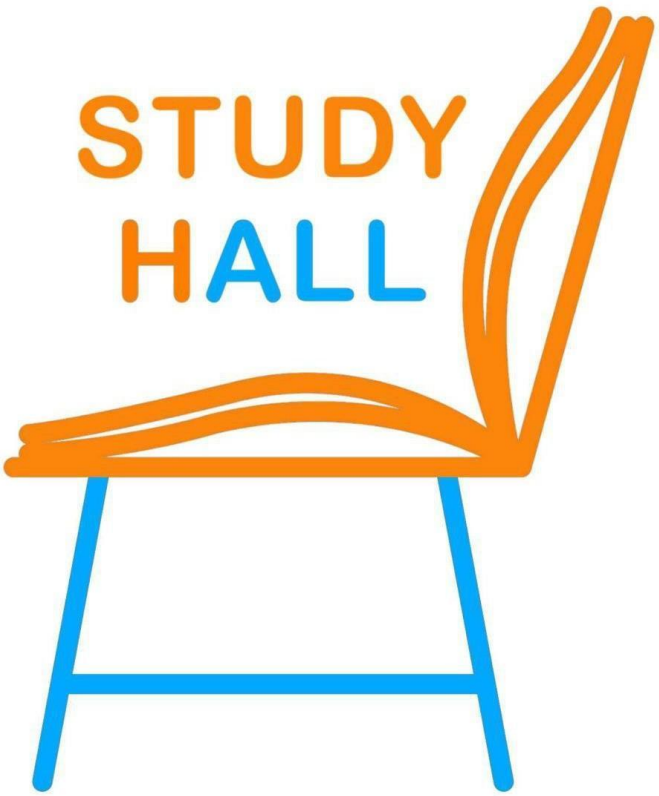
Test Plan



Studenti

Colella Maria Immacolata	05121**10431**
Mattiello Michele	05121**10185**
Sorrentino Carlo	05121**10884**

Test Plan

Test Plan

1. Introduzione

Il Test Plan è un documento che si focalizza sugli aspetti manageriali del testing gestendo lo sviluppo e le attività di testing effettuate sul sistema creato.
Saranno identificati gli elementi e le funzionalità da testare, le strategie di testing, gli strumenti utilizzati per effettuarlo. Si deve quindi verificare il funzionamento dell'intero sistema e dei relativi sottosistemi presi singolarmente.
Lo scopo del testing è, inoltre, quello di rilevare gli errori all'interno del codice realizzato, in modo che essi non si ripetano durante l'utilizzo da parte dell'utente finale.
I risultati dei test servono anche per rilevare bug e incongruenze del sistema ed intervenire per risolvere tali problemi.
Il risultato dovrà essere un miglioramento del sistema in tutta la sua interezza.

2. Documenti correlati

Il Test Plan richiede la conoscenza e l'utilizzo di tutti i documenti prodotti precedentemente. Infatti, i test case sono basati sulle funzionalità raccolte nel:
"Documento di raccolta ed Analisi dei Requisiti" (RAD) che contiene principalmente i requisiti funzionali, non funzionali, comportamento uguale al flusso di eventi dei vari casi d'uso, diagrammi e mockups;
"System Design Document" (SDD) nel quale vengono elencati e descritti i sottoinsiemi che effettivamente dovranno essere realizzati;
Object Design Document" (ODD) nel quale viene indicata la struttura dei packages e delle class interfaces da rispettare.

3. Funzionalità da testare

Va specificato che saranno trattati solo i test case relativi a casi d'uso NON banali, escludendo quindi dalla lista tutti quelli ritenuti poco utili da descrivere, e suddivisi per gestione del sistema.
Lista completa dei casi d'uso per i quali sarà effettuato test case:

Gestione Account

- Login Studente/Gestore
- Verifica studente/gestore loggato

Gestione Aula Studio Gestore

- Aggiunta Aula Studio
- Modifica Aula Studio

Gestioni Prenotazioni

- Prenotazione Aula Studio
- Cancellazione Prenotazione Studente

4. Pass / Fail Criteria

Lo scopo del testing è rilevare eventuali faults presenti nel sistema, quindi, il testing ha successo se vi è un'incongruenza tra l'output osservato e l'output atteso; una volta riscontrato un fault si interviene per correggerlo e si effettua di nuovo il testing per assicurarsi che la correzione non abbia prodotto effetti collaterali, producendo altri fault. Dovremmo ottenere il testing di tutti i requisiti ad alta priorità e non banali e raggiungere una file coverage pari o superiore al 50%.
In caso di eventuali modifiche bisogna rieseguire i test poiché potrebbero verificarsi nuovi faults.

5. Approccio

L'approccio della fase di testing si compone di tre fasi. Si inizia con il **testing di unità** che ha lo scopo di testare le componenti del sistema singolarmente, poi si passa al **testing di integrazione** in cui le componenti del sistema verranno combinate e testate come un unico gruppo ed infine, si effettua il **testing di sistema** che mira a verificare il funzionamento dell'intero sistema.
Per il **testing di unità** si utilizzerà la tecnica "BLACK-BOX" attraverso la quali si osserva semplicemente il comportamento dell'input/output delle singole componenti senza tener conto della loro struttura interna.
A causa dell'ingente numero di dati verrà utilizzata la strategia del "Category Partition", che consente di decomporre lo spazio di input in categorie per poi partizionare le "categorie" in classi di equivalenza chiamate "scelte".
Al termine saranno specificate le "combinazioni" delle scelte da testare creando delle istanze di casi di test specificando i valori dei dati effettivi per ciascuna scelta e determinare i risultati corrispondenti. Mediante il "Category Partition" si otterrà un test efficiente e privo di ridondanze.
Per il **testing d'integrazione** si utilizzerà la tecnica "WHITE-BOX". La "WHITE-BOX" testing viene utilizzata per testare la logica interna del sottosistema o dell'oggetto considerato, infatti, per trovare un errore nel codice bisogna usare dei dati che percorrono la parte errata del programma. Per testare una parte di programma si introduce il concetto di "cammino", ovvero, una sequenza di istruzioni attraversata durante un'esecuzione. Non esiste un criterio capace di testare ogni singolo cammino
(dato l'elevato numero di questi ultimi), è possibile trovare un numero finito di cammini indipendenti che combinati tra loro forniscono la maggior parte dei restanti cammini.
Per determinare se il sistema rispecchia tutti i requisiti funzionali e globali allora sarà effettuato il testing tramite il tool "Appium".

6. Sospensione e ripresa

La fase di testing verrà sospesa al raggiungimento di una percentuale di branch coverage pari almeno al 50%.
Quando il testing rileva un errore viene sospeso e la componente viene corretta facendo ripartire il processo di testing che dovrà ripetere l'intero processo per rilevare eventuali errori introdotti dalla correzione stessa.
La ripresa avverrà soltanto quando saranno risolti i problemi che ne hanno determinato la sospensione e riprende dal test case che ne ha causato la sospensione.

7. Materiale per il testing

Gli strumenti utilizzati sono:

- NodeJS, Docker, mongodb image per testare il server in locale
- Xcode e simulatore iPhone per testare l'app su dispositivi ios
- Android studio e simulatore android per testare l'app su dispositivi android
- Client-web per effettuare richieste al server
- Appium
- Jest

Il sistema verrà testato sui computer configurati dal team di sviluppo senza particolari richieste riguardo le componenti hardware; è comunque richiesto un dispositivo con installato Docker per avviare il server mentre per testare l'app mobile è richiesto o android studio oppure Xcode. Appium verrà utilizzato per il test automatico di sistema.
È infine richiesta una connessione ad Internet abbastanza performante da permettere all'app di comunicare con il server.
Le routes saranno testate utilizzando la libreria Jest e dato l'utilizzo di Mongoose non vi sarà la necessità di testare l'interfaccia per la comunicazione al backend visto che viene fornita Mongoose.

8. Test Cases

Gestione Account

TC_Login Gestore

Parametro: Email Associazione	
Categorie	Scelte
Esistenza su Firebase Auh - ESF	1. Non esiste su Firebase Auth [property NotESF] 2. Esiste su Firebase Auth [property ESF]
Formato email - FE	1. Non rispetta il formato [property invalidFEValue] 2. Rispetta il formato [property validFEValue]
Lunghezza email - LE	1. Lunghezza = 0 – campo vuoto [property invalidLEValue] 2. Lunghezza >= 1 [property validLEValue]

Parametro: Password	
Categorie	Scelte
Associazione all'utente - AS	1. Non è correttamente associata alla email [property invalidASValue] 2. È correttamente associata alla email[if EXUS] [property validASValue]
Lunghezza password - LPX	1. Lunghezza < 8 – campo non valido [property invalidLPXValue] 2. Lunghezza >= 8 [property validLPXValue]

Codice	Combinazione	Esito
TC_LoginSuccess	ESF2.FE2.LE2.AS2.LPX2	"Login Organization"
TC_LoginFailed1	ESF1.FE2.LE2.AS2.LPX2	Errore "Email non registrata”
TC_LoginFailed2	ESF2.FE1.LE2.AS2.LPX2	Errore "Credenziali Errate"
TC_LoginFailed3	ESF2.FE2.LE1.AS2.LPX2	Errore “Campi Richiesti”
TC_LoginFailed4	ESF2.FE2.LE2.AS1.LPX2	Errore “Password errata”
TC_LoginFailed5	ESF2.FE2.LE2.AS2.LPX1	Errore “Password troppo corta”

TC_VerifyToken Gestore

Parametro: Token	
Categorie	Scelte
Esistenza Token	1. Non è presente [property NotETG] 2. E’ presente [property ETG]
VerifyToken Firebase	1. Ritorna undefined [property NotVF] 2. Ritorna email gestore [property VF]

Codice	Combinazione	Esito
TC_VerifyTokenSuccess	ET2, VF2	"Verify Token Success"
TC_VerifyTokenFailed1	ET1, VF2	Errore "Token richiesto”
TC_VerifyTokenFailed2	ET2, VF1	Errore "Token non valido"

TC_Login Studente

Parametro: username	
Categorie	Scelte
Lunghezza username - LU	1. Lunghezza = 0 – campo vuoto [property invalidLUValue] 2. Lunghezza >= 1 [property validLUValue]
Verifica Esse3 - VES	1. Utente non registrato su Esse3 [property invalidVESValue] 2. Utente registrato su Esse3 [property validVESValue]

Parametro: password	
Categorie	Scelte
Associazione all'utente - ASZ	1. Non è correttamente associata alla username [property invalidASZValue] 2. È correttamente associata all'username [property validASZValue]
Lunghezza password - LPXZ	1. Lunghezza < 8 – campo vuoto [property invalidLPXZValue] 2. Lunghezza >= 8 [property validLPXZValue]

Codice	Combinazione	Esito
TC_LoginSuccess	LU2, VES2, ASZ2, LPXZ2	"Login student"
TC_LoginFailed1	LU1, VES2, ASZ2, LPXZ2	Errore "Campo richiesto"
TC_LoginFailed2	LU2, VES1, ASZ2, LPXZ2	Errore "Username errato"
TC_LoginFailed3	LU2, VES2, ASZ1, LPXZ2	Errore "Password errata"
TC_LoginFailed4	LU2, VES2, ASZ2, LPXZ1	Errore "Campo richiesto"

TC_CheckLogon Studente

Parametro: Token	
Categorie	Scelte
Esistenza Token Studente	1. Non è presente [property NotETS] 2. E’ presente [property ETS]
checkLogon Esse3	1. Ritorna ok = false [property NotCL] 2. Ritorna ok = true [property CL]

Codice	Combinazione	Esito
TC_VerifyTokenSuccess	ETS2, CL2	"Verify Token Success"
TC_VerifyTokenFailed1	ETS1, CL2	Errore "Token richiesto”
TC_VerifyTokenFailed2	ETS2, CL1	Errore "Token non valido"

TC_Aggiunta Aula Studio

Parametro: Nome	
Categorie	Scelte
Lunghezza Nome - LN	1. Lunghezza = 0 – campo vuoto [property invalidLNValue] 2. Lunghezza >= 1 [property validLNValue]
Esistenza Univoca - EU	1. Esiste già un nome uguale a quello inserito [property invalidEUValue] 2. Il nome è unico nel database [property validEUValue]

Parametro: Edificio	
Categorie	Scelte
Inserimento Edificio - IE	1. Lunghezza = 0, campo vuoto [property invalidIEValue] 2. Inserito [property validIEValue]
Esistenza nel database - END	1. Non esiste nel database [property NotEND] 2. Esiste nel database [property END]

Parametro: Piano	
Categorie	Scelte
Formato Piano - FP	1. Non rispetta il formato [property invalidFPValue] 2. Rispetta il formato [property validFPValue]
Lunghezza Piano - LP	1. NumeroPiano < 0 [property invalidLPValue] 2. 0 < NumeroPiano < 9 [property validLPValue] 3. NumeroPiano > 9 [property invalidLPValue]

Parametro: Posti	
Categorie	Scelte
Formato Posti - FPS	1. Non rispetta il formato [property invalidFPSValue] 2. Rispetta il formato [property validFPSValue]
Lunghezza Posti - LPS	1. NumeroPosti < 1 [property invalidLPSValue] 2. 0 < NumeroPosti < 99 [property validLPSValue] 3. NumeroPosti > 99 [property invalidLPSValue]

Parametro: Imagine di Copertina	
Categorie	Scelte
Inserimento Imagine - II	1. Non Inserito [property invalidIIValue] 2. Inserito [property validIIValue]

Codice	Combinazione	Esito
TC_AggiuntaAulaStudioSuccess	LN2.EU2.IE2.END2.FP2.LP2.FPS2.LPS2.II2, RUX2	“StudyRoomCreated”
TC_AggiuntaAulaStudioFailed1	LN1.EU2.IE2.END2.FP2.LP2.FPS2.LPS2.II2, RUX2	Errore “Campo Richiesto”
TC_AggiuntaAulaStudioFailed2	LN2.EU1.IE2.END2.FP2.LP2.FPS2.LPS2.II2	Errore “Campo Preesistente”
TC_AggiuntaAulaStudioFailed3	LN2.EU2.IE1.END2.FP2.LP2.FPS2.LPS2.II2	Errore “Campo Richiesto”
TC_AggiuntaAulaStudioFailed4	LN2.EU2.IE2.END1.FP2.LP2.FPS2.LPS2.II2	Errore “Edificio non valido”
TC_AggiuntaAulaStudioFailed5	LN2.EU2.IE2.END2.FP1.LP2.FPS2.LPS2.II2	Errore “Formato non valido”
TC_AggiuntaAulaStudioFailed6	LN2.EU2.IE2.END2.FP2.LP1/3.FPS2.LPS2.II2	Errore “Piano non valido”
TC_AggiuntaAulaStudioFailed7	LN2.EU2.IE2.END2.FP2.LP2.FPS1.LPS2.II2	Errore “Formato non valido”
TC_AggiuntaAulaStudioFailed8	LN2.EU2.IE2.END2.FP2.LP2.FPS2.LPS1/3.II2	Errore “numero posti non valido”
TC_AggiuntaAulaStudioFailed9	LN2.EU2.IE2.END2.FP2.LP2.FPS2.LPS2.II1	Errore “campo richiesto”

TC_Modifica Aula Studio

Parametro: Nome	
Categorie	Scelte
Lunghezza Nome - LNZ	1. .Lunghezza = 0 – campo vuoto [property invalidLNValue] 2. Lunghezza >= 1 [property validLNValue]
Esistenza Univoca - EUZ	1. Esiste già un nome uguale a quello inserito [property invalidEUValue] 2. Il nome è unico nel database [property validEUValue]

Parametro: Edificio	
Categorie	Scelte
Inserimento Edificio - IEZ	1. Lunghezza = 0, campo vuoto [property invalidIEValue] 2. Inserito [property validIEValue]
Esistenza nel database - ENDZ	1. Non esiste nel database [property NotEND] 2. Esiste nel database [property END]

Parametro: Piano	
Categorie	Scelte
Formato Piano - FPZ	1. Non rispetta il formato [property invalidFPValue] 2. Rispetta il formato [property validFPValue]
Lunghezza Piano - LPZ	1. NumeroPiano < 0 [property invalidLPValue] 2. 0 < NumeroPiano < 9 [property validLPValue] 3. NumeroPiano > 9 [property invalidLPValue]

Parametro: Posti	
Categorie	Scelte
Formato Posti - FPSZ	1. Non rispetta il formato [property invalidFPSValue] 2. Rispetta il formato [property validFPSValue]
Lunghezza Posti - LPSZ	1. NumeroPosti < 1 [property invalidLPSValue] 2. 0 < NumeroPosti < 99 [property validLPSValue] 3. NumeroPosti > 99 [property invalidLPSValue]

Parametro: Imagine di Copertina	
Categorie	Scelte
Inserimento Imagine - IIZ	1. Non Inserito [property invalidIIValue] 2. Inserito [property validIIValue]

Codice	Combinazione	Esito
TC_ModificaAulaStudioSuccess	LNZ2.EUZ2.IEZ2.ENDZ2.FPZ2.LPZ2. FPSZ2.LPSZ2.IIZ2	“StudyRoomCreated”
TC_ModificaAulaStudioFailed1	LNZ1.EUZ2.IEZ2.ENDZ2.FPZ2.LPZ2. FPSZ2.LPSZ2.IIZ2	Errore “Campo Richiesto”
TC_ModificaAulaStudioFailed2	LNZ2.EUZ1.IEZ2.ENDZ2.FPZ2.LPZ2. FPSZ2.LPSZ2.IIZ2	Errore “Campo Preesistente”
TC_ModificaAulaStudioFailed3	LNZ2.EUZ2.IEZ1.ENDZ2.FPZ2.LPZ2. FPSZ2.LPSZ2.IIZ2	Errore “Campo Richiesto”
TC_ModificaAulaStudioFailed4	LNZ2.EUZ2.IEZ2.ENDZ1.FPZ2.LPZ2. FPSZ2.LPSZ2.IIZ2	Errore “Edificio non valido”
TC_ModificaAulaStudioFailed5	LNZ2.EUZ2.IEZ2.ENDZ2.FPZ1.LPZ2. FPSZ2.LPSZ2.IIZ2	Errore “Formato non valido”
TC_ModificaAulaStudioFailed6	LNZ2.EUZ2.IEZ2.ENDZ2.FPZ2.LPZ1/ 3.FPSZ2.LPSZ2.IIZ2	Errore “Piano non valido”
TC_ModificaAulaStudioFailed7	LNZ2.EUZ2.IEZ2.ENDZ2.FPZ2.LPZ2. FPSZ1.LPSZ2.IIZ2	Errore “Formato non valido”
TC_ModificaAulaStudioFailed8	LNZ2.EUZ2.IEZ2.ENDZ2.FPZ2.LPZ2. FPSZ2.LPSZ1/3.IIZ2	Errore “numero posti non valido”
TC_ModificaAulaStudioFailed9	LNZ2.EUZ2.IEZ2.ENDZ2.FPZ2.LPZ2. FPSZ2.LPSZ2.IIZ1	Errore “campo richiesto”

Gestione Prenotazioni

TC_Prenotazione Aula Studio

Parametro: Data	
Categorie	Scelte
Formato Data - FD	1. Non rispetta il formato [property invalidFDValue] 2. Rispetta il formato [property validFDValue]
Validità Data - VD	1. Scelta <= Data Corrente [property invalidVDValue] 2. Scelta > Data Corrente [property validVDValue]
Validità Giorno - VG	1. giorno = 6° || 7° della settimana [property invalidVDValue] 2. giorno = 1° || 2° || 3° || 4° || 5° della settimana [property validVDValue]

Parametro: Start - End	
Categorie	Scelte
Fascia Oraria - FO	1. Start - End != 9:00 - 11:00 || 11:00 - 13:00 || 14:00 - 16:00 || 16:00 - 18:00[property invalidFAOValue] 2. Start - End = 9:00 - 11:00 || 11:00 - 13:00 || 14:00 - 16:00 || 16:00 - 18:00 [property validFAOValue]

Codice	Combinazione	Esito
TC_PrenotazioneAulaStudioSuccess	FD2,VD2,VG2,FO2	Reservation Create
TC_PrenotazioneAulaStudioFailed1	FD1,VD2,VG2,FO2	Errore “Data non valida”
TC_PrenotazioneAulaStudioFailed2	FD2,VD1,VG2,FO2	Errore “Data non valida”
TC_PrenotazioneAulaStudioFailed3	FD2,VD2,VG1,FO2	Errore “Giorno non disponibile”
TC_PrenotazioneAulaStudioFailed4	FD2,VD2,VG2,FO1	Errore “Fascia oraria non valida”

9. Scelte progettuali e chiarimenti

Le funzioni non analizzate sono le seguenti:

- Attivare/Disattivare aula studio
- Aggiunta/Rimozione aula studio ai preferiti
- Cancellare aula studio

Il corretto funzionamento di queste sarà determinato dal corretto funzionamento delle funzioni da cui dipendono. Tali funzioni utilizzate nel codice non vengono analizzate in quanto utilizzano parametri presi dai dati che ritornano le seguenti funzioni:

- checkLogon
- verifyToken

Le funzioni sopra citate hanno tutte come unico parametro (oltre all'id dell'aula) il ruolo di chi compie l'azione, quindi se è un gestore o uno studente.

Abbiamo deciso di non analizzare, poiché utilizzano parametri i quali vengono presi dai risultati che ritornano le già citate funzioni checkLogon e verifyToken, sui quali noi effettuiamo testing ed assicuriamo il corretto funzionamento e in caso di fallimento il corretto flusso di terminazione con errore.

Quindi se queste due funzioni già verificano la veridicità del ruolo di chi le richiama, nel senso che controllano se chi le sta usando sia un gestore o uno studente, pensiamo sia ridondante verificarle dato che prima di essere chiamate si affidano al controllo di checkLogon e verifyToken.