**CAHIER DE CHARGES : OUTIL DE TESTS AUTOMATISES CYPRESS**  
  
**1. Introduction et contexte**  
  
• **Présentation du projet** : Cypress, étant un outils de tests automatisés dédié principalement aux applications web, il offre en grandes lignes des services de tests de bout en bout, des tests d’intégration et des tests unitaires en permettant d’interagir directement avec l’interface utilisateur.  
• **Objectifs** :

• Accéder au projet cible.  
• Assurer la stabilité et la fiabilité des applications web via des tests automatisés.  
• Réduire les erreurs humaines et les coûts de maintenance en identifiant rapidement les bugs et en facilitant leur reproduction.  
• Offrir une interface simple pour l’écriture et la gestion des tests.  
  
**2. Description du besoin**  
  
• **Problématique** : Dans le développement web moderne, les cycles de déploiement sont rapides, nécessitant un retour immédiat sur la stabilité des fonctionnalités. Les méthodes manuelles de tests prennent du temps et sont souvent imprécises.  
• **Objectifs détaillés** :  
• Permettre aux développeurs d’écrire, exécuter et maintenir des tests automatisés pour améliorer la qualité du code.  
• Faciliter l’intégration de Cypress dans des environnements de développement existants pour des tests rapides et fiables.  
  
**3. Fonctionnalités et spécifications**  
  
• **Fonctionnalités principales** :  
1. **Tests de bout en bout** : Cypress doit permettre de simuler l’expérience utilisateur complète, de la navigation aux interactions complexes sur l’interface.  
2. **Tests en temps réel** : Cypress exécute les tests en temps réel, montrant instantanément les changements et erreurs détectées dans l’interface.  
3. **Rechargement automatique** : Recharger les tests en direct dès qu’une modification est faite dans le code source.  
4. **Support des assertions et des sélecteurs** : Cypress doit offrir une variété d’assertions (vérifications) et la possibilité de sélectionner des éléments par divers attributs (ex. id, class).  
• **Fonctionnalités secondaires** :  
• **Support des mocks et des stubs** : Simuler des réponses de serveur pour tester les cas où le backend ne fonctionne pas.  
• **Rapports détaillés** : Générer des rapports des tests pour faciliter l’analyse des erreurs.  
• **Intégration avec CI/CD** : Faciliter l’intégration avec des plateformes de CI/CD (intégration et déploiement continus) comme Jenkins, GitLab CI ou BitBucket dans notre cas d’utilisation.  
• **Cas d’utilisation** :  
• **Utilisateur** : Un développeur souhaite tester le processus d’achat d’un produit sur un site de e-commerce, simuler une série d’actions utilisateur et s’assurer que le flux fonctionne sans erreur.  
  
**4. Contraintes**  
  
• **Contraintes techniques** :  
• Cypress doit être compatible avec les navigateurs courants (Chrome, Firefox) et leur dernière version stable.  
• Il doit fonctionner avec les applications écrites en JavaScript, TypeScript, ou frameworks populaires (React, Angular, Vue).  
• **Contraintes de temps** :  
• Livrer une version stable dans les six mois suivant le début du développement, avec des fonctionnalités de base.  
• **Contraintes de budget** :  
• Limiter les coûts d’infrastructure pour l’intégration dans des pipelines de tests continus.  
• **Contraintes juridiques** :  
• Respecter les lois de protection des données pour les tests qui manipulent des données sensibles.  
  
**5. Livrables et qualité attendue**  
  
• **Livrables attendus** :  
• Documentation complète sur l’installation et l’utilisation de Cypress.  
• Des exemples de tests de démonstration pour aider les utilisateurs à se familiariser avec l’outil.  
• Un support de communauté et des guides pour intégrer Cypress avec des outils populaires de CI/CD.  
• **Critères de qualité** :  
• Cypress doit fournir une interface utilisateur ergonomique et facile à prendre en main pour les développeurs et testeurs.

• Le taux d’erreurs et de bugs dans l’outil Cypress doit être inférieur à 1 % pour garantir la fiabilité des tests.  
• Les tests doivent s’exécuter rapidement (en moins de 3 secondes par action sur l’interface).  
  
**6. Organisation et suivi du projet**  
  
• **Rôles et responsabilités** :  
• **Développeur** : Créer et maintenir les fonctionnalités de Cypress.  
• **Testeur** : Tester Cypress pour vérifier qu’il répond aux besoins en simulation de flux utilisateur. En ressortir le journal de tests effectif.  
• **Chef de projet** : Gérer le développement, organiser les mises à jour et vérifier la conformité du produit avec les exigences.  
• **Gestion du projet** :  
• Utiliser une méthodologie agile pour une flexibilité optimale.  
• **Modalités de suivi** :  
• Mise en place de points de suivi bi-hebdomadaires pour évaluer les progrès.  
• Tests de validation à chaque phase de développement.