Cheat Sheet : Test Manuel

Types de Tests Manuels

Type de Test	Description
Test Fonctionnel	Vérifie que le système répond aux exigences fonctionnelles.
Test Non Fonctionnel	Évalue la performance, la sécurité, la convivialité, et la fiabilité.
Test Exploratoire	Test sans plan prédéfini, basé sur l'intuition et la curiosité du testeur.
Test de Régression	Vérifie qu'une modification n'affecte pas les fonctionnalités existantes.
Test de Compatibilité	Teste l'application sur différents navigateurs, appareils ou systèmes d'exploitation.
Test de Sécurité	Évalue les vulnérabilités aux attaques ou à l'accès non autorisé.
Test de Charge	Teste le comportement sous charge importante (simulations d'utilisateurs).
Test de Validation	Vérifie si le produit final correspond aux besoins du client.

Phases de Test

1. **Analyse des Exigences**
   - Identifier les critères d'acceptation.
   - Vérifier les spécifications ambiguës ou manquantes.
2. **Planification des Tests**
   - Créer un plan de test.
   - Identifier les cas de test prioritaires.
3. **Création des Cas de Test**
   - Utiliser un format clair et structuré (par exemple, Gherkin).
   - Inclure des cas positifs, négatifs, limites et exceptionnels.
4. **Exécution des Tests**
   - Suivre le plan de test.
   - Consigner les résultats et les anomalies.
5. **Rapports**
   - Décrire les anomalies clairement : **Étapes, Résultat attendu, Résultat observé**.
   - Utiliser des outils de gestion (JIRA, TestRail).

Techniques de Conception de Tests

Technique	Description
Partition d'Équivalence	Divise les données en groupes pour réduire les cas à tester.

Analyse des Valeurs Limites	Teste les valeurs aux limites (min, max, juste en dessous/dessus).
Tests Basés sur des Cas d'Utilisation	Vérifie les scénarios typiques d'utilisateur.
Tests Exploratoires	Utilise l'intuition et l'expérience pour découvrir des bugs imprévus.
Pair Testing	Deux testeurs collaborent pour analyser une fonctionnalité.

Outils de Gestion des Tests

Outil	Description
TestRail	Gestion de cas de tests, exécution et reporting.
Zephyr	Intégré à JIRA pour la gestion des tests.
Xray	Extension JIRA pour planification et gestion.
qTest	Plateforme complète pour la gestion des tests.
PractiTest	Outil de gestion de tests agile et hiérarchique.
TestLink	Open-source pour la planification des tests.

Outils de Gestion des Bugs

Outil	Description
JIRA	Outil populaire pour gérer les bugs et les tâches.
Bugzilla	Suivi des bugs, facile à utiliser.
MantisBT	Outil open-source pour le suivi des anomalies.
Redmine	Gestion de projets et suivi des bugs.
Trello	Tableaux visuels pour organiser les tâches et bugs.
Azure DevOps	Gestion des anomalies et collaboration pour DevOps.

Autres Outils Utiles

Outils pour Capturer des Données

- **Snagit** : Capture d'écran annotée pour documenter les bugs.
- **Screencast-O-Matic** : Enregistre les étapes vidéo pour reproduire un bug.
- **ShareX** : Open-source pour captures, vidéos et GIFs.

Outils de Test Exploratoire

- **Session Tester** : Planification et journalisation des sessions de test.
- **Rapid Reporter** : Simplifie la prise de notes lors des tests exploratoires.

Outils de Test de Compatibilité

- **BrowserStack** : Tester sur plusieurs navigateurs et appareils en ligne.
- **Sauce Labs** : Tester en cloud sur divers navigateurs, OS et appareils.

Techniques Avancées de Tests

Technique	Description
Tests Basés sur le Risque	Prioriser les tests en fonction de l'impact et de la probabilité d'échec.
Tests d'Accessibilité	Vérifie que l'application est utilisable pour tous, y compris les personnes ayant des handicaps.
Tests de Cas Limites	Explore les scénarios où le système pourrait se comporter de manière inattendue.
Tests de Scénarios	Simule un chemin complet de bout en bout.

Modèle pour un Cas de Test

ID	Description
Titre	[Titre du cas de test]
Préconditions	[Conditions nécessaires avant le test]
Étapes	1. [Étape 1] 2. [Étape 2]
Résultat Attendu	[Ce qui doit se produire]
Résultat Observé	[Ce qui s'est produit réellement]

Modèle pour le Signalement d'un Bug

ID du Bug	Description
Titre	[Titre du bug]
Description	[Description claire du problème]

Étapes pour Reproduire	1. [Étape 1] 2. [Étape 2]
Résultat Attendu	[Ce qui était attendu]
Résultat Observé	[Ce qui s'est produit réellement]
Priorité/Gravité	[Critique, Majeur, Mineur]
Capture d'Écran/Log	[Joindre les preuves]

Bonnes Pratiques

1. **Préparation :**
   - Comprendre les exigences et prioriser les cas critiques.
   - Préparer un environnement de test stable.
2. **Exécution :**
   - Tester d'abord les cas positifs, puis les cas négatifs.
   - Documenter précisément chaque anomalie avec des étapes reproductibles.
3. **Collaboration :**
   - Travailler étroitement avec les développeurs pour résoudre les anomalies.
   - Participer aux revues de sprint et aux démonstrations.
4. **Amélioration Continue :**
   - Réaliser des rétrospectives pour améliorer les processus de test.
   - Former l'équipe sur de nouvelles techniques et outils.