Équipe 1

Reconnaissance automatique des matricules et compilation des notes pour optimiser le travail des enseignants
Plan de tests logiciels

Version 3.0

Historique des révisions

Date	Version	Description	Auteur
2022-10-06	1.0	Rédaction initiale du plan de tests	Équipe 1
2022-11-30	2.0	Mise à jour du plan de tests pour la remise finale	Équipe 1
2022-12-05	3.0	Nouvelle mise à jour du plan de tests pour la remise finale	Équipe 1

Table des matières

Plan de tests logiciels

1. Introduction

Ce document de test a pour objectif d'évaluer le bon fonctionnement de notre application, *RMN*. D'abord, nous allons déterminer les exigences à tester, ensuite nous allons décrire la stratégie qui va être appliquée, en s'assurant de spécifier les ressources utilisées. Enfin, nous allons conclure avec les principaux jalons relatifs à la discipline des tests tout au long de la réalisation du projet.

2. Exigences à tester

Exigences	**Tests associés**
Fonctionnelles	
Page de connexion	Tests de fonction, Tests d'interface usager, Tests d'intégrité des données, Tests de sécurité et de contrôle d'accès
Menu principal	Tests de fonction, Tests d'interface usager, Tests de configuration
Page de création d'une nouvelle tâche (importation des fichiers)	Tests de fonction, Tests d'interface usager, Tests de charge, Tests de stress, Tests de performance, Tests de configuration
Page de l'historique des tâches	Tests de fonction, Tests d'interface usager, Tests d'intégrité des données, Tests de charge, Tests de volume, Tests de performance, Tests de configuration
Exportation des copies/fichiers	Tests de fonction, Tests d'interface usager, Tests de performance, Tests de configuration
Configuration d'un nouveau gabarit	Tests de fonction, Tests d'interface usager, Tests d'intégrité des données,

	Tests de configuration
Modification d'un gabarit existant	Tests de fonction, Tests d'interface usager, Tests d'intégrité des données, Tests de configuration
Page de vérification manuelle du résultat d'une tâche finie	Tests de fonction, Tests d'interface usager, Tests de configuration
Reconnaissance automatique des notes	Tests de fonction, Tests de performance
Non-fonctionnelles	
Utilisabilité	Tests d'interface usager, Tests d'échec/récupération
Fiabilité	Tests d'intégrité des données, Tests de sécurité et de contrôle d'accès, Tests d'échec/récupération
Performance	Tests de performance, Tests de charge, Tests de stress, Tests de volume
Maintenabilité	Tests de configuration
Contraintes de conception	Tests de configuration
Sécurité	Tests d'intégrité des données, Tests de sécurité et de contrôle d'accès, Tests d'échec/récupération

3. Stratégie de test

3.1. Types de test

3.1.1. Tests de fonction

Objectif de test:	Exécuter les différents scénarios de la fonctionnalité décrits par les diagrammes de cas d'utilisation et s'assurer que son résultat reflète ce qui est attendu dans les requis du SRS et des vues de cas d'utilisation.
Technique:	Choisir un scénario spécifique d'une fonctionnalité, exécuter le scénario et vérifier que le résultat est celui attendu en le comparant au requis du SRS. Répéter la procédure jusqu'à ce que tous les scénarios de la fonctionnalité soient couverts.
Critère de complétion:	Le scénario du fonctionnement exécuté est conforme au requis du SRS.
Considérations spéciales:	

3.1.2. Tests d'interface usager

Objectif de test:	Tester les différents boutons, modales, et autres composantes de l'interface avec lesquelles l'utilisateur peut interagir et s'assurer que l'interface réponde correctement aux interactions de l'utilisateur. S'assurer également que l'utilisation est fluide et intuitive.
Technique:	Naviguer à travers l'application en essayant les différentes fonctionnalités implémentées. Vérifier que les différents boutons, modales, et autres composantes avec lesquels l'utilisateur peut interagir agissent comme indiqué dans le document de requis SRS et les diagrammes de cas d'utilisation.
Critère de complétion:	Les différentes fonctionnalités de l'interface usager fonctionnent correctement et permettent à l'utilisateur de naviguer l'application sans avoir de problèmes d'accès à un service.
Considérations spéciales:	

3.1.3. Tests d'intégrité des données

Objectif de test:	S'assurer que le serveur n'accepte que les données permises et que les données stockées en base de données ne sont pas corrompues et sont du bon type.
Technique:	Faire des requêtes HTTPS contenant des données valides et invalides au serveur à l'aide de l'outil Postman. Valider que le serveur gère les données invalides et qu'aucune donnée corrompue ne se retrouve dans la base de données.

Critère de complétion:	Les requêtes invalides retournent une erreur, et ne sont pas sauvegardées en base de données.
Considérations spéciales:	

3.1.4. Tests de performance

Objectif de test:	Mesurer les temps de réponse et les taux de transactions pour les requêtes entre le serveur et le client, entre le serveur et la base de données MongoDB et entre le serveur et Redis.
Technique:	Tester et mesurer les latences pour les fonctionnalités qui nécessitent des communications entre le serveur et le client, entre le serveur et la base de données MongoDB, entre le serveur et la base de données FireBase et entre le serveur et Redis.
Critère de complétion:	Les mesures de latences obtenues doivent respecter les limites décrites dans le document de requis SRS.
Considérations spéciales:	

3.1.5. Tests de charge

Objectif de test:	S'assurer du bon fonctionnement de l'application quand un grand nombre de personnes en font usage en même temps (*scalability)*. S'assurer que le bon fonctionnement de l'application est maintenu pour les limites de charges de travail précisées dans le document de requis SRS.
Technique:	Reproduire les scénarios de cas d'utilisation limite décrits dans le document de requis SRS. Vérifier que les scénarios limites sont traités de façon raisonnable comme décrite dans le document de requis SRS.
Critère de complétion:	Il n'y a pas de latence significative entre les actions et l'application n'échoue pas. Le bon fonctionnement est garanti selon les limites de charges de travail établies dans le document de requis SRS.
Considérations spéciales:	

3.1.6. Tests de stress

Objectif de test:	S'assurer que les différentes fonctionnalités qui accèdent à une même ressource partagée soient gérées de façon adéquate telles que: la concurrence lorsque deux ou plusieurs utilisateurs créent des nouvelles tâches en même temps, configurent/modifient des gabarits en même temps,etc.

	S’assurer que lorsqu’un utilisateur tente de répéter une action plusieurs fois à répétitions pendant des intervalles de temps très minime, l’application continue son bon fonctionnement et gère les erreurs de concurrence en affichant un message d’erreur.
Technique:	Tester les fonctionnalités qui accèdent à une même ressource à l’aide de deux utilisateurs en parallèle. S’assurer par la suite que l’ordre des exécutions a bien été respecté.
Critère de complétion:	S’assurer que les requêtes envoyées au serveur suivent l’ordre de réception. Par exemple, si un utilisateur 1 crée une nouvelle tâche en même temps qu’un utilisateur 2, mais qu’il est quelques millisecondes en avance. La tâche de l’utilisateur 1 doit être gérée avant la tâche de l’utilisateur 2.
Considérations spéciales:	

3.1.7. Tests de volume

Objectif de test:	Vérifier que charger et envoyer une grosse quantité de données ne va pas affecter la performance du logiciel et nuire à l’expérience utilisateur.
Technique:	Tester toutes les fonctionnalités qui n’ont pas de limites définies. Il y a, par exemple, le nombre de tâches à rouler dans la page de tâches et même la taille maximale des fichiers que le programme peut prendre. Chacune de ces fonctionnalités sera testée en ajoutant une grande quantité de données afin de s’assurer que le bon fonctionnement est maintenu.
Critère de complétion:	L’application ne devrait pas avoir un temps de réponse plus grand que 500ms et n’échoue pas soudainement à cause des grosses quantités de données (grand volume de données). Le bon fonctionnement doit être maintenu de façon acceptable.
Considérations spéciales:	

3.1.8. Tests d’échec/récupération

Objectif de test:	Vérifier que l’application est capable de récupérer d’une défaillance matérielle, logicielle ou réseau et s’assurer que ces défaillances ne compromettent pas l’intégrité des données dans la base de données.

Technique:	Tester pour des scénarios de défaillance matérielle et logicielle et s'assurer que le serveur puisse détecter cela et agir en conséquence. Deux scénarios principaux à tester: 1- L'ordinateur du client se ferme ou perd sa connexion à l'Internet soudainement. 2- Une tâche échoue durant l'exécution de son traitement par le module et ne retourne pas de réponse.
Critère de complétion:	1- Le serveur déconnecte l'utilisateur de l'application et l'application du client lourd retourne à la page de connexion. 2- La tâche va être remise dans la file de tâches (queue) afin d'être réexécutée.
Considérations spéciales:	

3.1.9. Tests de configuration

Objectif de test:	Vérifier que le client web doit pouvoir rouler sur n'importe navigateur courant. S'assurer que des librairies ou des méthodes de définition d'interface usager qui vont être utilisées durant le projet se transposent bien à travers les différents navigateurs.
Technique:	Rouleur le client sur les navigateurs suivants: - Google Chrome (Version 87.0.4280 au minimum) -Firefox (Version Firefox 83 au minimum) -Microsoft Edge (Version Edge 88 au minimum)
Critère de complétion:	Aucune fonctionnalité n'est affectée par le changement de navigateur et l'interface usager ainsi que l'expérience utilisateur (UX) ne sont pas affectées.
Considérations spéciales:	

3.2. Outils

Les outils suivants seront utilisés au sein de la discipline de test:

Type de test	Outil
Tests de fonction	Client web
Tests d'interface usager	Client web

Tests d'intégrité des données	Postman, MongoDB, Firebase
Tests de performance	Client web, MongoDB, Redis, Kubernetes, Keda Monitoring (*ScaledObject*)
Tests de charge	Plusieurs utilisateurs ayant chacun le client web
Tests de stress	Plusieurs utilisateurs ayant chacun le client web, Redis, Kubernetes
Tests de volume	Postman, MongoDB, Redis, Firebase
Tests d'échec/récupération	Postman, MongoDB, Redis, Keda Monitoring (*ScaledObject*)
Tests de configuration	Client web

4. Ressources

4.1. Équipe de test

Rôle	Membre de l'équipe	Responsabilités
Concepteur des tests client web	Aleksandar Stijelja	1- Conceptualiser les cas de tests pour le client web
Concepteur des tests serveur	Jason Thai	1- Conceptualiser des cas de tests pour le serveur
Testeur client web	Anis Zouatene	1- Exécuter les cas de tests pour le client web 2- Noter les résultats des cas de tests dans le document de résultats de tests
Testeur serveur	Andy Lam	1- Exécuter les cas de tests pour le serveur 2- Noter les résultats des cas de tests dans le document de résultats de tests

Évaluateur client web et serveur	Mohammed Ariful Islam	1- Évaluer les résultats des cas de tests du client web 2- Évaluer les résultats des cas de tests du serveur

4.2. Système

L'environnement et la configuration requis pour la discipline de test comprennent toutes les composantes retrouvées dans l'environnement de déploiement. C'est-à-dire qu'il faut au moins une instance pour les composantes suivantes dans Kubernetes: une instance du serveur API, une instance de l'application web Angular, une instance de la fille d'exécution Redis, et le module Keda qui gère le déploiement automatique d'instances d'exécution. Ces différentes instances seront conteneurisées dans Minikube. Bref, le système de tests nécessitera l'environnement de Minikube d'installé ainsi que toutes les images Docker des différentes composantes (l'image Docker pour l'application web, l'image Docker pour l'exécution du module de reconnaissance, et l'image Docker pour le serveur API). Ensuite, il faudra lancer les services Kubernetes à l'aide des fichiers de déploiement pour chaque module. Finalement, il sera nécessaire dans le système de tests d'avoir un fureteur tel que Google Chrome ou FireFox pour exécuter les tests.

5. Jalons du projet

Jalon	Effort (heure-personne)	Date de début	Date de fin
Conceptions des cas de tests	15	2022-10-04	2022-10-07
Implémentation des cas de tests	20	2022-10-07	2022-10-10
Exécution des cas de tests pré-bêta	15	2022-11-14	2022-11-21
Évaluation des cas de tests pré-bêta	10	2022-11-21	2022-11-22
Exécution des cas de tests avant remise finale	15	2022-12-01	2022-12-05
Évaluation des cas de tests avant remise finale	10	2022-12-05	2022-12-06

<