# 1 Introduzione

Lo scopo del Test Plan è quello di gestire lo sviluppo e le attività di test riguardanti l’app YouthClub. Verranno pertanto individuati: vari elementi e funzionalità da testare, le strategie di testing e gli strumenti che saranno utilizzati, gli autori dei test, le risorse e le attività richieste per completare i test e i rischi associati al piano. Lo scopo del testing è quello di rilevare errori in maniera pianificata all'interno del codice realizzato al fine di evitare che tali errori si presentino durante l’utilizzo del sistema da parte dell’utente finale. I risultati uscenti dai test saranno utilizzati per capire dove intervenire per correggere gli errori o apportare modifiche con l’intenzione di migliorare il sistema.

In tale documento, verranno prese in considerazione ed analizzate le seguenti attività:

* Gestione Ricerca;
* Gestione Api;
* Gestione Recensione;
* Gestione Autenticazione;

Per ogni gestione saranno testate solo le funzionalità contenente i requisiti con priorità **Alta**.

# 2 Documenti correlati

Il Test Plan che si sta sviluppando si trova in relazione con i documenti che in precedenza sono stati realizzati ovvero: RAD, SDD, ODD. Per verificare il corretto funzionamento dell’app YouthClub verranno utilizzati i test case individuati e documentati precedentemente nel processo di sviluppo del sistema. I test case si rifanno alle funzionalità individuate nel documento di raccolta ed analisi dei requisiti (RAD).

## **2.1 Relazioni con il documento di analisi dei requisiti (RAD)**

La relazione tra Test Plan e RAD (Requirement Analysis Document) riguarda in particolare i requisiti funzionali e non funzionali del sistema in quanto i test verranno realizzati su quelle funzionalità tenendo conto delle specifiche espresse nel documento precedente. Il RAD definisce le informazioni relative allo scopo, l’ambito e gli obiettivi del sistema, mettendo in evidenza una panoramica di requisiti funzionali, requisiti non funzionali, scenari, casi d’uso, diagrammi e mockup del sistema.

Il documento a cui si fa riferimento è:

* YouthClub\_RAD\_V\_2.0: Requirement Analysis Document.

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

L’SDD (System Design Document) rappresenta l’architettura del sistema MVC. In particolare, l’SDD contiene l’architettura del software corrente e proposto e i servizi dei sottosistemi.

Il documento a cui si fa riferimento è:

* YouthClub\_SDD\_V\_3.0: System Design Document.

## **2.3 Relazioni con l’Object Design Document (ODD)**

In questa sezione, verrà indicato il riferimento all’ODD (Object Design Document), documento in cui vengono evidenziate le specifiche legate alle interfacce per ogni sottosistema, definite le librerie per le classi e selezionati i design pattern per risolvere problemi comuni e proteggere le classi da interventi futuri.

Il documento a cui si fa riferimento è:

* ChooseIT\_ODD\_V\_2.0: Object Design Document.

# 3 Panoramica del Sistema

Come descritto e analizzato nell’SDD l’architettura scelta per il nostro sistema è ibrida in quando sfrutta un modello client-server per quando riguarda le richieste inviate ed elaborate dal server, ma il server viene suddiviso in 3 layer Model-View-Control.

Il lato client si occupa di eseguire le richieste e decodificare e mostrare opportunamente le risposte nell’applicazione.

Il lato server presenta invece la logica e la conoscenza del dominio applicativo, il concetto di view diventa più generico e si occupa solo della formattazione dei dati in risposta alle richieste del client.

Per seguire tale architettura abbiamo deciso di dividere il nostro sistema in sottosistemi più piccoli basandoci sui diversi tipi di gestione:

* Gestione Recensione;
* Gestione Api;
* Gestione Autenticazione;
* Gestione Ricerca;

# 4 Funzionalità da testare e da non testare

Di seguito verranno elencate le funzionalità da testare suddivise per ogni gestione del sistema:

* Gestione Recensione
* Inserimento
* Modifica
* Visualizzazione
* Gestione Api
* Google Api
* FourSquare Api
* Yelp Api
* Geocoding
* Gestione Autenticazione
* Salvataggio codice Imei
* Gestione Ricerca
* Ricerca per nome
* Ricerca per località

# 5 Criteri pass/failed

I dati di input del test saranno raggruppati in insiemi dalle caratteristiche comuni in modo da effettuare un test su di un unico elemento rappresentativo. Il testing ha successo se l’output osservato è diverso dall’output atteso: ciò significa che la fase di testing avrà successo se individuerà una failure. 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 altre componenti del sistema. La failure quindi è uno stato di condizione nel quale non si trova l’output desiderato, si può dire che è il contrario di un successo.

# 6 Approccio

L’approccio alla fase di testing si divide in 2 fasi:

* La prima servirà a testare le componenti una ad una.
* La seconda prevede di testare le funzionalità delle integrazioni dei vari sottosistemi.

## **6.1 Testing di unità**

Per realizzare il testing di ogni singola componente verrà utilizzata la tecnica “Black-Box Testing”. Utilizzando tale tecnica si andrà ad analizzare ed esaminare le funzionalità dell’applicazione e il comportamento input/output delle singole componenti senza dare importanza alla loro struttura interna. Essendo quasi impossibile generare tutti i possibili input, verranno create classi d’equivalenza scegliendo per ognuna un test case per ridurre la ridondanza e rendere il test più efficiente. I risultati del testing verranno analizzati e usati per correggere gli errori che causano il fallimento del sistema.

## **6.2 Testing di integrazione**

Dopo aver sottoposto ogni componente al testing di unità, ed aver corretto gli eventuali errori riscontrati dal test, essi verranno integrati in sottosistemi più grandi per sottoporli ad un test di integrazione. Per il testing di integrazione verrà utilizzato il Big Bang Testing (BBT): grazie ad esso, le componenti verranno testate prima singolarmente e poi messe insieme in un unico sistema, inoltre non sarà necessario sviluppare stub o driver per rendere funzionale un sottoinsieme. E’ però difficile, in sistemi complessi, individuare la componente responsabile di un errore.

# 7 Materiale per il testing

Lo strumento necessario per l’attività di testing è un computer contente tutti i file del progetto

# 8 Test Case