TraceX

Equipe de projet L2S1

Projet TraceX

Plan de tests

Version du document :	A.03
Date de document :	19/05/2025
Soumis le :	19/05/2025
Auteurs :	ALLAHOUM Abdelmalek Said, HUANG Maxime, KIM Léa, ZHENG Jacques
Type de diffusion :	Document électronique (.pdf)
Confidentialité :	Réservé aux étudiants UFR Maths-Info de l'Université Paris Cité

PROJET TRACEX – EQUIPE DE PROJET L2S1
04/05/2025

Table des matières

1. Introduction

Ce plan de test définit les stratégies de validation du logiciel de traçabilité automatique des exigences. Ce logiciel vise à extraire, analyser et relier les exigences issues de documents de type SSS et SRS en se basant sur leurs styles et contenus pour assurer une traçabilité descendante et ascendante.

1.1. Objectifs et méthodes

- **Vérifier l'extraction correcte des exigences depuis les fichiers Word (.docx). Qui représentent les documents de spécifications**. Cela consiste à tester si les exigences sont bien reconnues par le logiciel, en fonction de critères de style comme la police, la couleur ou la mise en forme.
- **Contrôler la cohérence des liens entre les documents SSS et SRS**. Les exigences du document SRS doivent pouvoir être reliées de manière automatique à leurs sources dans le SSS.
- **S'assurer que les liens de traçabilité sont bien visibles dans l'interface graphique**. Le but ici est de vérifier que l'utilisateur peut facilement consulter les relations entre les différents niveaux d'exigences (SSS → SRS → SDD) à travers une interface claire, sans bugs d'affichage ou d'interaction.
- **Couvrir l'ensemble du système avec des tests adaptés**, en procédant à :

  - Des **tests unitaires** pour valider le comportement isolé de chaque classe/fonction.
  - Des **tests d'intégration** pour tester la coopération entre modules.
  - Des **tests fonctionnels** pour simuler des parcours utilisateur complets et vérifier que l'application répond aux besoins décrits dans les cas d'usage.

1.2. Documents de référence

- [A02] Cahier des charges
- [A02] Cahier de recettes
- [B03] Document de conception générale
- [B03] Document de conception détaillée
- [A01] Manuel d'utilisation
- [A01] Manuel d'installation

2. Guide de lecture

Ce document a été rédigé pour différents types de lecteurs impliqués dans le projet:

- **Développeurs**

Les développeurs peuvent consulter les parties sur les tests unitaires et les tests d'intégration afin de vérifier que chaque partie du programme fonctionne correctement. Cela peut aussi les aider à détecter d'éventuels bugs et à mieux comprendre le comportement du code.

- **Responsable et Personnel administratif**

Les responsables et personnel administratif peuvent se concentrer sur les tests fonctionnels et la procédure de validation. Ces sections montrent si l'application répond bien aux besoins définis au départ et si elle fonctionne comme prévu.

- **Utilisateurs finaux**

Les utilisateurs finaux peuvent regarder les résultats des tests fonctionnels pour vérifier que les fonctionnalités importantes sont bien présentes et faciles à utiliser. Cela leur permet de se rassurer sur la qualité générale du logiciel avant une éventuelle mise en production.

2.1. Maîtrise d'œuvre

- L'équipe du projet est responsable du développement et de la livraison du produit final.

2.2. Maîtrise d'ouvrage

- Les représentants du client ou les utilisateurs finaux valident le bon fonctionnement du produit.

3. Concepts de base

Les concepts de base nécessaires à la compréhension du document:

- **Exigence** : besoin formulé dans un document SSS ou SRS.
- **Traçabilité** : capacité à retrouver l'origine d'une exigence ou sa déclinaison.
- **Référentiel** : document source (ex. : SSS) utilisé comme base de comparaison.
- **Hiérarchie des documents** :

Dans ce projet, les fichiers sont organisés ainsi :

→ Un ou plusieurs SSS
→ Chacun peut avoir un ou plusieurs SRS
→ Chaque SRS est lié à un SDD

4. Tests fonctionnels

Cette section présente les **tests fonctionnels** réalisés pour vérifier que l'application respecte bien les fonctionnalités définies dans le cahier des charges. Chaque test est associé à un **scénario d'utilisation** et permet de valider le bon fonctionnement du système du point de vue de l'utilisateur. Chaque test comporte :

- **Identifiant** : Un code unique pour chaque scénario.
- **Préconditions** : Ce qui doit être mis en place avant le test.

- **Principe de réalisation :** Étapes à suivre par l'utilisateur pour réaliser le test.
- **Résultat attendu** : ce que le système doit faire

Pour chaque scénario :

4.1. Identification

Un identifiant unique est attribué à chaque scénario

- **Identifiant –** *TF01 :*
  - Scénario : Importer des fichiers au format doc, docx, xlsx, xls ou csv.
- **Identifiant –** *TF02 :*
  - Scénario : Sélection des styles utilisés pour identifier les exigences dans des fichiers au format docx.
- **Identifiant** – *TF03* :
  - Scénario : Vérification de la présence des exigences définies dans les documents.
- **Identifiant** – *TF04* :
  - Scénario : Calcul du taux de traçabilité.
- **Identifiant** – *TF05* :
  - Scénario : Affichage du graphe.
- **Identifiant –** *TF06 :*
  - Scénario : Affichage des statistiques.
- **Identifiant –** *TF07 :*
  - Scénario : Filtrage des documents selon des critères choisis
- **Identifiant –** *TF08 :*
  - Scénario : Télécharger le rapport récapitulatif en format CSV.

4.2. Description

Identifiant – TF01 :

1) **Scénario :**

- Importation et structuration des fichiers au format doc, docx, xlsx, xls ou csv.

2) **Objectif :**

- Vérifier que l'application permet d'importer des fichiers dans ces formats sans générer d'erreurs, et qu'ils s'affichent correctement dans le tableau pour que l'utilisateur puisse structurer les documents.

3) **Principe de réalisation :**

- Lancer l'application et accéder à la page d'importation.
- Importer cinq fichiers de test, un pour chaque format : doc, docx, xls, xlsx et csv.
- Structurer l'arbre de données avec les fichiers importés pour chaque cas (SSS sans fils, un SSS avec SRS et sans SDD..).

Identifiant – TF02

1) **Scénario** :

- Sélection du style utilisé pour identifier les exigences dans des fichiers au format docx et doc.

2) **Objectif** :

- Ce test a pour objectif de d'assurer l'affichage correcte des styles et la sélection du style utilisé pour identifier les exigences dans des fichiers au format docx et doc.

3) **Principe de réalisation du test :**

- Appuyer sur le bouton "Select a style".

Identifiant – TF03

1) **Scénario :**

- Vérification de la présence des exigences définies dans les documents.

2) **Objectif :**

- Ce test a pour objectif de s'assurer que les exigences attendues sont bien extraites, visibles et correctes après la sélection.

3) **Principe de réalisation :**

- Sélectionnez le style permettant d'identifier les exigences selon le type de document
- Appuyer sur le bouton "Appuyer sur le bouton "extract requirements".

Identifiant – TF04 :

1) **Scénario :**

- Calcul du taux de traçabilité.

2) **Objectif :**

- Ce test a pour objectif de vérifier le bon calcul du taux de traçabilité entre les exigences des documents SSS, SRS et SDD.

3) **Principe de réalisation :**

- Sélectionnez le style permettant d'identifier les exigences selon le type de document
- Appuyer sur le bouton "Appuyer sur le bouton "extract requirements".

Identifiant – TF05 :

1) **Scénario :** Affichage d'un graphe dynamique sous forme d'arbres dont :

- Le premier niveau (noeuds bleus) représente les documents de type SSS
- Le second niveau (nœuds verts) représente les documents de type SRS
- Le troisième niveau (noeuds jaunes) représente les documents de type SDD
- Un dernier niveau met en évidence les documents de types SSS qui n'ont aucun SRS, représentés par des nœuds clignotants.
- Les nœuds des trois premiers niveaux sont connectés par des arêtes représentant le taux de traçabilité. Si aucune arête n'est affichée, le taux sera égal à 0 % et sera connecté par une ligne discontinue.

2) **Objectif :**

- Ce test a pour but de garantir un affichage correct et interactif du graphe de traçabilité des différents documents, une traçabilité fiable ainsi que celui des exigences associées à chacun d'eux.

3) **Principe de réalisation :**

- Appuyer sur le bouton "Graph" dans la barre de navigation.

Identifiant – TF06 :

1) **Scénario :**

   - Affichage des statistiques.

2) **Objectif :**

   - Ce test a pour objectif de s'assurer dans la page des statistiques le nombre correct des liens de traçabilité entre différents documents sous la forme d'un histogramme.

3) **Principe de réalisation :**

   - Appuyer sur le bouton "Statistics" dans la barre de navigation.

Identifiant – TF07

1) **Scénario :**

   - Filtrage des documents selon des critères choisis.

2) **Objectif :**

   - Tester la fonctionnalité de filtrage dynamique selon des critères (par expression régulière, cible, développé ou non, etc.)

3) **Principe de réalisation :**

   - Appuyer sur le bouton "Statistics" dans la barre de navigation.

Identifiant – TF08

1) Scénario :

   - Télécharger le rapport récapitulatif.

2) Objectif :

   - Ce test a pour objectif de télécharger un document .csv qui résume le graphe sous la forme d'un document .csv.

3) **Principe de réalisation :**

   - Appuyé sur "Download report" pour pouvoir télécharger un document .csv du rapport.

4.3. Contraintes

Les scénarios de test est soumis à certaines contraintes spécifiques qui doivent être respectées pour garantir sa validité :

- Contraintes techniques :
  - Système d'exploitation : Windows 10/11 64 bits, macOS ou Linux.
  - Fichiers conformes aux formats Office Open XML.
- Contraintes de fichiers :
  - Les fichiers test doivent suivre un **format** avec un nommage cohérent (ex : SSS_client.docx, SRS.xlsx, etc…).
  - Un fichier au format .doc contenant des exigences définies en un style créé par l'utilisateur.
  - Un fichier au format .docx contenant des exigences définies en un style créé par l'utilisateur.
  - Un fichier au format .xls contenant des exigences.
  - Un fichier au format .xlsx contenant des exigences.
  - Un fichier au format .csv contenant structuré avec une colonne contenant des exigences .
- Contraintes humaines :
  - L'utilisateur qui effectue le test doit connaître les **règles de structure** expliquées dans le manuel d'utilisation (ex : un SRS doit obligatoirement avoir un SDD).

4.4. Dépendances

1) **TF01 →** L'utilisateur doit avoir des fichiers compatibles (formats : doc, docx, xls, xlsx, csv) pour pouvoir les importer et les structurer.
2) **TF02 →** Dépend de TF01 : L'importation réussie de fichiers (TF01) est nécessaire pour sélectionner et identifier les styles dans les fichiers docx et doc.
3) **TF03 →** Dépend de TF02 : Ce test ne peut être effectué qu'après avoir sélectionné un style pour les exigences dans les fichiers docx ou doc (TF02).
4) **TF04 →** Dépend de TF03 : Le calcul du taux de traçabilité (TF04) nécessite la présence des exigences extraites et vérifiées dans le scénario TF03.
5) **TF05 →** Dépend de TF04 : L'affichage du graphe dynamique (TF05) repose sur le calcul correct du taux de traçabilité (TF04).

6) **TF06 →** Dépend de TF05 : L'affichage des statistiques (TF06) se base sur le graphe dynamique et les liens de traçabilité (TF05).

7) **TF07 →** Dépend de TF06 : Le filtrage des documents (TF07) nécessite d'avoir le graphe déjà affiché dans la scène (TF05).

8) **TF08 →** Dépend de TF06 : Le téléchargement du rapport récapitulatif (TF08) nécessite de l'affichage du graphe (TD05) et du calcul du taux de traçabilité pour garantir des liens corrects (TD4).

4.5. Procédure de test

Test Fonctionnel	Résultats attendus	Critères de validation
TF01 - Importation et structuration des fichiers	Les fichiers .doc, .docx, .xls, .xlsx, .csv sont bien importés et affichés dans la liste sans erreur. L'utilisateur doit pouvoir structurer ses fichiers dans l'arbre de données	L'application doit afficher chaque fichier correctement dans la liste, sans messages d'erreur ni plantage. Le système de glisser-déposer doit pouvoir fonctionner.
TF02 - Sélection du style	L'utilisateur voit la fenêtre avec les styles définis dans le fichier .docx, permettant d'identifier les exigences.	La fenêtre doit s'afficher correctement et tous les styles présents dans le fichier doivent être listés.
TF03 - Vérification des exigences	Les exigences définies dans le fichier .docx sont bien extraites et affichées correctement.	Les exigences extraites doivent correspondre à celles du fichier et être affichées de manière claire dans l'interface.
TF04 - Calcul du taux de traçabilité	Le taux de traçabilité doit être calculé et affiché en fonction des liens entre les documents (SSS, SRS, SDD).	Le taux doit être calculé correctement, et l'affichage du pourcentage doit être précis sans erreurs.
TF05 - Affichage du graphe	Un graphe dynamique est affiché avec des nœuds représentant les documents (SSS, SRS, SDD), et les liens entre eux sont visibles.	Le graphe doit montrer les relations entre les documents et les nœuds doivent être connectés avec les arêtes représentant les taux de traçabilité.
TF06 - Affichage des statistiques	Les statistiques sur le nombre de liens de traçabilité sont affichées sous forme d'histogramme.	L'histogramme doit afficher correctement le nombre de liens et être lisible sans erreurs.
TF07 - Filtrage des documents	L'utilisateur peut filtrer les documents en fonction de critères comme l'expression	Le filtrage doit être fonctionnel et permettre de trier les documents en fonction des

	régulière ou d'autres filtres définis.	critères choisis sans causer de problème.
TF08 - Téléchargement du rapport	L'utilisateur peut télécharger un fichier CSV contenant le rapport récapitulatif des documents et de leur traçabilité.	Le fichier CSV doit être généré correctement et contenir toutes les informations nécessaires sur les documents et leur traçabilité.

5. Tests d'intégration

Décrire les tests (ainsi que leur enchaînement) permettant de vérifier que les différents modules, paquetages, … de l'application s'interfacent correctement. On distinguera les interfaces internes des interfaces externes à l'application.

Pour chaque test d'intégration :

5.1. Identification

//Donner un identifiant unique à chaque test d'intégration

- **Identifiant –** *TI01*

  - Scénario **:** Vérification de l'intégration du module d'importation avec le module de traitement des fichiers.

- **Identifiant –** *TI02*

- **Scénario :** Vérification de l'intégration du module d'analyse des données avec l'interface utilisateur.

- **Identifiant :** *TI03*

Scénario : Vérification de l'intégration entre le module de gestion des erreurs et l'interface d'affichage des messages d'erreur.

- **Identifiant :** *TI04*

Scénario : Vérification de l'intégration entre le module de génération du rapport et le système de téléchargement.

- **Identifiant :** *TI05*

Scénario : Vérification de l'intégration entre le module de visualisation des statistiques et le module de calcul des données statistiques.

- **Identifiant :** *TI06*

Scénario : Vérification de l'intégration du module d'importation avec la gestion des fichiers multiples.

5.2. Description

Décrire le but du test, les caractéristiques de l'environnement de test et le principe de réalisation du test.

5.3. Contraintes

Décrire les contraintes liées à ce test : environnement de test particulier, installation particulière, intervention humaine spécifique, ... etc.

Indiquer les éléments de documentation faisant référence (spécifications, éléments de conception, etc …).

Même environnement général :

- OS compatible : Windows 10/11, Linux, macOS.
- Qt 6.8.1 + modules requis (pugixml…).
- L'utilisateur doit être familier avec les types de documents à importer

5.4. Dépendances

Lister et expliciter les tests à mener préalablement à la réalisation du test.

5.5. Procédure de test

Décrire les données en entrée, les résultats attendus, et les critères de validation

6. Tests unitaires

L'ensemble des tests unitaires de notre projet est réalisé à l'aide du framework Qt Test, intégré à Qt Creator. Qt Test fournit un ensemble de macros et de fonctions permettant d'écrire, et d'exécuter des tests unitaires en C++. Chaque test permet de vérifier qu'une méthode ou une classe se comportent correctement, en validant que les conditions attendues sont respectées. Tous les tests sont regroupés dans un dossier dédié du projet "tests", et organisés par classe.

Pour chaque test unitaire → Voir Annexe – Identification.

7. Vérification de la documentation

Ce test vise à valider la cohérence entre la documentation fournie (manuel d'utilisation, manuel d'installation, cahier des charges, etc.) et le comportement réel de l'application TraceX.

Objectifs :

- S'assurer que toutes les fonctionnalités décrites dans la documentation sont bien présentes et fonctionnelles dans l'application.
- Vérifier que les instructions de l'utilisateur pour l'installation, l'importation, la navigation dans l'interface, l'interprétation des résultats et l'exportation de rapports sont exactes.
- Identifier les écarts ou les manques éventuels dans la documentation.

Méthodologie :

- Suivre étape par étape les procédures indiquées dans les manuels.
- Comparer les résultats obtenus avec ceux attendus.
- Noter toute incohérence, information manquante ou instruction incorrecte.

Critère de validation : La documentation est validée si aucune étape décrite n'échoue, si toutes les instructions sont compréhensibles et fidèles au comportement réel du logiciel.

8. Glossaire

- **SSS (Spécification des Besoins du Système)** : Document décrivant les besoins globaux du système, exprimés de manière compréhensible pour un utilisateur. C'est le point de départ du projet.
- **SRS (Spécification des Exigences Logicielles)** : Document plus technique que le SSS. Il traduit les besoins du SSS en exigences logicielles précises à développer.
- **SDD (Spécification de la Conception Détaillée)** : Document décrivant comment chaque exigence sera techniquement mise en œuvre dans le code. Il s'adresse surtout aux développeurs.
- **Traçabilité :** Capacité à suivre un lien entre une exigence d'un niveau supérieur et sa déclinaison dans un niveau inférieur.
- **Taux de traçabilité :** Pourcentage de liens établis entre deux niveaux d'exigences (ex. : entre SRS et SDD).
- **Graphe de traçabilité :** Représentation graphique des exigences et de leurs liens. Chaque nœud représente une exigence, chaque arête un lien.
- **Test fonctionnel :** Test vérifiant qu'une fonctionnalité répond bien à l'exigence métier spécifiée.
- **Test d'intégration :** Test vérifiant que plusieurs modules interagissent correctement ensemble.
- **Test unitaire :** Test vérifiant individuellement le comportement d'une fonction ou d'une méthode.
- **Qt Test** : Framework de test unitaire intégré à Qt Creator

9. Références

- [R1] Cahier des charges (définit les objectifs, les exigences fonctionnelles et non fonctionnelles).

- [R2] Spécification technique des formats SSS, SRS et SDD (décrit la structure des fichiers XML à charger).
- [R3] Diagrammes de classes / Conception UML du projet (utile pour identifier les modules à tester).
- [R4] Documentation utilisateur de TraceX (sert à tester que les fonctionnalités correspondent bien à ce qui est prévu pour l'utilisateur final).
- [R5] Norme de nommage et conventions de codage utilisées (utile pour la cohérence des tests et du code).
- [R6] Plan de développement initial (aide à suivre les étapes de mise en œuvre prévues pour évaluer les tests associés à chaque phase).

10. Index

Annexe – Identification

ID	Nom de la méthode/Constructeur	Description	Résultat attendu	Statut
TU_SDD_01	addExigence()	Ajoute une exigence à un document SDD.	L'exigence est ajoutée au document.	réussite
TU_SDD_02	getExigence()	Récupère la liste des exigences d'un document SDD.	La liste d'exigences du document est retournée correctement.	réussite
TU_SDD_03	totalExigences()	Compte le nombre total d'exigences dans un document SDD.	Le nombre correct d'exigences est retourné.	réussite
TU_SDD_04	toQStringExigence()	Transforme les exigences du SDD en texte lisible.	Un texte contenant les IDs des exigences est généré.	réussite
TU_SDD_05	getIdsExigence()	Récupère tous les identifiants des exigences sans filtre.	Tous les IDs d'exigences sont récupérés correctement.	réussite
TU_SRS_01	addExigence()	Ajoute une exigence à un document SRS.	L'exigence est ajoutée avec succès au document.	réussite
TU_SRS_02	getExigence()	Obtient les exigences ajoutées au SRS.	Les exigences du document SRS sont retournées correctement.	réussite
TU_SRS_03	totalExigences()	Compte les exigences	Le total d'exigences dans le SRS est	réussite

		présentes dans le document SRS.	précis.	
TU_SRS_04	setFils()	Définit un document SDD en tant que fils du SRS.	Le document SDD est correctement associé au SRS.	réussite
TU_SRS_05	getFils()	Récupère le document SDD fils associé au SRS.	Le SDD fils du SRS est correctement récupéré.	réussite
TU_SRS_06	getIdsExigenceMatchesCriteres()	Retourne les IDs d'exigences correspondant à des critères donnés.	Les IDs correspondant exactement aux critères fournis sont retournés.	réussite
TU_SRS_07	toQStringExigence()	Transforme les exigences du SRS en texte lisible.	Texte contenant les exigences du SRS clairement affiché.	réussite
TU_SSS_01	addExigence()	Ajoute une exigence à un document SSS.	Exigence ajoutée avec succès au document SSS.	réussite
TU_SSS_02	getExigence()	Récupère les exigences d'un document SSS.	Les exigences du document SSS sont retournées correctement.	réussite
TU_SSS_03	totalExigences()	Compte le nombre d'exigences dans le SSS.	Retourne le nombre exact d'exigences.	réussite
TU_SSS_04	addFils()	Ajoute un document SRS en tant que fils du SSS.	Document SRS ajouté correctement en tant que fils.	réussite
TU_SSS_05	getFils()	Récupère les	Documents fils récupérés avec succès.	réussite

		documents SRS fils d'un SSS.		
TU_SSS_06	getIdsExigence()	Récupère tous les IDs des exigences d'un SSS.	Tous les IDs d'exigences récupérés précisément.	réussite
TU_SSS_07	toQStringExigence()	Convertit les exigences du SSS en texte.	Texte lisible contenant les exigences du SSS.	réussite
TU_STYLE_01	Style_Button_analyse()	Retourne le style CSS du bouton d'analyse.	Style CSS du bouton d'analyse retourné correctement.	réussite
TU_STYLE_02	Changer_bleu_bouton_nav()	Change la couleur du bouton de navigation en bleu.	Couleur du bouton de navigation correctement changée en bleu.	réussite
TU_STYLE_03	Style_liste_Fichiers()	Retourne le style CSS pour la liste des fichiers.	Style CSS pour la liste des fichiers retournés correctement.	réussite
TU_STYLE_04	Style_arbre_Fichiers()	Retourne le style CSS pour l'arbre des fichiers.	Style CSS pour l'arbre des fichiers correctement retourné.	réussite
TU_STYLE_05	Style_progressBar()	Retourne le style CSS des barres de progression.	Style CSS des barres de progression retourné correctement.	réussite
TU_STYLE_06	Style_CheckBox_Critere_Style()	Retourne le style CSS pour les cases à cocher.	Style CSS pour cases à cocher retourné correctement.	réussite
TU_FILE_01	constructeurParDefaut()	Construit un objet File sans chemin spécifique.	Objet créé avec un chemin vide.	réussite

TU_FILE_02	constructeurAvecChemin()	Construit un objet File avec un chemin spécifié.	Objet créé avec le chemin spécifié.	réussite
TU_FILE_03	getNom()	Retourne le nom du fichier extrait du chemin complet.	Nom du fichier correctement extrait.	réussite
TU_FILE_04	extensions()	Vérifie les extensions supportées (format Word, Excel et .csv).	Extensions correctement identifiées.	réussite
TU_FILE_05	setRadical() / getRadical()	Définit et récupère le radical d'un fichier.	Radical du fichier correctement défini et récupéré.	réussite
TU_FILEUTILS_01	testCreateCopy()	Crée une copie d'un fichier existant.	Une copie exacte du fichier source est créée.	réussite
TU_FILEUTILS_02	testCreateZipFolder()	Créer une archive ZIP d'un dossier spécifié.	Une archive ZIP valide du dossier est créée.	réussite
TU_FILEUTILS_03	testExtractXmlFolder()	Extrait le contenu XML d'un fichier DOCX.	Le contenu XML est correctement extrait et disponible.	réussite
TU_FILEUTILS_04	testDeleteExtra()	Supprime les fichiers et dossiers temporaires créés.	Tous les fichiers et dossiers temporaires sont correctement supprimés.	réussite
TU_RAPPORT_01	generation()	Génère un rapport détaillé à partir d'une liste de documents SSS.	Un rapport complet et précis est généré à partir des documents SSS.	réussite
TU_RAPPORT_02	getLignesRapport()	Récupère les lignes	Toutes les lignes du rapport sont	réussite

		générées dans le rapport.	correctement récupérées.	
TU_RAPPORT_03	isInSrs()	Vérifie si une exigence spécifique est incluse dans un document SRS donné.	Retourne VRAI si l'exigence est présente dans le document SRS.	réussite
TU_RAPPORT_04	initTestCase()	Initialise les objets nécessaires avant les tests du rapport.	Les objets nécessaires aux tests sont correctement initialisés.	réussite
TU_RAPPORT_05	cleanupTestCase()	Nettoie et libère la mémoire après les tests.	Tous les objets créés sont correctement nettoyés et la mémoire libérée.	réussite
TU_STYLEFILE_01	constructeurParDefaut()	Crée un objet StyleFile avec des valeurs par défaut.	Tous les attributs sont initialisés à des valeurs neutres ou par défaut.	réussite
TU_STYLEFILE_02	constructeurCompletEtAccesseurs()	Crée un objet StyleFile avec tous les attributs renseignés.	Les attributs de style sont correctement assignés.	réussite
TU_STYLEFILE_03	toQString()	Convertit un objet StyleFile en représentation textuelle Qt.	La chaîne contient tous les attributs visibles (police, couleur, style…).	réussite
TU_TRACABILITE_01	taux_traca_sss_srs()	Calcule le taux de traçabilité entre exigences SSS et SRS.	Le taux est un pourcentage représentant le lien entre les exigences.	réussite

TU_TRACABILITE_02	taux_traca_srs_sdd()	Calcule le taux de traçabilité entre exigences SRS et SDD.	Le taux est correctement évalué selon les recouvrements entre les niveaux.	réussite
TU_TRACABILITE_03	taux_traca()	Calcule un taux de recouvrement entre deux listes d'identifiants.	Retourne le pourcentage de correspondance	réussite
TU_XMLPARSERUTILS_01	nettoyage_exigence_srs()	Nettoie une ligne de texte et en extrait les identifiants d'exigences.	Une liste propre d'identifiants est extraite de la chaîne de texte.	réussite
TU_XMLPARSERUTILS_02	RGB_to_Color()	Convertit une couleur hexadécimale en nom ou code couleur interprétable.	Retourne la bonne couleur	réussite