1. **Introduzione**

In questo documento viene specificata la pianificazione della attività di testing della web App WhereDoIEat al fine di verificare se esistono differenze fra il comportamento atteso e il comportamento reale del sistema. In questa attività andremo a rilevare gli eventuali errori prodotti dal codice scritto, per evitare che essi si presentino nel momento in cui il sistema viene messo in esercizio. Le attività di testing sono state pianificate per le seguenti gestioni:

* Gestione Account
* Gestione Autenticazione
* Gestione Prenotazioni
* Gestione Recensioni
* Gestione Visualizzazioni
* Gestione Lista Preferiti
* Gestione Attività
* Gestione Segnalazioni
* Gestione dell'amministratore

1. **Documenti correlati**

Il test plan è strettamente collegato ai documenti prodotti fino ad ora, poiché prima di passare alla fase di testing abbiamo bisogno di avere una gran parte delle funzioni previste già implementate, queste sono state definite nei precedenti documenti. Di seguito verranno descritte le relazioni che ci sono fra il test plan e gli altri documenti

* 1. **Relazioni con il Requirement Analysis Document (RAD)**

La relazioni tra test plan RAD riguarda in particolare i requisiti funzionali e non funzionali del sistema poiché i test che saranno eseguiti su ogni funzionalità

Terranno conto delle specifiche espresse nel RAD

* 1. **Relazioni con il System Design Document (SDD)**

Nel System Design Document abbiamo suddiviso il nostro sistema in sottosistemi e l’architettura in tre livelli: Presentation Layer, Application Layer e storage Layer. Il test dei vari componenti deve rimanere fedele a queste suddivisioni il più possibile.

* 1. **Relazioni con l’Object Design Document (ODD)**

Il test d’integrazione farà quanto più riferimento possibile alle interfacce delle classi definite nell’ODD.

1. **Panoramica**

Come descritto nel System Design Document (SDD) il nostro sistema è costruito da una architettura “Three Layers” ovvero a tre livelli, al fine di garantire basso accoppiamento e alta coesione fra le classi. I tre livelli della architettura sono **Presentation, Application e Storage Layer**. I layer contengono rispettivamente la parte la parte visuale della applicazione, la parte applicativa e il database. Il sistema WhereDoIEat è stato diviso ulteriormente in sottosistemi più piccoli, in particolare è stato diviso per gestioni, le gestioni sono state definite nel capitoli 1. Le gestioni che abbiamo individuato prevedono principalmente metodi che effettuano operazioni CRUD(Create, Read, Update, Delete) e saranno queste le funzionalità che andranno testate.

1. **Funzionalità da testare/non testare**

Di seguito vengono elencate le funzionalità da testare per ogni gestione.

* Gestione Autenticazione
* Registrazione
* Autenticazione
* Gestione Account
* Modifica dati personali
* Gestione Prenotazione
* Prenotazione di un locale
* Gestione Recensione
* Valutazione e Recensione di un locale
* Gestione attività
* Aggiunta attività al sistema
* Gestione Segnalazioni
* Effettua suggerimento

Di seguito vengono elencate le funzionalità da non testare per ogni gestione

* Gestione Visualizzazione
* Visualizzazione tramite barra di ricerca
* Visualizzazione tramite home
* Visualizzazione tramite categorie
* Visualizzazione tramite comune
* Visualizzazione recensioni delle attività
* Visualizzazione mappa del locale
* Gestione Lista Preferiti
* Aggiungere un’attività alle sue preferite
* Visualizzare le sue attività preferite
* Eliminazione di un’attività dai preferiti
* Gestione Amministrazione
* Visualizzazione dati di ogni utente registrato
* Visualizzazione utenti registrati
* Eliminazione account
* Eliminazione commenti non appropriati
* Modificare password utenti

1. **Criteri Pass/Field**

I dati di input del test saranno suddivisi in classi di equivalenza, ovvero saranno raggruppati in insiemi con caratteristiche comuni, in modo tale da poter testare solo uno degli elementi della classe. L’input avrà superato il test se l’output risultante è quello atteso, il risultato atteso sarà quello specificato dal membro del team che si occuperà del testing su tale test case.

1. **Approccio**

La fase di testing sarà suddivisa in tre fasi:

**1. Testing di unità:**

verrà testata nello specifico il funzionamento di ogni singola unità del sistema;

**2. Testing di integrazione:**

dove verranno testate le interfacce delle suddette unità;

**3. Testing di sistema:**

dove verranno testato l’intero sistema assemblato

* 1. **Testing di Unità**

Durante questa fase verranno ricercate le condizioni di fallimento, isolando i componenti. La strategia usata per il testing è la tecnica Black-Box, che si concentra sul comportamento Input/Output ignorando la struttura interna della componente. Per minimizzare il numero di test cases i possibili input verranno partizionati in classi di equivalenza e per ogni classe verrà usato un test case. Gli errori scovati in questa fase, e anche in tutte le altre, devono essere comunicati agli sviluppatori in modo tale che questi possano correggere l’errore e ripristinare la fase di testing al più presto. I fallimenti identificati durante la fase di testing verranno specificati nel test incidenti report.

* 1. **Testing di Integrazione**

Questa fase ha lo scopo di integrare tutte le componenti di una funzionalità al fine di testarle nel complesso utilizzando una **strategia Big Bang**. L’approccio Big Bang mira a testare prima le componenti individualmente e poi tutte insieme, come un unico sistema.

* 1. **Testing di Sistema**

In questa ultima fase vogliamo dimostrare che il sistema soddisfi tutti i requisiti richiesti. Si cercherà di testare le funzionalità più utili per l’utente e quelle che hanno una maggiore probabilità di fallimento.

1. **Sospensione e ripresa**
   1. **Criteri di sospensione**

La fase di testing verrà sospesa una volta raggiunto un giusto compromesso fra qualità del prodotto e costi dell’attività di testing. Il testing verrà portato avanti per quanto più tempo possibile senza rischiare di ritardare ulteriormente la consegna finale del progetto.

* 1. **Criteri di ripresa**

In seguito alle modifiche o alle correzioni delle componenti che introdurranno errori o fallimenti, i test case verranno di nuovo sottoposti al sistema in modo da assicurarsi che la modifica o la correzione sia servita effettivamente per risolvere l’errore o il fallimento.

1. **Materiale per il testing**

L’hardware necessario alla attività di testing è un PC con connessione ad internet dato che il database che viene usato dal sistema è accessibile da remoto in modo tale che tutti possano avere una unica versione del db su cui lavorare.