**PLAN DETAILLE**

**Chapitre 1 : État de l'Art**

• 1.1 Les Tests Automatisés dans le Développement Logiciel

o Présentation des types de tests (tests unitaires, tests d’intégration, tests end-to-end)

o Importance des tests automatisés dans l’assurance qualité des logiciels

• 1.2 Présentation de Cypress

o Aperçu des fonctionnalités de Cypress : écriture de tests, exécution rapide, interface utilisateur

o Comparaison avec d’autres outils de test (Selenium, Puppeteer, etc.)

• 1.3 Avantages et Limites de Cypress

o Les points forts de Cypress pour les tests end-to-end

o Limites de Cypress, notamment pour certains types de tests ou intégrations spécifiques

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Chapitre 2 : Méthodologie et Résolution du Problème**

2.1 Analyse des Besoins et Cahier des Charges

• Identification des besoins : Quels sont les cas d'utilisation à couvrir avec Cypress ?

• Exigences du projet : Exigences fonctionnelles et techniques pour le système de tests automatisés

• Contraintes : Contraintes techniques, environnementales, et opérationnelles pour l'intégration de Cypress

2.2 Choix de l'Architecture et des Scénarios de Test

• Architecture du Système de Tests : Schéma et organisation des répertoires, modules et configurations nécessaires pour Cypress

• Définition des Scénarios de Test : Choix des scénarios en fonction des besoins de l'entreprise et des objectifs du projet

2.3 Outils et Processus de Développement

• Installation et Configuration de Cypress : Procédure d'installation, paramétrage, intégration avec le pipeline CI/CD

• Workflow de Tests : Description du processus de création, validation, et exécution des tests

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Chapitre 3 : Réalisation**

3.1 Mise en Place du Système de Tests Automatisés

• Développement des Cas de Test avec Cypress : Exemple de scripts de test, structure du code, et organisation des fichiers de test

• Paramétrage des Tests : Configuration des tests pour des environnements multiples (staging, production) et gestion des données de test

3.2 Intégration Continue et Déploiement Automatisé

• Intégration avec CI/CD : Configuration des tests dans un pipeline d'intégration continue (GitLab CI, Jenkins, etc.)

• Exécution des Tests : Automatisation de l'exécution des tests pour chaque build, et analyse des résultats

3.3 Analyse des Résultats et Optimisations

• Résultats des Tests : Analyse des succès, échecs, et détection de bugs

• Optimisations et Améliorations : Adaptations apportées en fonction des résultats observés, et propositions pour améliorer la couverture et la robustesse des tests