PROJET DE PLATEFORME DE STREAMING INTERACTIF

-

PLAN DE TESTS

PLAN DE TESTS

GIBBERISH

TABLE DES MATIÈRES

PLAN DE TESTS

GIBBERISH

PLAN DE TESTS

GIBBERISH

HISTORIQUE

Date	Version	Commentaires
Dec 2022	1.0	Version initiale

INTRODUCTION

Ce plan de test concerne le projet de création d'une nouvelle plateforme de streaming interactif.

Il s'agit de tester l'ensemble des fonctionnalités initialement prévues dans le projet tant en termes fonctionnels que techniques.

OBJECTIFS

L'objectif de ce plan de test est de définir les tâches et responsabilités nécessaires à la réussite du projet de nouvelle plateforme de streaming interactif de Gibberish.

Les tâches permettant d'atteindre cet objectif sont :

- définir la stratégie de tests
- lister les fonctionnalités et composants à tester
- identifier les parties prenantes et leurs rôles/responsabilités
- Identifier les risques inhérents et les hypothèses de prévention
- Définir la procédure de suivi des corrections

PÉRIMÈTRE

Le périmètre fonctionnel de ce plan de test ne se limite pas aux nouvelles fonctionnalités. L'évolution de l'architecture de notre plateforme nécessite de tester toutes les fonctionnalités en appliquant les tests de non-régression aux fonctionnalités qui ont évoluées techniquement (mais pas fonctionnellement).

Les portails utilisateurs devront être également testés car ils sont maintenant distincts entre portail utilisateurs internes et portail utilisateurs externes.

Il n'y a donc pas de composants ou de fonctionnalités qui ne seront pas testés dans ce plan de test.

Toutes les fonctions métiers de l'entreprise sont donc concernées par ce plan de test.

STRATÉGIE DE TEST

Stratégie adoptée

La stratégie de test adoptée pour ce projet est d'utiliser la méthode du développement piloté par le comportement plus communément appelée BDD (Business Driven Development ou Behaviour Driven Development en anglais).

Cette méthode consiste à définir des comportements attendus d'un applicatif logiciel ou matériel et de les traduire en un ensemble de cas / scenarii de test. Cette méthode, comme son nom l'indique, est une méthode de développement avant tout et peut être accompagnée des principes du développement piloté par les tests (ou TDD - Tests Driven Development) qui consistent à écrire les tests avant d'écrire le code. Le TDD est l'étape suivante naturelle du BDD.

Le BDD est basé sur le langage Gherkin qui utilise des mots tels que "Étant donné", "Quand", "Alors", et éventuellement "Et", qui vont décrire le comportement de votre fonctionnalité. On retrouve souvent la formulation anglaise "Given/When/Then/And".

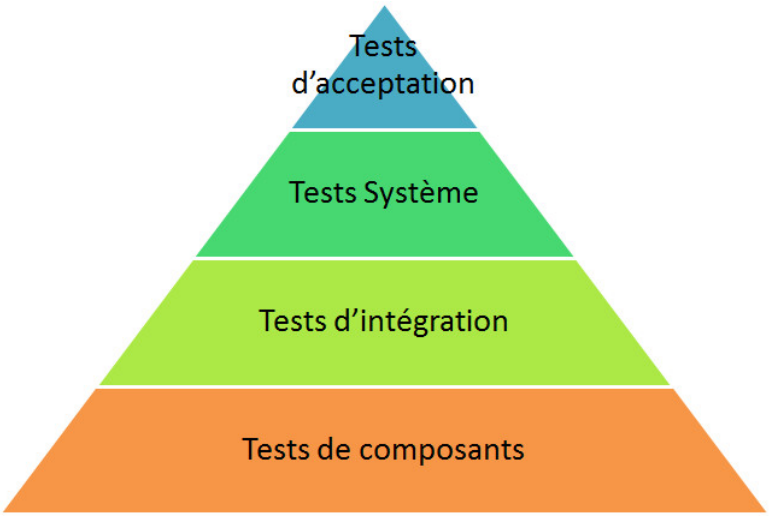
L'utilisation du BDD pour notre plan de tests s'appuiera sur l'ensemble des fonctionnalités d'interactivité décrites dans le cahier des charges du projet auxquelles viendront s'ajouter l'intégralité des fonctionnalités d'origine de l'ancienne architecture du fait de la refonte de cette architecte (détaillée dans le document de définition d'architecture du projet).

En plus de la méthode BDD, des tests de charge/performance devront être effectués pour contrôler les composants techniques assurant la scalabilité de l'architecture.

L'approche globale de ce projet d'évolution de notre plateforme se fera grâce à l'utilisation des méthodes agiles et de l'intégration continue (CI/CD).

Généralités sur l'organisation d'un plan de test

D'une manière générale, et selon l'ISTQB, on trouve 4 niveaux de test.



Les tests de composants

Les tests de composants ont pour but de tester les différents composants du logiciel séparément afin de s'assurer que chaque élément fonctionne comme spécifié. Ces tests sont aussi appelés tests unitaires ou encore alpha tests.

Ces tests sont généralement écrits et exécutés par le développeur qui a écrit le code du composant. Ces tests peuvent être manuels ou automatisés.

Les tests d'intégration

Les tests d'intégrations sont des tests effectués entre les composants afin de s'assurer du fonctionnement des interactions et de l'interface entre les différents composants.

Ces tests sont également réalisés, en général, par des développeurs. Ces tests peuvent être manuels ou automatisés.

Les tests système

Les tests système vérifient la conception, le comportement et les attentes présumées du client. Ils sont également destinés à réaliser des tests aux limites définies dans les spécifications du logiciel ou du matériel mais également par-delà ces limites

Ces tests vérifient que les différents aspects d'un système logiciel correspondent bien aux spécifications et constituent le premier niveau des tests dits « boite noire », c'est-à-dire qu'ils sont définis sur la base des spécifications du produit (règles métier, exigences de performance etc...) sans aucun accès au code source.

Ces tests sont le plus souvent réalisés de manière automatique.

Les tests d’acceptation

Les tests d'acceptation ont pour but de confirmer que le produit final correspond bien aux besoins des utilisateurs finaux.

Ces tests sont réalisés par le métier ou les utilisateurs finaux (par exemple avec un bêta test). Ces tests sont des tests manuels.

NB : Une application peut parfaitement répondre aux spécifications sans pour autant totalement répondre aux besoins des utilisateurs (spécifications non conforme aux processus métiers, problèmes d’ergonomie/cosmétiques, …).

Application des généralités à la méthodologie BDD

Intégration continue

L'intégration continue est un ensemble de pratiques utilisées en génie logiciel consistant à vérifier de manière automatique à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée.

L'approche BDD (couplée aux principes TDD) se combine parfaitement avec l'intégration continue et permet d'accélérer significativement les temps de traitement des tests et la qualité du code produit.

Les niveaux de test tels que définis par l'ISTQB couverts grâce à l'intégration continue sont :

- Les tests unitaires
- Les tests d'intégration
- Les tests système

Test d'acceptation

La méthode BDD étant particulièrement appropriée aux méthodes agiles, les tests d'acceptation seront intégrés au processus de développement (sans pour autant être automatisés).

Ces tests seront réalisés par des testeurs faisant partie des équipes de développement sur la base des critères d'acceptation inclus dans les users stories et ceci à la fin de chaque cycle de développement agile (appelé "sprint").

Les utilisateurs effectueront eux-aussi des tests d'acceptation à la fin des jalons de livraison de fonctionnalités définis en amont du projet. Ces tests s'appuieront à la fois sur les spécifications fonctionnelles et sur l'expérience des utilisateurs afin de contrôler l'utilisabilité des fonctionnalités mises en place (ergonomie, pertinence fonctionnelle, ...).

Le test sera fait manuellement par les utilisateurs concernés par les fonctionnalités métier impactées dans le sprint.

Ces tests, s'ils révèlent des résultats négatifs, devront suivre les procédures de changement de périmètre et de corrections.

Tests système

Il existe un grand nombre de tests système. Pour les besoins spécifiques aux composants et fonctionnalités nécessaires à l'architecture mise en place pour ce projet, les tests système suivants seront réalisés :

- test de charge
- Test de stress ou de résistance

Ces types de tests sont réalisés de manière automatique par un des outils préconisés

Test de charge

Le test de charge est un type de test de performance où l'application est testée pour ses performances en utilisation normale et maximale. Les performances d'une application sont vérifiées par rapport à sa réponse à la demande de l'utilisateur et sa capacité à répondre de manière cohérente dans une tolérance acceptée sur différentes charges d'utilisateurs.

Test de stress ou de résistance

Les tests de résistance sont utilisés pour trouver des moyens de mettre en défaut le système. Ils fournissent également la plage de charge maximale que le système peut supporter.

En règle générale, les tests de résistance ont une approche incrémentielle où la charge est augmentée progressivement. Le test démarre avec une charge pour laquelle l'application a déjà été testée. Ensuite, plus de charge est ajoutée lentement pour stresser le système. Le point auquel nous commençons à voir des serveurs ne répondant pas aux demandes est considéré comme le point de rupture.

Test de non-régression

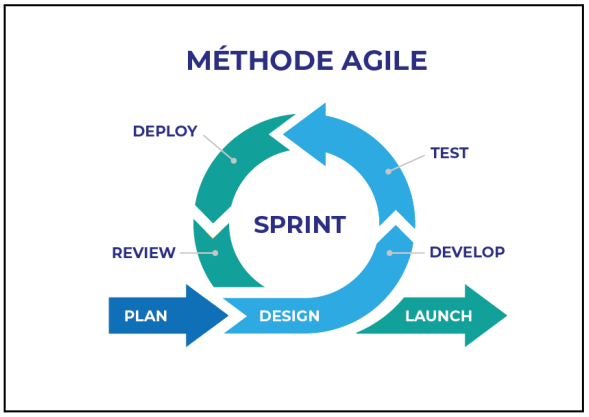
Les tests de non-régression sont à appliquer aux fonctionnalités qui étaient déjà couvertes par l'ancienne architecture et qui ont été reconduites dans la nouvelle.

Ces tests sont nécessaires car les composants techniques assurant ces fonctionnalités ont été modifiés dans la nouvelle architecture (nouveau portail, nouveaux services).

CALENDRIER DES TESTS

Comme indiqué dans le chapitre Stratégie de test, les principes agiles seront appliqués aux équipes chargées du développement et du test des composants nécessaires pour la nouvelle architecture.

Le terme "agile" définit une approche de gestion de projet qui, en impliquant les utilisateurs du début à la fin du projet et en adoptant un processus itératif et incrémental, permet de réduire le time to market des développement tout en permettant l'amélioration continue, le travail collaboratif et l'adaptation aux changements.



La planification des tests se fera donc en suivant le processus itératif de développement.

La recommandation de durée d'un sprint (sprint = une itération de développement). est comprise entre 1 et 4 semaines en fonction de la complexité des composants à développer et de la durée du projet. Il est généralement admis que la durée idéale est de 2 semaines.

PLAN DE TESTS

GIBBERISH

Concernant les tests d'acceptation réalisés par les utilisateurs (voir le chapitre Test d'acceptation), ils seront réalisés à la fin de chaque jalons de développement définis dans la roadmap du projet.

On ne peut pas seulement s'appuyer sur une approche fonctionnelle pour les livraisons des développements. Les sprints doivent donc à la fois embarquer des composants apportant une fonctionnalité mais aussi des composants assurant une fonction d'articulation au sein de l'architecture telle qu'une passerelle d'API par exemple.

L'organisation (de haut niveau) des sprints doit se faire de la manière suivante :

- Livraisons des fonctionnalités destinées aux utilisateurs internes
    - Portail interne
    - Connexion sécurisée
    - Application production de vidéos
- Livraisons des fonctionnalités destinées aux utilisateurs externes
    - Portails externes
    - Application de streaming
    - Application de production de vidéos personnalisées

On pourrait considérer qu'il y aura 6 sprints. Cependant, si on considère le nombre de composants à mettre en place et la durée maximale préconisée d'un sprint (4 semaines), l'objectif de couvrir l'intégralité des fonctionnalités à mettre en place en 24 semaines semble plutôt irréaliste. Il faudra donc découper les sprints pour qu'ils intègrent moins de fonctionnalités et réaliser ainsi plus de sprints et allonger la durée de développement. Ces informations peuvent se retrouver dans un plan d'implémentation détaillé.

COLLECTE DES DONNÉES

Test unitaires

Les cas de tests unitaire seront exécutés par les développeurs et ne généreront pas de données. Les tests en échec bloquant la validation du code produit, le code devra être modifié jusqu'à ce que celui-ci soit validé.

Tests d'acceptation

Les cas de tests d'acceptation produisent des données résultant de ces tests. Ces données sont :

- Le résultat du test
- L'analyse éventuelle du résultat (découlant du résultat)
- La classification du test (découlant du résultat)

Les résultats des tests d'acceptation seront consignés dans un outil adapté.

Les tests en échec devront être consignés dans un outil de gestion de projet en suivant la procédure de suivi des demandes de correction.

Tests système

Les résultats des tests système sont consignés dans l'outil permettant leur exécution qui généra les rapports d'exécution nécessaires à l'analyse des résultats.

PROCÉDURES DE CHANGEMENT DE PÉRIMÈTRE ET DE DEMANDES DE CORRECTION

Demande de correction

Toute demande de correction résultant d'un test devra être consignée dans l'outil de gestion de projet.

Un niveau de gravité devra être affecté à la demande de correction :

- critique : bloquant
- majeur : gênant et non bloquant
- mineur : non bloquant et non gênant

Changement de périmètre

Toute demande de changement du périmètre fonctionnel du projet devra faire l'objet d'un processus présenté ici sous une forme simplifiée :

- Consigner : Toute demande de changement devra être consignée de manière détaillée en utilisant autant que faire se peut un modèle de document ou un outil informatique pouvant structurer la demande
- Évaluer : L'évaluation doit porter sur les impacts sur le planning ou le budget du projet, les ressources nécessaires pour le traiter, les risques de sa mise en place. Elle doit se faire de manière collégiale avec le plus grand nombre de parties prenantes.
- Valider : Validation de la demande de changement en comité sur la base de l'évaluation et modification de la charte projet qui redéfinira le périmètre du projet.
- Hiérarchiser : Priorisation de la demande de changement selon son niveau d'urgence les ressources nécessaires à sa mise en place ou encore d'après les bénéfices apportés par ce changement.
- Planifier : Planification de la demande de changement à faire de manière

collégiale afin que tous les acteurs du projet aient le même niveau d'information en termes de timing du projet.

- Intégrer : Intégrer le changement dans le processus de développement.
- Contrôler : Contrôler que le changement fait bien ce qui était attendu sans effets de bord ni sur le logiciel ni sur le projet.

COMPOSANTS À TESTER

Chaque composant sera testé lors des phases de tests unitaires qui seront automatisés.

Les composants seront donc testés au fur et à mesure de leur développement à l'intérieur de chaque sprint.

La liste des composants ne peut être réalisée à cette étape du projet car elle dépend de ce que réaliseront précisément les développeurs.

FONCTIONNALITÉS À TESTER

Tests de non-régression

Produire une vidéo d'animation

Étant donné que le portail utilisateur interne permet la production de vidéos, **lorsque** l'utilisateur interne veut produire une vidéo d'animation, **alors** il doit pouvoir le faire.

Produire une vidéo commerciale

Étant donné que le portail utilisateur interne permet la production de vidéos, **lorsque** l'utilisateur interne veut produire une vidéo commerciale, **alors** il doit pouvoir le faire.

Produire une vidéo documentaire

Étant donné que le portail utilisateur interne permet la production de vidéos, **lorsque** l'utilisateur interne veut produire une vidéo documentaire, **alors** il doit pouvoir le faire.

Tests des fonctionnalités liées à la sécurité

Les tests suivants concernent les fonctionnalités qui permettent de combler la lacune “Sécurisation des accès” et qui couvrent les besoins en termes de portails utilisateurs.

Connexion sécurisée des utilisateurs internes

Étant donné que le portail utilisateur interne permet une connexion sécurisée, **lorsque** l'utilisateur interne utilise son compte et son mot de passe, **alors** il doit avoir accès au portail dédié aux utilisateurs internes.

Connexion sécurisée des utilisateurs externes

Étant donné que le portail utilisateur externe permet une connexion sécurisée, **lorsque** l'utilisateur interne utilise son compte et son mot de passe, **alors** il doit avoir accès au portail dédié aux utilisateurs externes.

Tests relatifs aux utilisateurs externes non authentifiés

Étant donné que le portail utilisateur interne n’est pas accessible aux utilisateurs externes, **lorsque** l'utilisateur externe utilise son compte et son mot de passe, **alors** il ne doit pas y avoir accès.

Étant donné que le portail utilisateur interne n’est pas accessible aux utilisateurs externes, **lorsque** l'utilisateur externe non authentifié tente d’accéder au portail interne, **alors** il ne doit pas y avoir accès.

Étant donné que le portail utilisateur externe n’est pas accessible aux utilisateurs non authentifiés, **lorsque** l'utilisateur externe non authentifié tente d’accéder au portail utilisateur externe, **alors** il ne doit pas y avoir accès.

Tests des fonctionnalités destinées aux utilisateurs internes

Les tests suivants concernent les fonctionnalités qui permettent de combler la lacune “Fonctionnalités intégrées” permettant notamment aux utilisateurs internes de produire des vidéos composites.

Produire une vidéo composite

Étant donné que le portail utilisateur interne permet la production de vidéos composites, **lorsque** l'utilisateur interne veut produire une vidéo commerciale avec de l’animation, **alors** il doit pouvoir le faire.

Étant donné que le portail utilisateur interne permet la production de vidéos composites, **lorsque** l'utilisateur interne veut produire une vidéo documentaire avec de l’animation, **alors** il doit pouvoir le faire.

Étant donné que le portail utilisateur interne permet la production de vidéos composites, **lorsque** l'utilisateur interne veut produire une vidéo d’animation avec de l’interactivité, **alors** il doit pouvoir le faire.

Étant donné que le portail utilisateur interne permet la production de vidéos composites, **lorsque** l'utilisateur interne veut produire une vidéo commerciale avec de l’interactivité, **alors** il doit pouvoir le faire.

Étant donné que le portail utilisateur interne permet la production de vidéos composites, **lorsque** l'utilisateur interne veut produire une vidéo documentaire avec de l’interactivité, **alors** il doit pouvoir le faire.

Étant donné que le portail utilisateur interne permet la production de vidéos composites, **lorsque** l'utilisateur interne veut produire une vidéo commerciale avec de l’animation et de l’interactivité, **alors** il doit pouvoir le faire.

Étant donné que le portail utilisateur interne permet la production de vidéos composites, **lorsque** l'utilisateur interne veut produire une vidéo documentaire avec de l’animation et de l’interactivité, **alors** il doit pouvoir le faire.

Tests des fonctionnalités destinées aux utilisateurs externes

Les tests suivants concernent les fonctionnalités qui couvrent les besoins des utilisateurs externes (Streaming, Interactivité et Auto-Production) et qui permettent de combler les lacunes "Plateforme publique", “Sécurisation des accès” et “Fonctionnalités intégrées”.

Montage d’une vidéo composite

Étant donné qu’il est possible de monter une vidéo composite, **lorsque** l'utilisateur externe veut monter une vidéo commerciale, **alors** il doit pouvoir le faire **et** il doit pouvoir y ajouter de l’animation 2D et 3D.

Étant donné qu’il est possible de monter une vidéo composite, **lorsque** l'utilisateur interne veut monter une vidéo documentaire, **alors** il doit pouvoir le faire **et** il doit pouvoir y ajouter de l’animation 2D et 3D.

Étant donné qu’il est possible de monter une vidéo composite, **lorsque** l'utilisateur interne veut monter une vidéo commerciale, **alors** il doit pouvoir le faire **et** il doit pouvoir y ajouter de l’interactivité.

Étant donné qu’il est possible de monter une vidéo composite, **lorsque** l'utilisateur interne veut monter une vidéo documentaire, **alors** il doit pouvoir le faire **et** il doit pouvoir y ajouter de l’interactivité.

Étant donné qu’il est possible d’ajouter de l’interactivité dans les vidéos que nous montons, **lorsque** l'utilisateur interne veut ajouter une donnée interactive de type url, **alors** il doit pouvoir le faire.

Étant donné qu’il est possible d’ajouter de l’interactivité dans les vidéos que nous montons, **lorsque** l'utilisateur interne veut monter une vidéo permettant de choisir la suite de l’histoire, **alors** il doit pouvoir le faire.

Étant donné qu’il est possible d’ajouter de l’interactivité dans les vidéos que nous montons, **lorsque** l'utilisateur interne veut monter une vidéo 360, **alors** il doit pouvoir le faire **et** il doit pouvoir changer l’angle de vue de la vidéo.

PLAN DE TESTS

GIBBERISH

Fonctionnalités d'interactivité

Téléchargements en utilisant les liens à l'intérieur de la vidéo

Étant donné qu'un lien est disponible dans dans les métadonnées d'une vidéo, **lorsque** l'utilisateur clique sur le lien, **alors** il doit se voir proposer de télécharger la vidéo.

Étant donné que plusieurs liens sont disponibles dans les métadonnées d'une vidéo, **lorsque** l'utilisateur clique sur un des liens, **alors** il doit se voir proposer de télécharger la vidéo correspondant au lien.

Modification de l'histoire en fonction des choix de l'utilisateur

Étant donné qu'un une vidéo propose de faire un choix pour avancer dans une vidéo interactive, **lorsque** l'utilisateur se voit proposer de faire un choix, **alors** il doit pouvoir choisir la suite de l'histoire en cliquant sur le bouton du choix qu'il désire.

Étant donné que l'utilisateur se voit proposer un choix dans une vidéo interactive, **lorsque** l'utilisateur fait un choix, **alors** le flux vidéo doit continuer en diffusant les éléments du flux correspondant à son choix.

Étant donné qu'il est possible d'ajouter de l'interactivité dans les vidéos que nous montons, **lorsque** l'utilisateur interne veut ajouter un choix d'histoire, **alors** il doit pouvoir le faire.

Rotation interactive tridimensionnelle (vidéo 360)

Étant donné qu'une vidéo 360 est disponible sur la plateforme de streaming, **lorsqu'**un utilisateur la visionne, **alors** il doit pouvoir changer l'angle de vue de la caméra sans arrêter la vidéo.

Étant donné qu'une vidéo 360 est disponible dans les bases de données de l'application de montage, **lorsqu'**un utilisateur interne monte une vidéo, **alors** il doit pouvoir l'ajouter au montage.

Les éléments du menu ou de la table des matières permettent, lorsque l'on clique dessus, de passer directement à un segment spécifique de la vidéo

Étant donné qu'une vidéo dispose d'un chapitrage, **lorsque** l'utilisateur clique sur un chapitre qui est après le chapitre sur lequel il est positionné actuellement, **alors** il doit directement passer au segment de la vidéo associé au chapitre sur lequel il a cliqué.

Étant donné qu'une vidéo dispose d'un chapitrage, **lorsque** l'utilisateur clique sur un chapitre qui est avant le chapitre sur lequel il est positionné actuellement, **alors** il doit directement passer au segment de la vidéo associé au chapitre sur lequel il a cliqué.

Étant donné qu'il est possible d'ajouter de l'interactivité dans les vidéos que nous montons, **lorsque** l'utilisateur interne veut chapitrer une vidéo, **alors** il doit pouvoir le faire.

Saisie des données et des commentaires des utilisateurs dans la vidéo

Étant donné qu'une vidéo propose une zone de saisie, **lorsque** l'utilisateur saisit des données/commentaires dans cette zone, **alors** ces données doivent être sauvegardées dans les métadonnées de la vidéo

Étant donné qu'une vidéo propose des zones de saisie et un chapitrage, **lorsque** l'utilisateur revient sur un chapitre qui contient des données qu'il a précédemment saisi, **alors** il doit pouvoir les modifier.

PLAN DE TESTS

GIBBERISH

Fonctionnalités d'hébergement et de streaming

Création d'une plateforme d'hébergement avec accès personnel sécurisé

Étant donné qu'un utilisateur arrive sur la page principale de la plateforme d'hébergement, **lorsque** l'utilisateur clique sur le bouton Créer un compte, **alors** il doit pouvoir créer un compte.

Étant donné qu'un utilisateur arrive sur la page principale de la plateforme d'hébergement, **lorsque** l'utilisateur saisit correctement ses identifiants puis clique sur le bouton Se connecter, **alors** il doit pouvoir se connecter.

Étant donné qu'un utilisateur arrive sur la page principale de la plateforme d'hébergement, **lorsque** l'utilisateur saisit incorrectement ses identifiants puis clique sur le bouton Se connecter, **alors** il ne doit pas pouvoir se connecter.

Hébergement et mise à disposition de contenus audio/vidéo/360/2D/3D/Interactif pour la création de vidéos personnalisées

Étant donné qu'un utilisateur a un compte sur la plateforme d'hébergement, **lorsque** l'utilisateur clique sur le bouton génère une vidéo personnalisée, **alors** il doit pouvoir utiliser les éléments audio et vidéo disponibles sur la plateforme.

Génération dynamique de vidéos personnalisées

Étant donné qu'une vidéo peut être personnalisée, **lorsque** l'utilisateur effectue des modifications dans une vidéo, **alors** il doit pouvoir sauvegarder cette vidéo.

Étant donné qu'une vidéo personnalisée est disponible sur la plateforme, **lorsqu'** un utilisateur qui n'est pas propriétaire de la vidéo la sélectionne, **alors** il ne doit pas pouvoir la modifier.

Sécurisation de parties de la vidéo avec un mot de passe

Étant donné qu'une vidéo chapitrée peut être personnalisée, **lorsque** l'utilisateur modifie une vidéo, **alors** il doit pouvoir protéger le chapitre modifié avec un mot de passe.

Étant donné qu'une vidéo peut être personnalisée, **lorsque** l'utilisateur modifie une vidéo, **alors** il doit pouvoir protéger la vidéo modifiée avec un mot de passe.

Visualiser une vidéo disponible sur la plateforme d'hébergement

Étant donné qu'un utilisateur a un compte sur la plateforme d'hébergement, **lorsque** l'utilisateur clique sur le bouton Lire d'une vidéo, **alors** il doit pouvoir visualiser cette vidéo

Étant donné qu'un utilisateur n'a pas de compte sur la plateforme d'hébergement, **lorsque** l'utilisateur clique sur le bouton Lire d'une vidéo, **alors** il doit pouvoir visualiser cette vidéo

Enregistrer une vidéo disponible sur la plateforme d'hébergement dans une liste personnelle

Étant donné qu'un utilisateur a un compte sur la plateforme d'hébergement, **lorsque** l'utilisateur clique sur le bouton Enregistrer une vidéo, **alors** un menu (pop-in) de sélection de liste doit lui être affiché.

Tests liés à la portabilité

Utilisation du navigateur Chrome

Étant donné que l'utilisateur dispose d'un PC Windows avec Chrome, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'un Mac Apple avec Chrome, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'un PC Android avec Chrome, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'une tablette Android avec Chrome, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'une tablette iOS avec Chrome, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'un téléphone portable Android avec Chrome, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'un téléphone portable iOS avec Chrome, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Utilisation du navigateur Edge

Étant donné que l'utilisateur dispose d'un PC Windows avec Edge, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'un Mac Apple avec Edge, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'un PC Android avec Edge, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'une tablette Android avec Edge, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'une tablette iOS avec Edge, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'un téléphone portable Android avec Edge, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'un téléphone portable iOS avec Edge, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Utilisation du navigateur Safari

Étant donné que l'utilisateur dispose d'un PC Windows avec Safari, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'un Mac Apple avec Safari, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'un PC Android avec Safari, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'une tablette Android avec Safari, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'une tablette iOS avec Safari, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'un téléphone portable Android avec Safari, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Étant donné que l'utilisateur dispose d'un téléphone portable iOS avec Safari, **lorsque** celui-ci accède au portail Gibberish.net, **alors** il doit pouvoir correctement visualiser le portail.

Tests système

Test de charge

Le test de charge se basera sur la capacité de notre application de streaming à supporter une charge de visualisation d'une vidéo en simultané par un nombre croissant d'utilisateurs.

Le test sera simulé pour un ordinateur (Windows, Android ou Apple), une tablette (Windows, Android ou Apple) et un smartphone (Apple ou Android).

Le test se fera sur la base d'une vidéo ayant une définition de 480p d'une durée de 60 secondes pour un nombre d'utilisateurs initial de 1000. La charge sera augmentée de manière incrémentale en multipliant le nombre d'utilisateurs par 2 à chaque incrément. Chaque incrément sera réalisé à la fin de la lecture de la vidéo (toutes les 60 secondes).

La charge maximale attendue est de 64000 utilisateurs simultanés (soit 6 incréments).

Si cette charge est atteinte sans mettre en défaut l'application, alors le test de charge sera rejoué avec :

- une vidéo de 720p pour une charge maximale attendue de 32000 (soit 5 incréments)
- une vidéo de 1080p pour une charge maximale attendue de 16000 (soit 4 incréments)

Test de stress ou de résistance

Le test de stress se basera sur la capacité de notre application de streaming à supporter l'exploitation simultanée de l'intégralité des fonctionnalités disponibles pour les utilisateurs :

- visualisation d'une vidéo
- visualisation d'une vidéo interactive
- génération d'une vidéo d'animation interactive
- génération d'une vidéo interactive

Le test sera simulé pour un ordinateur (Windows, Android ou Apple), une tablette (Windows, Android ou Apple) et un smartphone (Apple ou Android).

Le test se fera sur la base de vidéos visualisées et générées ayant une définition de 720p d'une durée de 30 secondes pour un nombre d'utilisateurs initial de 400 (100 ou ¼ du nombre total d'utilisateurs par type de vidéos). La charge sera augmentée de manière incrémentale en multipliant le nombre d'utilisateurs par 2 toutes les 15 secondes.

Le test sera réalisé jusqu'au point de rupture où un utilisateur ne pourra plus lire ou générer une vidéo (par défaillance de nos serveurs).

L'objectif à atteindre est de pouvoir supporter une charge de 25600 utilisateurs soit 6 incréments.

RESSOURCES, RÔLES ET RESPONSABILITÉS

Les rôles et responsabilités sont définies dans la matrice RACI ci-dessous.

	Parties prenantes												
Description	**Dir. technique**	**Responsable ingénierie**	**Administrateur du système**	**Architecte**	**Chef de projet**	**Product Owner**	**Scrum Master**	**Tech Lead**	**Développeurs**	**QA Lead**	**Testeurs**	**Dev Ops**	**Utilisateurs**
Rédaction des cas de tests	A	A		C	A	R	I	I		I			C
Tests unitaires	A	A		C	A	C	R	R	P	I	I		
Tests d'intégration	A	A	R	C	A	C	I	P	P	I	I	P	
Tests d'acceptation - Sprint	A	A		C	A	C	R			R	P	C	
Automatisation des tests	A	A	R	C	A	C	I	C	C			P	
Tests d'acceptation - Jalon	A	A		C	R	I	I			C	C		¨P

R	Responsible	Réalisateur
A	Accountable	Approbateur
C	Consulted	Consulté
I	Informed	Informé
P	Participates	Participe

La répartition des rôles nécessite l'embauche de salariés ou l'utilisation de prestataires pour couvrir les métiers suivants :

- Product Owner
- Scrum Master
- Tech Lead
- Développeurs
- QA Lead
- Testeurs
- DevOps

Ces métiers pouvant s'exercer à distance, les recrutements peuvent se faire partout dans le monde.

RISQUES & HYPOTHÈSES

Les risques suivants ont été identifiés pour ce projet :

Id	Risque	Type	Gravité	Probabilité	Criticité	Description	Facteur de réduction
1	Ressources insuffisantes	Organisationnel	4	3	12	Avons-nous les ressources adéquates en interne ?	Recruter en s'appuyant sur la capacité des fonctions nécessaires à l'exécution du plan de test à travailler à distance
2	Budget/Délai non tenus	Stratégique	3	3	9	Si le plan de test nécessite l'embauche de plusieurs personnes pour couvrir l'ensemble des besoins en ressources humaines, le budget risque d'être élevé et si on ne le fait pas le délai risque de s'allonger	Recruter les personnes les plus adaptées à chaque (expérience sur les fonctions métiers, les technologies employées, les postes à pourvoir, ...)
3	Environnements non adaptés	Technique	3	2	6	L'exécution du plan de test pourrait nécessiter l'achat de serveurs	Avoir recours à l'hébergement en s'appuyant sur des solutions scalables qui s'adaptent en temps réel à notre besoin
4	Fonctionnalités non adaptées	Fonctionnel	3	1	3	Les développements livrés doivent correspondre au besoin métier.	Le plan de test doit s'assurer que les cas de tests vérifient correctement les fonctions métiers décrites dans le cahier des charges

ENVIRONNEMENT ET MATÉRIEL REQUIS

- Hébergement de Jenkins sur CloudBees.
- Hébergement des environnements de tests pour les tests automatisés.
- Hébergement des environnements de recette pour les tests d'acceptation.

OUTILS

Outils pour les tests unitaires

Les outils préconisés sont :

- Selenium intégré à l'IDE pour les développeurs
- ...

Outils pour gérer les tests d'intégration

Les outils préconisés sont :

- Jenkins pour gérer les opérations DevOps

Outils pour gérer les tests système

Les outils préconisés sont :

- Jmeter, Flood.io ou LoadView pour les tests de charge (outils couplés à un outil d'intégration)

Outils pour les tests d'acceptation

Les outils préconisés sont :

- FitNesse pour les tests d'acceptation réalisés par les testeurs
- Cahier de tests pour les tests d'acceptation réalisés par les utilisateurs

Outils de gestion de projet

Le plan de test nécessite également l'utilisation d'un outil de gestion de projet afin de pouvoir suivre précisément les demandes de changements :

- JIRA
- Azure DevOps
- Asana (open source)
- Gouti (open source)

APPROBATIONS

Nom & fonction	Date	Signature
Alex Z (Dir. technique)		
Marie M (Responsable ingénierie)		
Nicolas Oger (Architecte logiciel)		
Pierre Parker (Administrateur du système)		