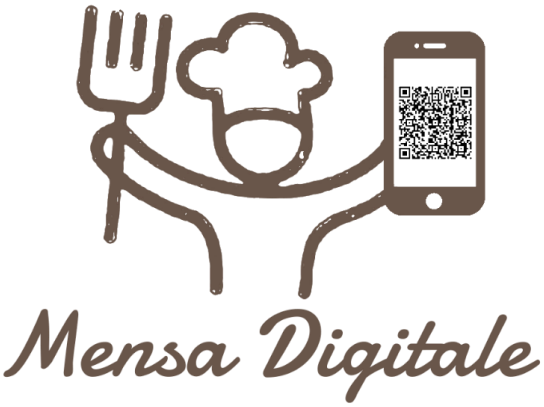

Test Plan Document

Mensa Digitale

Riferimento	
Versione	1.4
Data	18/12/2020
Destinatario	Prof.ssa F. Ferrucci
Presentato da	Francesco Capriglione
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
18/12/2020	1.0	Prima stesura Test Plan	Francesco Capriglione
29/12/2020	1.1	Modifica Test Case Inserimento fascia orario	Francesco Capriglione
03/01/2020	1.2	Correzione piè di pagina Modifica Test Case "Inserimento fascia oraria"	Francesco Capriglione
06/01/2020	1.3	Correzione TC (9.1.1, 9.1.2)	Francesco Capriglione
14/01/2020	1.4	Riadattamento Testing di sistema	Francesco Capriglione

Sommario

1. Introduzione

Durante la realizzazione di un software, ci si pone sempre l'obiettivo di ottenere un buon prodotto. Al fine di garantire che ciò avvenga, è necessario realizzare un sistema che possa essere considerato affidabile.

Nasce così la necessità di creare un prodotto che sia privo (o quasi) di errori prodotti durante la fase di implementazione per far si che il prodotto diventi uno strumento di cui l'utente finale si possa fidare.

A tal proposito è stato definito il seguente piano di test, il cui obiettivo è quello di analizzare e pianificare le attività di testing relative al sistema proposto.

Visto che è necessario garantire il corretto funzionamento del sistema, sono stati pensati input e casi di test specifici in modo da mettere alla prova le funzionalità offerte dal sistema stesso.

I risultati dei test, che verranno eseguiti successivamente, saranno fondamentali al fine di individuare le aree su cui bisogna intervenire per rimuovere i fault presenti all'interno del sistema.

2. Documenti correlati

2.1. Relazione con il documento di analisi

La progettazione dei casi di test avviene ignorando la struttura interna del sistema e operando solo sulle specifiche. Grazie agli scenari e agli use case prodotti nel documento di analisi otteniamo in maniera dettagliata le specifiche del sistema e per questo motivo è necessario far riferimento a questo documento.

2.2. Relazione con il System Design Document

Il sistema proposto, come si evince dal System Design Document è stato suddiviso in tre sottosistemi:

- GUI
- Business
- Storage

Il livello di business sarà il punto cruciale dei nostri test. Verranno pianificate attività di testing relative alle funzionalità specificate nei sottosistemi business, senza però trascurare quelle del sottosistema storage.

2.3. Relazione con l'Object Design Document

L'Object Design Document verrà utilizzato per la verifica dei contratti e dei comportamenti raffinati all'interno di tale documento.

3. Panoramica del sistema

Mensa Digitale è una piattaforma web che mira ad aumentare gli introiti dell'ADISURC e per farlo punta ad un alto tasso di soddisfazione degli studenti che accedono alla mensa, il tutto rispettando le normative vigenti in merito alla pandemia con cui stiamo convivendo.
Tenendo conto degli obiettivi sopra elencati, abbiamo individuato i seguenti sottosistemi per quanto riguarda il livello di business:

- Gestione domande di attivazione
- Gestione prenotazioni e fasce orarie
- Gestione piatti e menu
- Gestione utenza
- Gestione tessera, ricarica saldo e pagamento consumazione

4. Funzionalità da testare

Per quanto l'ideale sia testare ogni singolo componente del sistema, a causa del budget ridotto non è possibile pensare ad uno scenario del genere. Il team si occuperà pertanto di testare le funzionalità principali del sistema.

Le funzionalità che verranno testate sono riassunte nella seguente tabella:

Sottosistema	Funzionalità
Gestione domande di attivazione	Sottomissione di una domanda di attivazione dei servizi di ristorazione
	Valutazione di una domanda di attivazione dei servizi di ristorazione
Gestione prenotazioni e fasce orarie	Prenotazione accesso in mensa
	Inserimento fascia oraria
Gestione piatti e menù	Inserimento di un piatto
	Inserimento valutazione piatto
Gestione utenza	Accesso all'area personale

Non verranno invece testate:

- Sicurezza
- Performance

5. Criteri di Pass/Failed

L'approccio utilizzato per testare il sistema sarà del tipo test to fail. L'obiettivo che ci poniamo è quello di individuare quanti più fault possibili durante le fasi di sviluppo in modo che, una volta rilasciato il software, esso contenga quanti meno fault possibili. L'approccio test to fail ci aiuta in questo senso in quanto spinge la nostra soluzione ai suoi limiti, evitando così che la conoscenza di ciò che abbiamo realizzato inganni la nostra fase di testing. Piuttosto invece incita a provare a riprovare una funzionalità fino a quando un errore non è stato individuato.
Pertanto, il test viene marcato come PASS se il comportamento osservato è diverso da quello atteso. In questo caso analizzeremo la causa dell'errore e verrà risolto. Il test viene marcato come FAILED nel caso in cui non vengano scovati errori nelle componenti.

6. Approccio

Le attività da svolgere per realizzare il testing sono tre. Nella fase iniziale ci occuperemo di individuare gli errori su una singola componente; nella fase successiva invece, testeremo le funzionalità nate

dall'integrazione dei vari sottosistemi; infine, andremo a testare l'intero sistema per verificare che le caratteristiche richieste dal nostro committente siano rispettate o meno.

6.1. Testing di unità

Il testing di unità è necessario al fine di evidenziare gli errori delle singole componenti che compongono il nostro sistema. Il testing di unità si focalizza sostanzialmente sul comportamento di una componente e verrà svolto in modalità black-box. Questa scelta strutturerà il testing unitario in un'analisi Input/Ouput delle singole componenti andando ad astrarre la verifica della struttura interna. Per minimizzare i casi di test, gli input verranno divisi in classi di equivalenza e ogni componente avrà un singolo caso di test per ogni classe di equivalenza strutturata. In questa fase, quindi, si avrà particolare attenzione sulla suddivisione delle classi degli input così da poter verificare ogni componente su ogni possibile tipo di input del dominio. La gestione del caso di errore, ossia lo stato in cui il comportamento atteso non è equivalente al comportamento ottenuto, comporterà l'informare gli sviluppatori della presenza di un fault in modo tale da poterlo correggerlo tempestivamente per poi ripassare ad una verifica della correzione.

6.2. Testing di integrazione

Una volta rilevati e sistemati i fault per una singola componente, essa è pronta per essere integrata in un sottosistema più grande andando ad effettuare così il test di integrazione.
Per realizzarlo verrà utilizzata una strategia bottom-up.

6.3. Testing di sistema

Il testing di sistema concluderà la fase di test del prodotto ed il primo ciclo di sviluppo. Per questa tipologia di test, ci affidiamo all'utilizzo di un software ausiliario come Katalon al fine di osservare il comportamento del sistema.

7. Sospensione e ripresa

7.1. Criteri di sospensione

La fase di testing verrà sospesa nel momento in cui almeno il 10% dei casi di test riportano errori: in queste condizioni, il team deve provvedere a correggere i fault prima di procedere con l'implementazione o il testing di nuove funzionalità.

7.2. Criteri di ripresa

Visto che l'approccio per lo sviluppo del software è di tipo incrementale, la fase di testing potrebbe riprendere nel momento in cui verranno introdotti cambiamenti e/o nuove componenti. In questo caso, andranno testati i nuovi elementi introdotti e, tramite regression testing, anche quelli già precedentemente testati. Pertanto, faremo affidamento su un servizio che ci permetta di lavorare in un ambiente di Continous Integration

7.3. Criteri di terminazione

Il test si considera concluso nel momento in cui verrà raggiunta una branch coverage reputata sufficientemente alta, ricordando che il minimo è del 75%.

8. Materiale per il testing

L'esecuzione dei test necessita di un ambiente in cui siano installati Java SE 14 o successivi e MySQL.
Il testing viene effettuato utilizzando i framework JUnit, Mockito e Jacoco, molto affermati in ambiente Java, e il software Katalon.
Mentre JUnit viene utilizzato per il testing d'unità, Mockito viene utilizzato per mascherare le dipendenze tra le componenti. Jacoco verrà utilizzato per generare i report sui test. Katalon viene utilizzato per realizzare i test di sistema.
Come evidenziato prima, si lavorerà in un ambiente di continous integration: ciò è possibile grazie all'utilizzo di Travis CI

9. Test cases

In questa sezione, per ogni sottosistema verranno mostrate le funzionalità che verranno testate. Ogni funzionalità avrà una tabella per ognuno dei suoi parametri che conterrà i vincoli affinché il valore dell'input sia valido. Infine, vi sarà una tabella che va ad indicare i test case individuati e i relativi esiti attesi dalle combinazioni dei vari vincoli: in caso di errore l'esito sarà uguale a ✗, altrimenti sarà uguale a ✔.

9.1. Gestione domande di attivazione

9.1.1. Sottomissione di una domanda di attivazione dei servizi di ristorazione

Tra le funzionalità da testare per il sottosistema studenti è prevista la sottomissione di una domanda di attivazione dei servizi di ristorazione. Questa funzionalità prevede che uno studente del primo anno compili un modulo composto dai seguenti dati:

- Cognome: stringa con un numero di caratteri compreso tra 2 e 50 e composta da sole lettere
- Nome: stringa con un numero di caratteri compreso tra 2 e 20 e composta da sole lettere
- Data di nascita: data nel formato GG/MM/AAAA dove:
  - AAAA: numero intero compreso tra 1900 e 2020
  - MM: numero intero compreso tra 1 e 12
  - GG: numero intero compreso tra 1 e 31
- Provincia di nascita: stringa composta da esattamente due caratteri e appartenente all'insieme delle province italiane. Se la provincia è estera, allora dovrà essere inserita la stringa "EE"
- Comune di nascita: stringa con un numero di caratteri compreso tra 2 e 50 e composta da sole lettere
- Codice fiscale: stringa composta da esattamente 16 caratteri alfanumerici che rispettino il formato del codice fiscale italiano
- Cittadinanza: stringa con un numero di caratteri compreso tra 2 e 50 composta da sole lettere
- Rifugiato: valore booleano posto a vero se lo studente è un rifugiato, falso altrimenti

- Residenza nucleo familiare estero: valore booleano posto a vero se lo studente ha un nucleo familiare residente all'estero, falso altrimenti.
- Indirizzo: stringa con un numero di caratteri compreso tra 4 e 100
- Telefono: stringa con un numero di caratteri compreso tra 6 e 20
- Cellulare: stringa con un numero di caratteri compreso tra 6 e 20
- E-mail: stringa con un numero di caratteri compreso tra 5 e 150. Contiene una @.
- Conferma E-mail: stringa con un numero di caratteri compreso tra 5 e 150. Contiene una @. Deve essere identica a E-mail.

Il modulo richiede inoltre che vengano accettate:

- Prelazione dei dati relative alla condizione economica e al merito: valore booleano, deve essere posto sempre a true;
- Responsabilità e sanzioni penali derivanti da false dichiarazioni: valore booleano, deve essere posto sempre a true.

Parametro	Cognome
Formato	^[a-zA-Z]+$
LC - Lunghezza	1. < 2 or > 50 [error] 2. >= 2 and <= 50 [property LC_OK]
FC - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if LC_OK] [property FC_OK]

Parametro	Nome
Formato	^[a-zA-Z]+$
LN - Lunghezza	1. < 2 or > 20 [error] 2. >= 2 and <= 20 [property LN_OK]
FN - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if LN_OK] [property FN_OK]

Parametro	Data di nascita
Formato	^(?:(?:31(\/|-|\.)(?:0?[13578]|1[02]))\1|(?:(?:29|30)(\/|-|\.)(?:0?[13-9]|1[0-2])\2))(?:(?:1[6-9]|[2-9]\d)?\d{2})$\|^(?:29(\/\|-\|\.)0?2\3(?:(?:(?:1[6-9]\|[2-9]\d)?(?:0[48]\|[2468][048]\|[13579][26])\|(?:(?:16\|[2468][048]\|[3579][26])00))))$|^(?:0?[1-9]|1\d|2[0-8])(\/|-|\.)(?:(?:0?[1-9])|(?:1[0-2]))\4(?:(?:1[6-9]|[2-9]\d)?\d{2})$
FD - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [property FD_OK]

Parametro	Provincia di Nascita
Formato	^(AG | AL | AN | AR | AP | AT | AV | BA | BT | BL | BN | BG | BI | BO | BZ | BS | BR | CA | CL | CB | CI | CE | CT | CZ | CH | CO | CS | CR | KR | CN | EE | EN | FM | FE | FI | FG | FC | FR | GE | GO | GR | IM | IS | AQ | SP | LT | LE | LC | LI | LO | LU | MC | MN | MS | MT | VS | ME | MI | MO | MB | NA | NO | NU | OG | OT | OR | PD | PA | PR | PV | PG | PU | PE | PC | PI | PT | PN | PZ | PO | RG | RA | RC | RE | RI | RN | RM | RO | SA | SS | SV | SI | SR | SO | TA | TE | TR | TO | TP | TN | TV | TS | UD | AO | VA | VE | VB | VC | VR | VV | VI | VT)/i$
LPN - Lunghezza	1. != 2 [error] 2. == 2 [property LPN_OK]
FPN - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if LPN_OK][property FPN_OK]

Parametro	Comune
Formato	^[a-zA-Z]+$
LCN - Lunghezza	1. < 2 or > 50 [error] 2. >= 2 and <= 50 [property LCN_OK]
FCN - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if LCN_OK] [property FCN_OK]

Parametro	Codice Fiscale
Formato	/^(?:[A-Z][AEIOU][AEIOUX] | [B-DF-HJ-NP-TV-Z]{2}[A-Z]){2}(?:[\dLMNP-V]{2}(?:[A-EHLMPR-T](?:[04LQ][1-9MNP-V] | [15MR][\dLMNP-V] | [26NS][0-8LMNP-U]) | [DHPS][37PT][0L] | [ACELMRT][37PT][01LM] | [AC-EHLMPR-T][26NS][9V]) | (?:[02468LNQSU][048LQU] | [13579MPRTV][26NS])B[26NS][9V])(?:[A-MZ][1-9MNP-V][\dLMNP-V]{2} | [A-M][0L](?:[1-9MNP-V][\dLMNP-V] | [0L][1-9MNP-V]))[A-Z]$/i
FCF - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [property FCF_OK]

Parametro	Cittadinanza
Formato	^[a-zA-Z]+$
LCT - Lunghezza	1. < 2 or > 50 [error] 2. >= 2 and <= 50 [property LCT_OK]
FCT - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if LCT_OK] [property FCT_OK]

Parametro	Indirizzo
LI - Lunghezza	1. < 4 or > 100 [error] 2. >= 4 and <= 100 [property LI_OK]

Parametro	Telefono
LT - Lunghezza	1. < 6 or > 20 [error] 2. >= 6 and <= 20 [property LT_OK]

Parametro	Cellulare
LCLL - Lunghezza	1. < 6 or > 20 [error] 2. >= 6 and <= 20 [property LCLL_OK]

Parametro	E-Mail
Formato	^(([^<>()[\]\.,;:\s@\"]+(\.[^<>()[\]\.,;:\s@\"]+)*) | (\".+\"))@(([^<>()[\]\.,;:\s@\"]+\.)+[^<>()[\]\.,;:\s@\"]{2,})$/i
LE - Lunghezza	1. < 6 or > 150 [error] 2. >= 6 and <= 150 [property LE_OK]
FE - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if LE_OK] [property FE_OK]

Parametro	Ripeti E-Mail
VRE - Valore	1. Valore diverso da E-mail [error] 2. Valore uguale a E-mail [property VRE_OK]

Codice	Combinazione	Esito
TC_ARS_1:1	LC1	X
TC_ARS_1:2	LC2.FC1	X
TC_ARS_1:3	LC2.FC2.LN1	X
TC_ARS_1:4	LC2.FC2.LN2.FN1	X
TC_ARS_1:5	LC2.FC2.LN2.FN2.FD1	X
TC_ARS_1:6	LC2.FC2.LN2.FN2.FD2.LPN1	X
TC_ARS_1:7	LC2.FC2.LN2.FN2.FD2.LPN2.FPN1	X
TC_ARS_1:8	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN1	X
TC_ARS_1:9	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN2.FCN1	X
TC_ARS_1:10	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN2.FCN2.FCF1	X
TC_ARS_1:11	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN2.FCN2.FCF2.LCT1	X
TC_ARS_1:12	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN2.FCN2.FCF2.LCT2.FCT1	X
TC_ARS_1:13	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN2.FCN2.FCF2.LCT2.FCT2.LI1	X
TC_ARS_1:14	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN2.FCN2.FCF2.LCT2.FCT2.LI2.LT1	X
TC_ARS_1:15	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN2.FCN2.FCF2.LCT2.FCT2.LI2.LT2.LCLL1	X
TC_ARS_1:16	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN2.FCN2.FCF2.LCT2.FCT2.LI2.LT2.LCLL2.LE1	X

TC_ARS_1:17	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN2.FCN2.FCF2.LCT2.FCT2.LI2.LT2.LCLL2.LE2.FE1	X
TC_ARS_1:18	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN2.FCN2.FCF2.LCT2.FCT2.LI2.LT2.LCLL2.LE2.FE2.VRE1	X
TC_ARS_1:19	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN2.FCN2.FCF2.LCT2.FCT2.LI2.LE2.FE2.VRE2	✔
TC_ARS_1:20	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN2.FCN2.FCF2.LCT2.FCT2.LI2.LT2.LE2.FE2.VRE2	✔
TC_ARS_1:21	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN2.FCN2.FCF2.LCT2.FCT2.LI2.LCLL2.LE2.FE2.VRE2	✔
TC_ARS_1:22	LC2.FC2.LN2.FN2.FD2.LPN2.FPN2.LCN2.FCN2.FCF2.LCT2.FCT2.LI2.LT2.LCLL2.LE2.FE2.VRE2	✔

9.1.2. Valutazione di una domanda di attivazione dei servizi di ristorazione

Un'altra funzionalità è quella della valutazione di una domanda di attivazione. Questa funzionalità permette ad un addetto di approvare o rifiutare una domanda presentata da uno studente. In input l'addetto dovrà fornire:

- Esito della domanda: valore intero compreso tra 0 e 2. Allo stato iniziale l'esito è impostato a 0 ad indicare che la richiesta è in sospeso. Con valore 2 la domanda risulta rifiutata e con valore 1 la domanda risulta accettata.

Parametro	Esito della domanda
VE - Valore	1. != 1 and != 2[error] 2. == 1 or == 2 [property VE_OK]

Codice	Combinazione	Esito
TC_ARS_2:1	VE1	X
TC_ARS_2:2	VE2	✔

9.2. Gestione prenotazioni e fasce orarie

9.2.1. Prenotazione accesso in mensa

Tra le funzionalità da testare per il sottosistema Gestione prenotazioni e fasce orarie vi è la prenotazione dell'accesso in mensa. Questa funzionalità prevede che uno studente per effettuare una prenotazione compili un modulo composto dai seguenti dati:

- Fascia oraria: un intero compreso tra 1 e N, dove N è il numero di fasce orarie attualmente registrate nella base di dati.
- Sala: un intero compreso tra 1 e 5

Parametro	Fascia Oraria
VF - Valore	1. < 1 or > N[error] 2. >= 1 and <= N [property VE_OK]

Parametro	Sala
VS - Valore	1. < 1 or > 5[error] 2. >= 1 and <= 5 [property VS_OK]

Codice	Combinazione	Esito
TC_PSR_1:1	VF1	✗
TC_PSR_1:2	VF2.VS1	✗
TC_PSR_1:3	VF2.VS2	✔

9.2.2. Inserimento fascia oraria

Un'altra importante funzionalità da testare è quella dell'inserimento della fascia oraria. Qui l'admin dovrà specificare una nuova fascia oraria disponibile per la prenotazione nella forma HH:MM che dovrà essere distante almeno 40 minuti da una fascia oraria preesistente. Inoltre, la fascia oraria dovrà essere compresa tra le 9:00 e le 21:59.

Parametro	Fascia oraria
Formato	^(09 | 1[0-9] | 2[0-1]):[0-5][0-9]$
VFO - Valore	Sia v il valore corrente della fascia oraria inserita. Siano FO1 e FO2 tale FO1 < v < FO2. 1. (v – FO1) < 40 or (FO2 - v) < 40 [error] 2. (v – FO1) >= 40 and (FO2 - v) >= 40 [property VFO_OK]
FFO - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if VFO_OK] [property FFO_OK]

Codice	Combinazione	Esito
TC_PSR_2:1	VFO1	X
TC_PSR_2:2	VFO2.FFO1	X
TC_PSR_2:3	VFO2.FFO2	✔

9.3. Gestione piatti e menù

9.3.1. Inserimento di un piatto

Tra le funzionalità da testare per il sottosistema Gestione piatti e menu vi è l'operazione di inserimento di un piatto. Questa funzionalità prevede che un addetto per inserire un nuovo piatto compili un modulo composto dai seguenti dati:

- Ingredienti: una stringa con un numero di caratteri compreso tra 4 e 200.
- Calorie: un intero compreso tra 1 e 2000
- Proteine: un intero positivo
- Grassi: un intero positivo
- Sodio: un intero positivo
- Carboidrati: un intero positivo

Parametro	Ingredienti
Formato	^([A-Z ,])+$
LI - Lunghezza	1. < 4 or > 200 [error] 2. >= 4 and <= 200 [property LI_OK]
FI - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if LI_OK] [property FI_OK]

Parametro	Calorie
VC - Valore	1. < 0 [error] 2. >= 0 [property VC_OK]

Parametro	Proteine
VP - Valore	1. < 0 [error] 2. >= 0 [property VP_OK]

Parametro	Grassi
VG - Valore	1. < 0 [error] 2. >= 0 [property VG_OK]

Parametro	Sodio
VS - Valore	1. < 0 [error] 2. >= 0 [property VS_OK]

Parametro	Carboidrati
VCARB - Valore	1. < 0 [error] 2. >= 0 [property VCARB_OK]

Codice	Combinazione	Esito
TC_PM_1:1	LI1	X
TC_PM_1:2	LI2.FI1	X
TC_PM_1:3	LI2.FI2.VC1	X
TC_PM_1:4	LI2.FI2.VC2.VP1	X

TC_PM_1:5	LI2.FI2.VC2.VP2.VG1	X
TC_PM_1:6	LI2.FI2.VC2.VP2.VG2.VS1	X
TC_PM_1:7	LI2.FI2.VC2.VP2.VG2.VS2.VCARB1	X
TC_PM_1:8	LI2.FI2.VC2.VP2.VG2.VS2.VCARB2	✔

9.3.2. Inserimento valutazione piatto

Un'altra importante funzionalità da testare è quella dell'inserimento della valutazione di un piatto. Uno studente, con una prenotazione effettuata in mensa, può lasciare una valutazione sul piatto consumato inserendo una valutazione su uno dei piatti presenti nel menu odierno.

Parametro	**Piatto**
Formato	^([A-Z])+$
VP - Valore	1. Il valore non corrisponde ad un piatto presente nel menu [error] 2. Il valore corrisponde ad un piatto presente nel menu [property VP_OK]
FP - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if VP_OK] [property FP_OK]

Parametro	**Recensione**
VR - Valore	1. < 1 or > 5 [error] 2. >= 1 and <= 5 [property VR_OK]

Codice	Combinazione	Esito
TC_PM_2:1	VP1	X
TC_PM_2:2	VP2.FP1	X
TC_PM_2:3	VP2.FP2.VR1	X
TC_PM_2:4	VP2.FP2.VR2	✔

9.4. Gestione utenza

Tra le attività del sottosistema Gestione utenza abbiamo deciso di testare l'accesso alla piattaforma. L'accesso sarà effettuato tramite le API di Google OAuth 2.0. Tuttavia, l'accesso alla piattaforma sarà consentito solo se la mail appartiene al dominio @unisa.it oppure al dominio @studenti.unisa.it

Parametro	Email
VE - Email	1. Il dominio NON è @studenti.unisa.it and Il dominio NON è @unisa.it 2. Il dominio è @studenti.unisa.it or Il dominio è @unisa.it [property VE_OK]

Codice	Combinazione	Esito
TC_GU_1:1	VE1	X
TC_GU_1:2	VE2	✔

10. Riferimenti ad altri documenti di test

Documento	Riferimento
Test Case Specification	Combinazioni di input fornite al sistema
Test Execution Report	Risultati del testing funzionale
Test Incident Report	Fault individuati
Test Summary Report	Riassunto dei riscontri ottenuti