TSR: Test Summary Report

Riferimento	
Versione	1.0
Data	22/01/2022
Destinatario	Prof. Carmine Gravino
Presentato da	Danilo Aliberti
Approvato da	

Revision history

Data	Versione	Descrizione	Autore
18/01/2022	0.1	Stesura iniziale	Danilo Aliberti
22/01/2022	1.0	Revisione e aggiornamento	Danilo Aliberti

Sommario

1. Introduzione

Dopo aver puntato l'attenzione alla specifica dei diversi test case, si passa alla rendicontazione dei risultati ottenuti dalle effettive attività di testing effettuate su di essi.
Come già specificato nel Test Plan, per la fase di testing sono stati utilizzati gli strumenti **unittest** e **Coverage.py**, successivamente sarà invece usato Selenium IDE per la fase di testing di sistema.
Una prima analisi sarà effettuata sui difetti riscontrati nell'applicazione, ovvero un insieme di valutazioni rispetto ai fallimenti o gli errori riscontrati nei test, che necessitano di ulteriori approfondimenti.
Un report significativo delle attività di testing svolte è necessario, per poter portare a conoscenza di chi usufruirà della piattaforma, di eventuali failure o bug presenti nel sistema e permettere agli sviluppatori di poter giungere a soluzioni a tali limiti riscontrati nei test.

1.1 Acronimi e abbreviazioni

- **TP:** Test Plan;
- **TCS:** Test Case Specification;
- **TIR:** Test Incident Report;
- **ODD:** Object Design Document.

2. Riferimenti

Di seguito vengono elencate le relazioni tra il presente documento e gli altri documenti di testing.

Relazione con il Test Plan

Il Test Summary Report fa riferimento alle attività di testing specificate nel Test Plan.

Relazione con il Test Case Specification

Il Test Summary Report contiene il sunto dell'esecuzione dei test di sistema specificati nel Test Case Specification.

Relazione con il Test Incident Report

Il Test Summary Report contiene il sunto dei risultati sull'esecuzione specificati nel Test Incident Report.

3. Testing di unità e di Integrazione

Nel corso dello sviluppo di tutto il progetto, i test unitari e di integrazione sono stati scritti in una unica classe. Tutti i test sono stati divisi in base al package di appartenenza: i test di Gestione Utenza saranno situati nel package *gestioneutenza* nel file *tests.py*, come indicato al punto 2 del documento ODD, e così via.
Durante la fase di sviluppo e alla fine è stato utilizzato il tool Coverage.py per la raccolta di metriche sulla coverage del testing. Di seguito sono riportati i risultati della coverage dei test.

# Branch	# Branch covered	Branch Coverage	Code coverage
238	195	82%	83%

Nella cartella del progetto è presente un file XML contenente i dettagli generati dal report della coverage, tra cui anche dettagli sulla branch coverage.
Una review della code coverage è presente nella cartella "htmlcoverage" del repository di Github.

3.1 Screenshots Coverage.py

Coverage report: 83%

Module ↑	statements	missing	excluded	branches	partial	coverage
core/decorators.py	95	30	0	20	1	63%
core/management/commands/wait_for_db.py	15	0	0	4	0	100%
core/views.py	60	5	0	22	0	94%
gestionecampionato/caricamentovoti/ImportVoti.py	45	0	0	4	0	100%
gestionecampionato/commandpattern/command.py	6	1	0	2	0	88%
gestionecampionato/commandpattern/generacalendariocommand.py	9	0	0	2	0	100%
gestionecampionato/commandpattern/invoker.py	8	0	0	4	1	92%
gestionecampionato/commandpattern/receiver.py	32	0	0	10	0	100%
gestionecampionato/exceptions.py	8	0	0	8	0	100%
gestionecampionato/facadepattern/facade.py	69	10	0	18	2	77%
gestionecampionato/forms.py	82	19	0	36	4	77%
gestionecampionato/views.py	179	55	0	38	2	71%
gestionesquadra/exceptions.py	2	0	0	2	0	100%
gestionesquadra/forms.py	29	0	0	8	0	100%
gestionesquadra/views.py	122	25	0	18	1	81%
gestioneutenza/forms.py	26	0	0	8	1	97%
gestioneutenza/strategyclasses/allenatorestrategy.py	39	1	0	10	1	96%
gestioneutenza/strategyclasses/castrategy.py	18	0	0	2	0	100%
gestioneutenza/strategyclasses/invitenotfound.py	2	0	0	2	0	100%
gestioneutenza/strategyclasses/strategy.py	18	3	0	4	0	86%
gestioneutenza/views.py	66	4	0	16	0	95%
Total	**930**	**153**	**0**	**238**	**13**	**83%**

coverage.py v6.2, created at 2022-01-30 19:43 +0000

Figura 1 Code coverage totale (include branches)

```
branches-valid="238" branches-covered="195" branch-rate="0.8193" complexity="0">
```

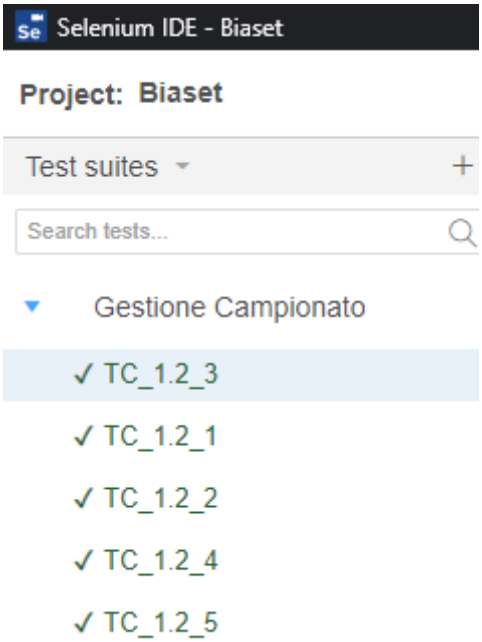
Figura 2 Estratto del file Coverage.XML. Su 238 branches ne sono stati testati 195, ovvero l'81,93%

4. Testing di sistema

Per il testing di sistema sono state definiti alcuni test suites, suddivise per package, tramite il tool Selenium IDE per Google Chrome. In totale, quindi, sono state definite 3 test suites, una per ogni package sviluppato. Al termine di ogni esecuzione è stato necessario fare un flush sul database, per cancellare i dati inseriti.
Di seguito sono riportati i risultati ottenuti

Data esecuzione	Numero fallimenti	Numero successi
19/01/2022	3	18
22/01/2022	0	21

4.1 Screenshots Selenium IDE Tests



- Gestione Squadra
  - ✓ TC_1.3_1
  - ✓ TC_1.3_2
  - ✓ TC_1.3_3
  - ✓ TC_1.4_1
  - ✓ TC_1.4_2
  - ✓ TC_1.4_3
  - ✓ TC_1.4_4
  - ✓ TC_1.5_1
  - ✓ TC_1.5_2
  - ✓ TC_1.5_3
  - ✓ TC_1.5_4
  - ✓ TC_1.5_5
- Gestione Utenza
  - ✓ TC_1.1_2
  - ✓ TC_1.1_3
  - ✓ TC_1.1_4
  - ✓ TC_1.1_5