione utilizzata per indicare le Application Programming Interface.

1.1.3. Riferimenti

- *Cengage Learning – "Information Technology Project Management", Autori: Kathy Schwalbe;*
- *Prentice Hall – Pearson – Object-Oriented Software Engineering – Using UML, Patterns and Java. Autori: Bernd Bruegge & Allen H. Dutoit;*
- *Documentazione di Progetto:*
- *C06_RAD;*
- *C06_SDD;*
- *C06_SOW_Vers1.1;*
- *C06_TCS;*

2. Documenti Correlati

Il presente documento è in stretta relazione con i documenti prodotti fino al rilascio della versione 1.0 del Test Plan, esso è in forte relazione anche con documenti che verranno sviluppati e rilasciati in futuro quindi sarà soggetto a modifiche ed aggiornamenti. I test case sono basati sulle funzionalità individuate nel documento di raccolta ed analisi dei requisiti.

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

La relazione fra TP e RAD riguarda i requisiti funzionali e non funzionali del sistema, in quanto esso contiene la descrizione dettagliata delle funzionalità con scenari, use case, diagrammi e mock-ups e inoltre vi è indicata anche la priorità dei requisiti.

2.2. Relazione con il System Design Document (SDD)

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

2.3. Relazione con l'Object Design Document (ODD)

Nell'ODD (ancora non sviluppato al momento del rilascio dell'SE_TP_Vers.1.0) sono contenuti i package e le classi del sistema.

2.4. Relazione con lo Statement Of Work (SOW)

Nello Statement Of Work uno dei criteri di accettazione, è quello di ottenere una branch coverage del 75%, i test case verranno strutturati in modo tale da poter soddisfare tale criterio.

3. Panoramica del Sistema

Come definito nel System Design Document la struttura del nostro sistema segue l'architettura MVC (Model View Controller) che espone i servizi utilizzando un approccio REST.

La componente fondamentale di questa architettura è il controller. Per ogni sottosistema ci sarà un controller che si occuperà della logica di business della specifica gestione. Nel model verranno mappate le entità persistenti sul DB come oggetti. La view si occuperà di mostrare le interfacce utente.

L'approccio REST verrà implementato utilizzando l'oggetto router o endpoint. Esisterà un router per ogni sottosistema, il quale definirà una specifica URL sfruttando il nome del sottosistema e la firma di un metodo. Presso tale URL verranno esposti i risultati di una interrogazione al DB.

I sottosistemi sono:

- Gestione Prenotazione;
- Gestione Utente;
- Gestione Accesso;
- Prenotazione (View);
- Prenotazioni Effettuate (View);
- Profilo Utente (View);
- Login/Logout (View);
- Gestione Database.

4. Possibili Rischi

ID	Rischio	Risvolto	Probabilità	Impatto
R1	Pianificazione Inadeguata delle attività di testing	Negativo	Bassa	Alto
R2	Definizione dei casi di test poco chiara	Negativo	Bassa	Alto
R3	Project Manager poco vigile	Variabile	Media	Alto
R4	Le attività di testing proseguono oltre i tempi previsti	Negativo	Media	Alto
R5	I casi di test definiti non riescono a ottenere una branch coverage del 75%	Negativo	Media	Alto

5. Funzionalità da Testare

Come indicato in precedenza andranno testati solo i requisiti che presentano una priorità media o alta, di seguito l'elenco dei requisiti da testare per ogni gestione.
Le funzionalità da testare sono presenti nella Test Case Specification contenuta nel Test Case Document.

6. Funzionalità da Non Testare

Il modulo di IA non verrà testato sull'ottimalità della soluzione fornita.

7. Approccio

7.1. Testing di unità

In questa fase andremo a testare ogni singola funzione degli oggetti creati. Questa rappresenterà la nostra unità. Verrà utilizzato un approccio black-box, ovvero non sarà basato sulla conoscenza dell'architettura e del funzionamento interno di un componente ma sulle sue funzionalità esternamente esposte. Per tale fase utilizzeremo il tool JUnit e lo Spring Framework.

7.2. Testing di integrazione

In questa fase andremo a testare alcuni sottosistemi tra loro utilizzando il tool Katalon.

7.3. Testing di sistema

Non previsto.

8. Criteri di Pass / Fail

Una volta individuati i vari dati di input del test, questi verranno raggruppati in base a caratteristiche comuni in insiemi. In questo modo ci sarà possibile diminuire ed ottimizzare il numero di test.
La fase di test avrà successo se individuerà una failure, cioè se l'output osservato sarà diverso da quello atteso. Ogni qual volta verrà individuata una failure, questa verrà analizzata e, se legata ad un fault, si procederà alla sua correzione. Una volta completata la correzione, si procederà, in modo iterativo, ad una nuova fase di test per verificare che la modifica non ha impattato su altri componenti del sistema.
Al contrario, il testing fallirà se l'output osservato sarà uguale all'oracolo.

9. Deliverables

I documenti rilasciati durante e al termine della fase di test sono:

- Test Plan;
- Test Case Specification (contenuta nel Test Case Document);
- Test Incident Report;
- Test Summary Report;

10. Responsabilità

Ogni team member sarà responsabile del testing della gestione alla quale è stato assegnato. Il team non verrà diviso in "tester" e "developer" per evitare overhead di comunicazione.

11. Glossario

- **Rischio:** risultato potenziale, imprevedibile e incontrollabile di un'azione (o inazione);
- **Failure:** Qualsiasi deviazione del comportamento osservato dal comportamento specificato;
- **Fault:** La causa meccanica o algoritmica di un errore.
- **Developer:** è una figura professionale che si occupa dello sviluppo di applicazioni, in particolare dello sviluppo del codice sorgente di programmazione, dell'ottimizzazione dei database e dei linguaggi di markup attraverso tecniche di programmazione;
- **JUnit:** Framework di Unit Testing per il linguaggio di programmazione Java;
- **Katalon:** Strumento per l'automatizzazione del testing;
- **Spring:** Framework open source per lo sviluppo di applicazioni su piattaforma Java;
- **Tester:** una persona, una macchina o dispositivo utilizzato per verificare se un sistema o componente funziona correttamente;
- **Testing:** processo o metodo per trovare gli errori in un'applicazione o un programma software in modo che l'applicazione funzioni in base ai requisiti dell'utente finale;
- **Timeline:** Rappresentazione grafica della sequenza cronologica degli eventi più significativi (che include anche i documenti più importanti) che si verificano prima di partire, durante e dopo l'Erasmus;
- **Tool:** Strumento software utilizzato per ottenere un dato risultato.