|  |
| --- |
| **Università degli Studi di Salerno Corso di Ingegneria del Software** |

**ClickFly  
Test Plan Document  
Versione 0.1**

****

Data: 16/12/2024

**Coordinatore del progetto:**

|  |  |
| --- | --- |
| Nome | Matricola |
|  |  |
|  |  |

**Partecipanti:**

|  |  |
| --- | --- |
| Nome | Matricola |
| Fabio Pennarella | 05121 17301 |
| Antonio Sirico | 05121 18702 |
| Giusy Chierchia | 05121 17508 |
| Cuono Crimaldi | 05121 16728 |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Scritto da:** | Fabio Pennarella, Giusy Chierchia |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. **Introduzione**

Questo documento ha lo scopo di definire gli obiettivi, le strategie e le attività necessarie per garantire che il sistema ClickFly funzioni correttamente, andando a verifica se questo soddisfa i requisiti funzionali e non funzionali stabiliti. L’obiettivo è assicurare che il sistema sia affidabile, sicuro, performante e facilmente utilizzabile, indicando le specifiche strategie di testing e i casi di test corrispondenti.

1. **Relazione con gli altri documenti**

In questo documento verranno citate informazioni contenute nel documento di analisi dei requisiti (RequirementAnalisysDocument(RAD)\_ClickFly.docx), nel documento di System Design (SystemDesignDocument(SDD)\_ClickFly.docx), e nel documento di Object Design (ObjectDesignDocument(ODD)\_ClickFly.docx).

1. **System Overview**

Il sistema ClickFly è composto da tre rincipali moduli che lavorano insieme:

* Gestione Utente
* Gestione Amministratore
* Gestione Voli

Questi moduli vengono testati separatamente durante la fase di test d’unita per garantire il corretto funzionamento di ciascun componente, mentre i test di integrazione e di sistema verificano la corretta interazione tra i moduli e l'esperienza complessiva dell'utente.  
Verranno testati solo i requisiti funzionali con alta priorità.

1. **Funzionalità da testare**

Di seguito le funzionalità che saranno oggetto di test:

* 1. Gestione Utente:
     1. Registrazione
     2. Autenticazione
     3. Aggiunta voli nel carrello
     4. Modifica quantità voli nel carrello
     5. Rimozione voli dal carrello
     6. Acquisto biglietti
     7. Visualizzazione delle informazioni personali
     8. Visualizzazioni dei biglietti acquistati
     9. Aggiunta saldo al portafoglio virtuale
  2. Gestione Amministratore:
     1. Aggiunta volo al catalogo
     2. Modifica volo nel catalogo
     3. Rimozione volo dal catalogo
     4. Visualizza utenti
     5. Visualizza i biglietti acquistati dagli utenti
  3. Gestione Voli:
     1. Filtraggio dei voli

1. **Criteri di successo e fallimento**

Il testing è un metodo che mira a generare condizioni di errore o di malfunzionamento.

Per verificare se un caso di test ha identificato un problema, si ricorre a un "oracolo", ovvero il risultato previsto dell'esecuzione, confrontandolo con l'output reale del test.

I criteri di pass/fail del testing si suddivideranno come segue:

* Corretto: Il valore ricevuto è uguale all’oracolo quindi nessun errore è stato rilevato
* Errato: Il valore ricevuto è differente dall’oracolo quindi è stato riscontrato un errore

1. L’approccio utilizzato per il testing è composto da tre fasi:

* Testing di Unità:

Il testing di unità sarà effettuato tramite l’utilizzo di un’analisi dinamica con approccio Black-Box. La tecnica Black-Box si concentra sulle funzionalità esterne. Per gestire l'impossibilità di prevedere tutti i casi di test, tramite il partizionamento delle classi di equivalenza, le condizioni di input saranno divise in classi di equivalenza e saranno selezionati test case rappresentativi per ciascuna classe.  
Tramite il Category Partition verranno individuati i casi di test dai requisiti funzionali e dai casi d'uso nel RAD.

* Testing di integrazione:

Il testing di integrazione individua bug che non sono stati scoperti durante il testing di unità. Il test verrà effettuato tramite la strategia di tipo big-bang.

* Testing di sistema:

Il testing di sistema verifica l'intero sistema integrato per assicurarsi che soddisfi i requisiti specificati. Utilizzeremo la tecnica delle Category Partition, che suddivide i requisiti in categorie e varianti significative, per creare casi di test che coprano diverse combinazioni di condizioni. Questo approccio garantisce una copertura completa ed efficace dei test.