Laurea Triennale in Informatica - Università di Salerno
Corso di *Ingegneria del Software* - Prof C. Gravino

Test Plan Document

Easy Pass

Riferimento	
Versione	1.1
Data	14/1/2022
Destinatario	Prof. C. Gravino
Presentato da	Montefusco Alberto Mulino Martina Rinaldi Viviana Spina Gennaro
Approvato da	

Sommario

Revision History

Data	Versione	Descrizione	Autori
5/12/2021	0.1	Prima stesura: aggiunta capitoli 1 – 2 – 3 – 4	Alberto Montefusco
6/12/2021	0.2	Aggiunta capitoli da 5 – 8	Alberto Montefusco Viviana Rinaldi
10/12/2021	0.3	Aggiunta capitoli 6 – 7 – 10	Martina Mulino
11/12/2021	0.4	Aggiunta Specifica dei Test Cases – capitolo 9	Alberto Montefusco Viviana Rinaldi Martina Mulino Gennaro Spina
13/12/2021	1.0	Revisione finale	Alberto Montefusco Viviana Rinaldi Martina Mulino Gennaro Spina
14/1/2022	1.1	Correzioni	Alberto Montefusco Viviana Rinaldi Martina Mulino Gennaro Spina

1. Introduzione

1.1 Descrizione del Documento

Il Test Plan del progetto Easy Pass presenta la pianificazione e la specifica di quelli che sono i Test Case, ovvero l'insieme di input e di risultati attesi che servono a testare una componente del Sistema. Nel caso in cui delle attività di testing evidenziassero degli errori che possano causare comportamenti diversi da quelli attesi e che possano compromettere il buon utilizzo del Sistema da parte degli utenti, quest'ultimo può essere sottoposto ad un processo di correzione degli errori individuati, per garantire agli utenti finali un prodotto software che rispecchi tutte le specifiche finora stabilite nelle precedenti fasi di sviluppo.

2. Relazioni con altri Documenti

Il Test Plan presenta diversi punti di correlazione con i documenti stilati durante le fasi precedenti dello sviluppo di Easy Pass. In particolare, il presente Documento fa riferimento ai casi d'uso descritti nel Requirement Analysis Document (RAD). Ulteriori dettagli sono specificati nel Test Case Specification Document (TCSD).

3. Panoramica del Sistema

Il Sistema Easy Pass fornisce tutte le sue funzionalità attraverso una Web Application. Per assicurarsi che ciascuna si comporti come previsto, bisogna quindi assicurarsi che vengano testate tutte le funzionalità offerte dal Sistema. Nel caso in cui queste funzionalità si comportino nella maniera prevista, si può ritenere che il Sistema Easy Pass soddisfi gli obiettivi che si erano prefissati.

4. Funzionalità da testare e no

Le attività di testing previste per il Sistema Easy Pass prevedono di verificare il corretto funzionamento della maggior parte delle funzionalità del Sistema. Quelle che saranno oggetto dell'attività di testing sono:

1. Registrazione di un Docente;

2. Selezione di un numero di studenti da validare;
3. Invio di un Green Pass;
4. Ricerca dei report tramite filtri;
5. Selezione di un formato dei report.

Per raggiungere la Branch Coverage del 75% dei casi di test, nel rispetto dei criteri di accettazione, si è deciso di sottoporre al testing anche tutte le classi DAO e le classi Bean di Direttore e Dipartimento. Sono state escluse dall'attività di testing le classi dei restanti Bean, poiché contengono solo getter e setter, le classi Mapper, che contengono un solo metodo, i metodi non utilizzati ma forniti comunque per completezza, le classi del package Utils eccetto i Validator e le componenti off-the-shelf.

4.1 Matrice di tracciabilità del Testing

Requisiti	Casi di Test				
	TC_1	TC_2	TC_3	TC_4	TC_5
RF[1]	✓				
RF[2]					
RF[3]		✓			
RF[4]			✓		
RF[5]					
RF[6]					
RF[7]					
RF[8]					
RF[9]					
RF[10]					
RF[11]					
RF[12]				✓	
RF[13]					✓

5. Criteri di successo/fallimento

Un caso di test ha esito positivo se l'output osservato è differente dal risultato previsto dall'oracolo; al contrario, un caso di test ha esito negativo se l'output osservato coincide con il risultato previsto dall'oracolo. Pertanto, le attività di test hanno successo nei casi in cui riescono ad individuare dei comportamenti anomali nell'esecuzione delle funzionalità del Sistema. Nel caso in cui uno o più casi di testi riscuotono successo, è possibile attuare un'opportuna procedura di correzione del difetto riscontrato e, successivamente, ricorrere ad un test di regressione per testare nuovamente la funzionalità modificata ed accertarsi che il problema sia stato risolto.

6. Approccio

L'attività di Testing del Sistema è stata organizzata in tre categorie principali:

- Testing di Unità, che si occupa di testare le singole componenti del Sistema;
- Testing di Integrazione, che si occupa di integrare le unità testate in modo isolato, così da controllare che il sistema funzioni correttamente quando più unità collaborano;
- Testing di Sistema, che include l'attività di Testing Funzionale, la quale si occupa di testare le funzionalità del Sistema definite dai requisiti funzionali. Il Testing di Sistema include anche il Testing di Performance e il Testing Pilota, che però non verranno effettuate poiché il budget a disposizione per il progetto non copre l'ammontare di risorse necessarie.

6.1 Testing di Unità

L'attività di Testing di Unità consiste nell'isolare un'unità del Sistema e provare che funzioni correttamente. La strategia prevista per affrontare tale attività consiste nel testare i metodi delle classi del Sistema e, in particolare, quelli che permettono le validazioni dell'input dell'utente, come descritto nei test cases. I casi di test saranno definiti attraverso un approccio Black-box attraverso l'uso del framework JUnit. Inoltre, per isolare le componenti testate è stato utilizzato Mockito, mentre per il calcolo della Branch Coverage del testing è stato sfruttato JaCoCo.

6.2 Testing di Integrazione

L'attività di Testing di Integrazione consiste nell'integrare le componenti testate singolarmente in sottosistemi più grandi, così da rilevare i bug che non sono stati evidenziati con il Testing di Unità. Per questa attività è stata scelta la strategia di testing Bottom-Up, in quanto è ritenuta migliore per i sistemi software Object-Oriented, paradigma utilizzato per Easy Pass. In particolare, sono stati testati tutti i metodi delle classi Bean che richiamano i metodi delle classi DAO e i DAO stessi. Le tecnologie utilizzate sono le stesse di quelle del Testing di Unità, dunque JUnit, JaCoCo e Mockito.

6.3 Testing di Sistema

L'attività di Testing Funzionale, inclusa in quella di Testing di Sistema, consiste nell'individuare i possibili fault generati dagli input degli utenti. L'approccio utilizzato per testare le funzionalità del Sistema, definite a partire dai requisiti funzionali e dai casi d'uso esposti nel documento "RAD", è quello del "Black-box Testing". La tecnica utilizzata per individuare i casi di test e per descriverli è la "Category Partition", in cui sfruttiamo le classi di equivalenza per partizionare l'insieme dei possibili input dell'utente.
L'attività di Testing di Sistema è invece effettuata tramite Selenium IDE e consiste nell'utilizzare tale tool per verificare che i test case descritti nel Testing Funzionale non rilevino errori.

7. Strumenti per il testing (Hardware/Software)

Lo strumento hardware utilizzato per l'attività di testing è il computer e, poiché il Sistema non è stato ancora rilasciato, non necessariamente deve essere connesso ad Internet.
Come anticipato nei precedenti paragrafi, gli strumenti software utilizzati per le attività di testing sono:

- JUnit: framework open source usato per scrivere ed eseguire Testing di Unità per Java;
- JaCoCo: libreria per il Code Coverage in ambito Java;
- Mockito: framework utilizzato per la realizzazione di oggetti mock durante il Testing di Unità;
- Selenium IDE: tool open source utilizzato per la gestione automatizzata dei browser, utilizzato come framework di testing. Esso permette di registrare le azioni che un utente può effettuare nell'interazione con il Sistema, così da poter eseguire i casi di test.

8. Test Case

Per sviluppare i Test Case sarà utilizzato il metodo del Category Partition. Questo metodo consiste nell'identificare per ogni funzionalità da testare dei parametri; per ogni parametro verranno individuate delle categorie, le quali poi saranno suddivise in scelte. Alle scelte verrà assegnato un valore. I Test Case verranno definiti nel documento di Test Cases Specification (TCS).

9. Specifica dei Test Case

9.1 Registrazione di un Docente

TC_1 Registrazione Docente

Nota: sebbene nel TCSD si legga che i messaggi di errore sono mostrati uno per volta, ciò non accade nell'effettivo utilizzo del Sistema perchè, per migliorare la User Experience, sono stati utilizzati dei metodi nel front-end che effettuano il controllo di tutti i campi in contemporanea. Se tali controlli vengono disabilitati, il back-end si occuperà di mostrare i messaggi di errore uno per volta come espresso nella documentazione.

Parametro: Nome **Formato**: ^[a-zA-Z .']+$	
Categorie	**Possibili Scenari**
Campo vuoto cvn	**1.** Il campo non è stato compilato [**errore**] **2.** Il campo è stato compilato [property **campoVuotoCVNok**]
Lunghezza ln	**1.** Lunghezza <= 0 [if **campoVuotoCVNok**] [**errore**] **2.** Lunghezza > 30 [if **campoVuotoCVNok**] [**errore**] **3.** 0 < Lunghezza <= 30 [if **campoVuotoCVNok**] [property **lunghezzaLNok**]
Formato fn	**1.** Non rispetta il formato [if **campoVuotoCVNok** and **lunghezzaLNok**] [**errore**] **2.** Rispetta il formato [if **campoVuotoCVNok** and **lunghezzaLNok**] [property **formatoFNok**]

Parametro: Cognome **Formato:** ^[a-zA-Z .']+$	
Categorie	**Scelte**
Campo vuoto cvc	**1.** Il campo non è stato compilato [**errore**] **2.** Il campo è stato compilato [property **campoVuotoCVCok**]
Lunghezza lc	**1.** Lunghezza <= 0 [if **campoVuotoCVCok**] [**errore**] **2.** Lunghezza > 30 [if **campoVuotoCVCok**] [**errore**] **3.** 0 < Lunghezza <= 30 [if **campoVuotoCVCok**] [property **lunghezzaLCok**]
Formato fc	**1.** Non rispetta il formato [if **campoVuotoCVCok** and **lunghezzaLCok**] [**errore**] **2.** Rispetta il formato [if **campoVuotoCVCok** and **lunghezzaLCok**] [property **formatoFCok**]

Parametro: Dipartimento (Dropdown menù) **Formato:** -	
Categorie	**Scelte**
Scelta sd	**1.** Scelta non effettuata [**errore**] **2.** Scelta effettuata [property **sceltaSDok**]

Parametro: E-mail **Formato:** ^[a-zA-Z0-9_.]{3,}@(?:(?:[a-zA-Z0-9-]+\.)?[a-zA-Z]+\.)?(unisa)\.it$	
Categorie	**Scelte**
Campo vuoto cve	1. Il campo non è stato compilato [**errore**] 2. Il campo è stato compilato [property **campoVuotoCVEok**]
Formato fe	1. Non rispetta il formato [if **campoVuotoCVEok**] [**errore**] 2. Rispetta il formato [if **campoVuotoCVEok**] [property **formatoFEok**]
Esiste ee	1. Esiste nel database [if **campoVuotoCVEok** and **formatoFEok**] [**errore**] 2. Non esiste nel database [if **campoVuotoCVEok** and **formatoFEok**] [property **esisteEEok**]

Parametro: Password **Formato:** ^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[=^ ì{}+çò°àù§èé#@$!%€*?&:,;'._<>|-])[A-Za-z\d=^ ì{}+çò°àù§èé#@$!%€*?&:,;'._<>|-]{8,}$	
Categorie	**Scelte**
Campo vuoto cvp	1. Il campo non è stato compilato [**errore**] 2. Il campo è stato compilato [property **campoVuotoCVPok**]
Lunghezza lp	1. Lunghezza < 8 [if **campoVuotoCVPok**] [**errore**] 2. Lunghezza >= 8 [if **campoVuotoCVPok**] [property **lunghezzaLPok**]
Formato fp	1. Non rispetta il formato [if **campoVuotoCVPok** and **lunghezzaLPok**] [**errore**] 2. Rispetta il formato [if **campoVuotoCVPok** and **lunghezzaLPok**] [property **formatoFPok**]

Codice	Combinazione	Esito
TC_1_01	cvn1	FAIL
TC_1_02	cvn2.ln1	FAIL
TC_1_03	cvn2.ln2	FAIL
TC_1_04	cvn2.ln3.fn1	FAIL
TC_1_05	cvn2.ln3.fn2.cvc1	FAIL
TC_1_06	cvn2.ln3.fn2.cvc2.lc1	FAIL
TC_1_07	cvn2.ln3.fn2.cvc2.lc2	FAIL
TC_1_08	cvn2.ln3.fn2.cvc2.lc3.fc1	FAIL
TC_1_09	cvn2.ln3.fn2.cvc2.lc3.fc2.sd1	FAIL
TC_1_10	cvn2.ln3.fn2.cvc2.lc3.fc2.sd2.cve1	FAIL
TC_1_11	cvn2.ln3.fn2.cvc2.lc3.fc2.sd2.cve2.fe1	FAIL
TC_1_12	cvn2.ln3.fn2.cvc2.lc3.fc2.sd2.cve2.fe2.ee1	FAIL
TC_1_13	cvn2.ln3.fn2.cvc2.lc3.fc2.sd2.cve2.fe2.ee2.cvp1	FAIL
TC_1_14	cvn2.ln3.fn2.cvc2.lc3.fc2.sd2.cve2.fe2.ee2.cvp2.lp1	FAIL
TC_1_15	cvn2.ln3.fn2.cvc2.lc3.fc2.sd2.cve2.fe2.ee2.cvp2.lp2.fp1	FAIL
TC_1_16	cvn2.ln3.fn2.cvc2.lc3.fc2.sd2.cve2.fe2.ee2.cvp2.lp2.fp2	PASS

9.2 Selezione di un numero di studenti da validare

TC_2 Inserimento numero studenti

Parametro: Numero Studenti **Formato:** ^[1-9]+[0-9]*$	
Categorie	**Scelte**
Campo vuoto cvns	**1.** Il campo non è stato compilato [**errore**] **2.** Il campo è stato [property **campoVuotoCVNSok**]
Formato fns	**1.** Non rispetta il formato [if **campoVuotoCVNSok**] [**errore**] **2.** Rispetta il formato [if **campoVuotoCVNSok**] [property **formatoFNSok**]
Valore vns	**1.** Valore <= 0 [if **campoVuotoCVNSok** and **formatoFNSok**] [**errore**] **2.** Valore > 0 [if **campoVuotoCVNSok** and **formatoFNSok**] [property **valoreVNSok**]

Codice	Combinazione	Esito
TC_2_01	cvns1	FAIL
TC_2_02	cvns2.fns1	FAIL
TC_2_03	cvns2.fns2.vns1	FAIL
TC_2_04	cvns2.fns2.vns2	PASS

9.3 Invio di un Green Pass

Nota. Nel testing con Selenium è necessario inserire un path valido per caricare le immagini che vengono testate poichè non esiste un path univoco per tutte le macchine.

TC_3 Invio del Green Pass

Questo test case verifica se la stringa decodificata dal QR Code dell'immagine inserita dallo Studente rappresenti un Green Pass (un Green Pass è definito tale se la stringa che lo rappresenta è in base 45).

Parametro: Green Pass **Formato:** "HC1:*"	
Categorie	**Scelte**
Campo vuoto cvgp	1. campo vuoto [**errore**] 2. campo non vuoto [property **campoVuotoCVGPok**]
Formato fgp	1. input non corrispondente al formato [if **campoVuotoCVGPok**] [**errore**] 2. input corrispondente al formato [if **campoVuotoCVGPok**] [property **formatoFGPok**]

Codice	combinazione	Esito
TC_3_01	cvgp1	FAIL
TC_3_02	cvp2.fgp1	FAIL
TC_3_03	cvp2.fgp2	PASS

9.4 Ricerca dei report

TC_4 Ricerca dei Report

Poiché il caso d'uso UC_RicercaReport (vedi documento RAD) indica che la ricerca dei Report può essere effettuata specificando o solo il Docente che ha generato i Report, o il range di date entro cui vengono generati i Report, o entrambi, il test case TC_4 Ricerca dei Report è l'aggregazione di tre test case componenti:

1) TC_4.1 RicercaSoloDocente;
2) TC_4.2 RicercaSoloDate;
3) TC_4.3 RicercaCompleta.

Il motivo di tale suddivisione dei test case è che, nel caso in cui venga specificato solo il nome e cognome del Docente e non le date, non si tratterà di un errore, in quanto è una funzionalità prevista dall'use case UC_RicercaReport. Stessa cosa nel caso in cui vengano specificate le date ma non il Docente.
È bene specificare pertanto che alcuni test case di TC_4.2 RicercaSoloDate e di TC_4.3 RicercaCompleta non verranno testati con Selenium IDE per le seguenti ragioni:

- **TC_4.2 RicercaSoloDate**:
  - **TC_4.2_01 RicercaSoloDate**: in questo specifico test case l'utente vuole cercare i Report inserendo solo un range di date senza specificare il nome ed il cognome del Docente. In particolare, il test case tratta l'errore nel caso in cui la prima data non ha un valore: il messaggio del seguente oracolo non verrà mai mostrato all'utente, bensì gli sarà notificato "Inserire un Docente." poiché i controlli sono effettuati in modo sequenziale e, di conseguenza, verrà mostrato all'utente l'oracolo del TC_4.1_01;

  - **TC_4.2_02** e **TC_4.2_05**: i seguenti oracoli non verranno mostrati all'utente in quanto, il tag HTML utilizzato per la specifica delle date non permette di inserire date diverse dal formato citato nella documentazione (dd/MM/yyyy), bensì restituirà una stringa vuota ed il Sistema notificherà ciò con il messaggio "Inserire la prima data." per il TC_4.2_02 e "Inserire la seconda

data." per il TC_4.2_05. Nel caso in cui sia disabilitato il vincolo imposto dal tag HTML, la suddetta situazione non si presenta.

- **TC_4.3 RicercaCompleta**:
  - **TC_4.3_04**: il Sistema non genererà un errore in quanto, come già detto in precedenza, effettuerà i controlli in maniera sequenziale e, poiché il nome e il cognome del Docente inserito rispettano il formato ed esistono nel database, i report verranno ricercati mediante l'anagrafica del Docente;
  - **TC_4.3_05** e **TC_4.3_08**: spiegazione analoga a TC_4.2_02 e TC_4.2_05.

TC_4.1 RicercaSoloDocente

Parametro: Nome completo del Docente **Formato:** ^[a-zA-Z .']+$	
Categorie	**Scelte**
Campo vuoto cvndoc	1. Il campo è vuoto [**errore**] 2. Il campo non è vuoto [property **campoVuotoCVNDOCok**]
Formato fndoc	1. input non corrispondente al formato [if **esisteENDOCok**] [**errore**] 2. input corrispondente al formato [if **esisteENDOCok**] [property **formatoFNDOCok**]
Esiste endoc	1. non esiste nel DB [if **campoVuotoCVNDOCok**] [**errore**] 2. esiste nel DB [if **campoVuotoCVNDOCok**] [property **esisteENDOCok**]

Codice	Combinazione	Esito
TC_4.1_01	cvndoc1	FAIL
TC_4.1_02	cvndoc2.fndoc1	FAIL
TC_4.1_03	cvndoc2.fndoc2.endoc1	FAIL
TC_4.1_04	cvndoc2.fndoc2.endoc2	PASS

TC_4.2 RicercaSoloDate

Parametro: PrimaData, SecondaData **Formato:** ^\d{4}\-(0?[1-9]|1[012])\-(0?[1-9]|[12][0-9]|3[01])$	
Categorie	**Scelte**
Campo vuoto cvfdat	1. campo prima data vuoto [**errore**] 2. campo prima data non vuoto [property **campoVuotoCVFDATok**]
Formato ffdat	1. input non corrispondente al formato [if **campoVuotoCVFDATok**] [**errore**] 2. input corrispondente al formato [property **formatoFFDATok**]
Campo vuoto cvsdat	1. campo vuoto [if **formatoFFDATok**] [**errore**] 2. campo non vuoto [if not **campoVuotoCVFDATok**] [**errore**] 3. campo non vuoto [if **formatoFFDATok**] [property **campoVuotoCVSDATok**]
Formato fsdat	1. input non corrispondente al formato [if **campoVuotoCVSDATok**] [**errore**] 2. input corrispondente al formato [if **campoVuotoCVSDATok**] [property **formatoFSDATok**]
Confronto cdat	1. ffdat > fsdat [**formatoFFDATok** and **formatoFSDATok**] [**errore**] 2. ffdat <= fsdat [if **formatoFFDATok** and **formatoFSDATok** and **campoVuotoCVSDATok**] [property **confrontoCVDATok**]

Codice	Combinazione	Esito
TC_4.2_01	cvfdat1	FAIL
TC_4.2_02	cvfdat2.ffdat1	FAIL
TC_4.2_03	cvfdat2.ffdat2.cvsdat1	FAIL
TC_4.2_04	cvfdat2.ffdat2.cvsdat2	FAIL
TC_4.2_05	cvfdat2.ffdat2.cvsdat3.fsdat1	FAIL
TC_4.2_06	cvfdat2.ffdat2.cvsdat3.fsdat2.cdat1	FAIL
TC_4.2_07	cvfdat2.ffdat2.cvsdat3.fsdat2.cdat2	PASS

TC_4.3 RicercaCompleta

Parametro: Nome completo del Docente **Formato:** ^[a-zA-Z .']+$	
Categorie	**Scelte**
Campo vuoto cvndoc	**1.** Il campo è vuoto [**errore**] **2.** Il campo non è vuoto [property **campoVuotoCVNDOCok**]
Esiste endoc	**1.** non esiste nel DB [if **campoVuotoCVNDOCok**] [**errore**] **2.** esiste nel DB [if **campoVuotoCVNDOCok**] [property **esisteENDOCok**]
Formato fndoc	**1.** input non corrispondente al formato [if **esisteENDOCok**] [**errore**] **2.** input corrispondente al formato [if **esisteENDOCok**] [property **formatoFNDOCok**]

Parametro: PrimaData, SecondaData **Formato:** ^\d{4}\-(0?[1-9]|1[012])\-(0?[1-9]|[12][0-9]|3[01])$	
Categorie	**Scelte**
Campo vuoto cvfdat	1. campo prima data vuoto [**errore**] 2. campo prima data non vuoto [property **campoVuotoCVFDATok**]
Formato ffdat	1. input non corrispondente al formato [if **campoVuotoCVFDATok**] [**errore**] 2. input corrispondente al formato [property **formatoFFDATok**]
Campo vuoto cvsdat	1. campo vuoto [if **formatoFFDATok**] [**errore**] 2. campo non vuoto [if not **campoVuotoCVFDATok**] [**errore**] 3. campo non vuoto [if **formatoFFDATok**] [property **campoVuotoCVSDATok**]
Formato fsdat	1. input non corrispondente al formato [if **campoVuotoCVSDATok**] [**errore**] 2. input corrispondente al formato [if **campoVuotoCVSDATok**] [property **formatoFSDATok**]
Confronto cdat	1. ffdat > fsdat [**formatoFFDATok** and **formatoFSDATok**] [**errore**] 2. ffdat <= fsdat [if **formatoFFDATok** and **formatoFSDATok** and **campoVuotoCVSDATok**] [property **confrontoCVDATok**]

Codice	Combinazione	Esito
TC_4.3_01	cvndoc1	FAIL
TC_4.3_02	cvndoc2.endoc1	FAIL
TC_4.3_03	cvndoc2.endoc2.fndoc1	FAIL
TC_4.3_04	cvndoc2.endoc2.fndoc2.cvfdat1	FAIL
TC_4.3_05	cvndoc2.endoc2.fndoc2.cvfdat2.ffdat1	FAIL
TC_4.3_06	cvndoc2.endoc2.fndoc2.cvfdat2.ffdat2.cvsdat1	FAIL
TC_4.3_07	cvndoc2.endoc2.fndoc2.cvfdat2.ffdat2.cvsdat2	FAIL
TC_4.3_08	cvndoc2.endoc2.fndoc2.cvfdat2.ffdat2.cvsdat3.fsdat1	FAIL
TC_4.3_09	cvndoc2.endoc2.fndoc2.cvfdat2.ffdat2.cvsdat3.fsdat2.cdat1	FAIL
TC_4.3_10	cvndoc2.endoc2.fndoc2.cvfdat2.ffdat2.cvsdat3.fsdat2.cdat2	PASS

9.5 Selezione del formato dei report

Nota: il Test Case seguente si occupa di testare l'input dell'utente fornito da checkbox opzionali. Pertanto, nelle scelte di ogni categoria, sarà evidente che non è considerato errore né il caso in cui la checkbox è stata selezionata né il caso in cui non è stata selezionata. Di conseguenza, i singoli parametri verranno inseriti nel Test Case solo per evidenziare gli errori risultanti dalla combinazione degli stessi (es: non si può selezionare la checkbox della data di nascita se non è stata selezionata quella dell'anagrafica).

TC_5 Selezione formato Report

Parametro: Anagrafica - checkbox **Formato:** -	
Categorie	**Scelte**
FlagAnagrafica fan	1. flag selezionato [property **flagAnagraficaFANok**] 2. flag non selezionato [property **flagAnagraficaFANok**]

Parametro: Data di nascita - checkbox **Formato:** -	
Categorie	**Scelte**
FlagDDN fdn	1. flag selezionato [property **flagDDN_FDNok**] 2. flag non selezionato [property **flagDDN_FDNok**]

Parametro: Numero di Studenti controllati - checkbox **Formato:** -	
Categorie	**Scelte**
FlagNum fns	1. flag selezionato [property **flagNumStudenti_FNSok**] 2. flag non selezionato [property **flagNumStudenti_FNSok**]

Parametro: Numero GreenPass Validi - checkbox **Formato:** -	
Categorie	**Scelte**
FlagNumGPValidi fnv	1. flag selezionato [property **flagNumGPValidi_FNVok**] 2. flag non selezionato [property **flagNumGPValidi_FNVok**]

Parametro: Numero GreenPass Validi - checkbox **Formato:** -	
Categorie	**Scelte**
FlagNumGPNonValidi fni	1. flag selezionato [property **flagNumGPNonValidi_FNIok**] 2. flag non selezionato [property **flagNumGPNonValidi_FNIok**]

Codice	Combinazione	Esito
TC_5_01	fan2.fdn1	FAIL
TC_5_02	fan2.fdn2.fns2,fnv2,fni2	FAIL

Nota: per questo Test Case si è deciso di indicare solamente i casi di errore (visto che i casi di successo sono determinati dalle restanti combinazioni, e che pertanto non necessitano di testing).

10. Pianificazione del Testing e Assegnazione dei ruoli

Per documentare l'organizzazione adottata, in merito alle attività di pianificazione del testing e di assegnazione dei ruoli, presentiamo la tabella sottostante che funge da matrice di tracciabilità per facilitare l'identificazione delle informazioni. In particolare, le attività di testing inizieranno con il Testing Funzionale, quindi con l'individuazione dei Test Cases, e a seguire saranno realizzate tutte le altre categorie di testing descritte.

Test Case	Autore
TC_01	Martina Mulino
TC_02	Martina Mulino
TC_03	Viviana Rinaldi
TC_04	Alberto Montefusco
TC_05	Gennaro Spina