



NODE.JS JWT AUTHENTICATION TESTING

Projet : TJAA-2025

Référence : TJAA-PT-001



01/12/2025 v1.0 Équipe QA

IDENTIFICATION DU DOCUMENT

Informations Générales

Nom du document	Plan de Test - Authentification JWT Node.js
Référence du projet	TJAA-2025
Référence du document	TJAA-PT-001
Version	v 1.0
Préparé par	Oulimata SALL & Rana ROMDHANE
Date	01/12/2025

Historique des Changements

Version	ID Demande	Date	Modifié par	Description
1.0	TJAA-PT-001	01/12/2025	O. SALL & R. ROMDHANE	Création initiale du plan de test

Distribution et Validation

Méthode RACI

- **R** : Réalisation (Responsible)
- **A** : Approbation (Accountable)
- **C** : Consultation (Consulted)
- **I** : Information (Informed)

NOM Prénom	Rôle	Entité	RACI	Date validation
SALL Oulimata	Test Lead	Équipe QA	R	01/12/2025
ROMDHANE Rana	Test Engineer	Équipe QA	R	01/12/2025
AOUADI Hela	Encadrant	ENICAR	A	À valider

1	INTRODUCTION	6
1.1	Objectif	6
1.2	Références	7
2	APERÇU GÉNÉRAL DU PROJET	8
2.0.1	Fonctionnalités Principales	8
2.0.2	Périmètre des Tests	9
2.1	Jalons du Projet.	9
2.2	Jalons Clés du Projet et du Test	10
3	ÉLÉMENTS À TESTER	11
4	CARACTÉRISTIQUES À TESTER	12
4.0.1	Fonctionnalités Métier.	12
4.0.2	Aspects Techniques	12
4.0.3	Aspects Non Fonctionnels.	12
4.0.4	Infrastructure	12
5	CARACTÉRISTIQUES À NE PAS TESTER	14
6	APPROCHE TEST	15
6.1	Criticité des Caractéristiques à Tester	15
6.2	Effort de Test	16
6.3	Niveaux de Test	16
6.4	Techniques de Test	17
6.4.1	Tests Statiques	17
6.4.2	Tests Dynamiques	17
6.4.3	Techniques de Conception de Tests	18
6.5	Priorisation de l'Exécution des Tests.	18
6.5.1	Niveau 1 - Tests Unitaires	18
6.5.2	Niveau 2 - Tests d'Intégration	18
6.5.3	Niveau 3 - Tests Système / E2E	19
6.5.4	Niveau 4 - Tests Non Fonctionnels	19
	Performance	19
	Sécurité	19

- 6.6 Automatisation des Tests 20
- 6.7 Suivi et Contrôle de l’Avancement 20
 - 6.7.1 Métriques Globales du Projet 20
 - 6.7.2 Métriques par Niveau de Test 20
 - Tests Unitaires (Jest) 21
 - Tests d’Intégration 21
 - Tests E2E 21
 - Tests de Performance 21
 - 6.7.3 Fréquence de Collecte et Reporting 22
- 6.8 Gestion de Configuration 22
- 6.9 Gestion des Défauts 22
 - 6.9.1 Cycle de Vie d’un Défaut 23
 - 6.9.2 Criticité des Défauts 23
- 6.10 Outils de Test Utilisés 23
 - 6.10.1 Vue d’ensemble des Outils 23
 - 6.10.2 Détail des Outils 24
 - 6.10.3 Traçabilité Complète 24
- 6.11 Gestion de la Qualité 24
 - 6.11.1 Audits Qualité Planifiés 25
- 7 BESOINS EN ENVIRONNEMENTS 26
- 8 BESOINS EN RESSOURCES ET FORMATION 27
 - 8.1 Ressources Humaines 27
 - 8.2 Ressources Matérielles 27
 - 8.3 Besoins en Formation 28
- 9 TÂCHES DE TEST ET RESPONSABILITÉS 29
 - 9.1 Rôles et Responsabilités 29
 - 9.2 Préparation des Tests 30
 - 9.3 Exécution des Tests 30
- 10 CRITÈRES D’ARRÊT ET CONDITIONS DE REPRISE DES TESTS 31
 - 10.1 Critères d’Arrêt (Suspension des Tests) 31
 - 10.2 Conditions de Reprise des Tests 31
- 11 LIVRABLES DU TEST 33
 - 11.1 Cas de Test 33

11.2	Scripts d'Automatisation (Jest / Supertest)	34
11.3	Rapports de Couverture	34
11.4	Rapports de Tests E2E.	35
11.5	Rapport de Performance.	35
11.6	Rapport d'Anomalies	36
11.7	Rapport Final de Test (Rapport de Clôture)	36
11.8	Données de Test	37
11.9	Outils de Support	38
12	RISQUES ET CONTINGENCES	39
12.1	Tableau des Risques	39
13	CRITÈRES DE PASSAGE OU ÉCHEC	40
13.1	Critères de Succès (GO)	40
13.2	Critères d'Échec (NO-GO)	41
13.3	Critères d'Acceptation Conditionnelle	41
13.4	Processus de Validation Finale	42
14	INFORMATIONS COMPLÉMENTAIRES	43
14.1	Contexte Académique	43
14.2	Contraintes Spécifiques.	43
14.3	Spécificités Techniques	44
14.3.1	Architecture du Projet	44
14.3.2	Technologies Utilisées	44
14.4	Points d'Attention Particuliers	44
14.5	Livrables Finaux du Projet Global	45
14.6	Critères d'Évaluation Académique.	45
14.7	Contacts et Support	45
15	GLOSSAIRE	47
	SIGNATURES ET APPROBATIONS	49

1

CHAPITRE

INTRODUCTION

1.1 Objectif

Ce Plan de Test est élaboré conformément au template **Certilog** et constitue le document officiel définissant l'ensemble des activités de tests pour le projet **Node.js JWT Authentication Testing**, développé par **Oulimata SALL** et **Rana ROMDHANE**.

✔ Objectifs Principaux

- ✔ Vérifier la conformité fonctionnelle de l'API d'authentification JWT
- ✔ Valider les aspects techniques (middleware, base de données MongoDB)
- ✔ Assurer la robustesse des aspects non fonctionnels (sécurité, performance)
- ✔ Garantir la fiabilité du pipeline CI/CD (GitHub Actions)
- ✔ Confirmer la maintenabilité et la qualité du code

Ce document sert de référence pour toutes les parties prenantes du projet et définit clairement les stratégies, ressources, livrables et critères de qualité attendus.

1.2 Références

ID	Titre du document	Lien ou emplacement	Date
REF-01	Dépôt GitHub du projet	https://github.com/RanaRomdhane/node-js-jwt-auth-testing	15/11/2025
REF-02	Campagne de tests Jira/Xray	[TJAA-35] Campagne de Test - Sprint 1	20/11/2025
REF-03	Modèle Certilog	Certilog_Plan_de_Test_1.0.1	01/10/2025
REF-04	Documentation API	README.md du repository GitHub	15/11/2025

2

CHAPITRE

APERÇU GÉNÉRAL DU PROJET

Le projet consiste à concevoir, développer, tester et automatiser une API RESTful moderne utilisant les technologies suivantes :

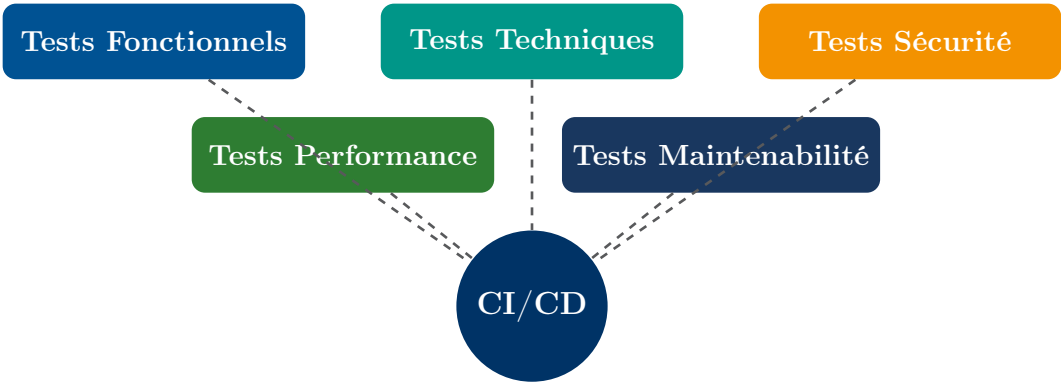


2.0.1 Fonctionnalités Principales

Fonctionnalités Métier

- Gestion complète de l'authentification utilisateur (inscription, connexion, déconnexion)
- Système de gestion des rôles et permissions (user, moderator, admin)
- Sécurisation des endpoints via JWT
- Accès conditionnel aux ressources protégées selon les rôles
- Validation robuste des données entrantes

2.0.2 Périmètre des Tests



2.1 Jalons du Projet

Jalons métiers	Date	Commentaires
Étude d'opportunité (T-1)	01/10/2025	Analyse de faisabilité
Analyse détaillée (T0)	05/10/2025	Étude du cahier des charges
Développement (T1)	15/10/2025	Mise en œuvre de l'API
Tests et validation (T1.5)	01/11/2025	Phase de tests complète
Déploiement (T2)	26/11/2025	Mise en production
Lancement sur le marché (T3)	28/11/2026	Communication externe
Clôture du projet (T4)	29/11/2026	Bilan final

2.2 Jalons Clés du Projet et du Test

Jalons clés du test	Date	Commentaires
Livraison des endpoints REST	01/10/2025	Début officiel du projet de test
Mise en place du pipeline CI/CD	17/11/2025	Fondation de l'automatisation
Création des cas de tests	20/10 - 25/10/2025	Rédaction dans Xray/Jira
Exécution des tests unitaires	01/11/2025	Validation avec Jest
Exécution des tests d'intégration	02/11/2025	Tests API avec Supertest
Exécution des tests E2E	02/11/2025	Scénarios Chai/Postman
Tests de sécurité	05/11/2025	Validation des vulnérabilités
Tests de performance	05/11/2025	Mesure des temps de réponse
Analyse des résultats	20/11/2025	Consolidation des métriques
Publication des rapports	01/12/2025	Rapports de couverture
Validation finale	15/12/2025	Acceptation du projet

3

CHAPITRE

ÉLÉMENTS À TESTER

ID	Nom	Version	Type	Commentaires
API-AUTH-01	Code source Node.js	1.0	Logiciel	Logique métier de l'API
API-AUTH-02	Middleware JWT	1.0	Logiciel	Gestion authentication
API-AUTH-03	Modèles MongoDB	1.0	Logiciel	Schémas User, Role
API-AUTH-04	Pipeline CI/CD	1.0	Logiciel	GitHub Actions
API-AUTH-05	Documentation	1.0	Document	README, API docs
API-AUTH-06	Tests automatisés	1.0	Logiciel	Jest + Supertest
API-AUTH-07	Configuration	1.0	Config	Variables d'environnement

4

CHAPITRE

CARACTÉRISTIQUES À TESTER

Les caractéristiques suivantes seront testées de manière exhaustive :

4.0.1 Fonctionnalités Métier

- ✓ Inscription utilisateur (signup) avec validation des données
- ✓ Connexion utilisateur (signin) et génération de tokens JWT
- ✓ Gestion des rôles (user, moderator, admin)
- ✓ Accès protégé aux endpoints selon les rôles
- ✓ Refresh token et renouvellement de session

4.0.2 Aspects Techniques

- ✓ Validation des données d'entrée (format, longueur, types)
- ✓ Hachage sécurisé des mots de passe (bcrypt)
- ✓ Génération et vérification des tokens JWT
- ✓ Middleware d'authentification et autorisation
- ✓ Intégration MongoDB (CRUD opérations)

4.0.3 Aspects Non Fonctionnels

- ✓ Sécurité (injections SQL/NoSQL, XSS, tokens invalides/expirés)
- ✓ Performance (temps de réponse, charge modérée)
- ✓ Maintenabilité du code (qualité, lisibilité, respect des standards)

4.0.4 Infrastructure

- ✓ Tests automatisés dans le pipeline CI/CD

- ✓ Couverture de code (statements, branches, functions)
- ✓ Analyse statique du code (ESLint, SonarQube)

CARACTÉRISTIQUES À NE PAS TESTER

Certaines fonctionnalités sont explicitement hors périmètre pour les raisons suivantes :

Caractéristique	Raison de l'exclusion
Interface front-end (UI)	Aucune interface utilisateur n'existe (API backend uniquement)
Tests de charge extrême	Stress tests > 10 000 req/s non requis pour ce projet académique
Scalabilité horizontale	Clusterisation et load balancing non prévus
Tests multi-navigateurs	Pas d'interface web à tester
Compatibilité mobile	Application backend uniquement
Intégrations tierces	Aucun service externe (paiement, email)
Tests de récupération	Backup/restore non implémentés
Tests d'accessibilité	Non applicable pour une API REST

Note importante

Ces exclusions pourront être reconsidérées dans les versions futures du projet selon l'évolution des besoins.

6

CHAPITRE

APPROCHE TEST

L'approche de test adoptée repose sur une méthodologie hybride combinant :

- Tests fonctionnels et techniques
- Tests automatisés (majoritaires) et manuels (ponctuels)
- Tests à différents niveaux (unitaires, intégration, système)
- Traçabilité complète des exigences aux résultats

i Principes Directeurs

Automatisation maximale Priorité aux tests automatisés pour garantir la répétabilité

Shift-left testing Tests précoces dès le développement








Continuous testing Intégration dans le pipeline CI/CD

Traçabilité totale Liens exigences ↔ tests ↔ défauts

Qualité mesurable Métriques objectives et KPI définis

6.1 Criticité des Caractéristiques à Tester

La criticité est évaluée selon l'impact métier et le risque technique :

Caractéristique	Criticité	Justification
Inscription / Connexion	 CRITIQUE	Fonctionnalité fondamentale bloquante
Sécurité JWT	 CRITIQUE	Faille = compromission totale
Gestion des rôles	 HAUTE	Contrôle d'accès essentiel
Validation des données	 HAUTE	Protection contre injections
Pipeline CI/CD	 HAUTE	Garantit la qualité en continu
Performance API	 MOYENNE	Important mais non bloquant
Documentation	 FAIBLE	Utile mais non critique

6.2 Effort de Test

L'effort de test est distribué stratégiquement sur les différents niveaux :

Type de test	Effort	Justification
Tests unitaires	30%	Base solide, tests nombreux et rapides
Tests d'intégration	25%	Validation des interactions critiques
Tests système / E2E	25%	Scénarios utilisateur complets
Tests de sécurité	10%	Spécifiques mais essentiels
Tests de performance	5%	Charge modérée suffisante
Vérification CI/CD	5%	Validation pipeline automatique

Total estimé : 8 jours/personne pour l'ensemble des activités de test.

6.3 Niveaux de Test

Niveau 1 - Tests Unitaires (Jest)

- **Objectif :** Valider les fonctions isolées
- **Scope :** Fonctions utilitaires, validation, hashing, génération JWT
- **Outils :** Jest avec mocks et stubs
- **Critère de succès :** 100% des fonctions critiques testées

Niveau 2 - Tests d'Intégration (Supertest + MongoDB)

- **Objectif** : Vérifier les interactions API ↔ Base de données
- **Scope** : Endpoints REST, middleware, persistance MongoDB
- **Outils** : Supertest, MongoDB Memory Server
- **Critère de succès** : Tous les endpoints testés

Niveau 3 - Tests Système / E2E (Chai / Postman)

- **Objectif** : Valider les parcours utilisateur complets
- **Scope** : Scénarios métier de bout en bout
- **Outils** : Chai, Postman/Newman
- **Critère de succès** : Tous les parcours critiques validés

Niveau 4 - Tests Non Fonctionnels

- **Objectif** : Valider sécurité et performance
- **Scope** : Vulnérabilités OWASP, temps de réponse, charge
- **Outils** : Scripts personnalisés, autocannon/loadtest
- **Critère de succès** : Conformité aux standards

Niveau 5 - Tests CI/CD (GitHub Actions)

- **Objectif** : Garantir la stabilité du pipeline
- **Scope** : Installation, lint, tests, build, déploiement
- **Outils** : GitHub Actions, rapports automatiques
- **Critère de succès** : Pipeline stable avec 95%+ de succès

6.4 Techniques de Test

6.4.1 Tests Statiques

- **ESLint** : Analyse syntaxique et respect des conventions
- **SonarQube** : Détection de code smells, bugs potentiels, vulnérabilités
- **Audit manuel** : Revue de code peer-to-peer

6.4.2 Tests Dynamiques

- **Jest** : Exécution des tests unitaires avec assertions
- **Supertest** : Tests HTTP des endpoints API
- **Chai** : Assertions avancées pour tests E2E

6.4.3 Techniques de Conception de Tests





- **Partitionnement en classes d'équivalence** : Validation des formulaires (email valide/invalid)
- **Analyse des valeurs limites** : Longueur minimale/maximale des champs
- **Tests en boîte noire** : Validation des entrées/sorties API
- **Tests en boîte blanche** : Couverture du code, branches conditionnelles

6.5 Priorisation de l'Exécution des Tests

6.5.1 Niveau 1 - Tests Unitaires

Objectif : Valider les fonctions internes de l'API.

Critères de Priorisation

-  **Complexité du code** Fonctions avec logique conditionnelle avancée
-  **Risque technique** Fonctions susceptibles de provoquer des erreurs silencieuses
-  **Impact sur composants** Fonctions réutilisées dans plusieurs modules
-  **Fréquence d'appel** Fonctions fortement sollicitées




Ordre d'exécution :

1. Fonctions de validation des données
2. Hashing et cryptographie (bcrypt, JWT)
3. Utilitaires et helpers
4. Fonctions métier simples

6.5.2 Niveau 2 - Tests d'Intégration

Objectif : Vérifier l'interaction API ↔ Base de données / middleware.

Critères de Priorisation

-  **Risque métier** Scénarios d'inscription et connexion
-  **Sévérité potentielle** Erreurs d'intégration bloquantes
Criticité des flux Login → génération JWT → accès protégé
-  **Fréquence d'utilisation** Endpoints les plus appelés

Ordre d'exécution :

1. Endpoints d'authentification (signup, signin)
2. Endpoints protégés par rôles
3. Middleware d'autorisation
4. Opérations CRUD MongoDB

5. Endpoints secondaires

6.5.3 Niveau 3 - Tests Système / E2E

Objectif : Valider le parcours utilisateur complet.

Ordre d'exécution :

1. Parcours utilisateur standard (signup → login → accès)
2. Parcours avec différents rôles (user, moderator, admin)
3. Scénarios de gestion de session (refresh, logout)
4. Scénarios d'erreur (credentials incorrects, accès refusé)

6.5.4 Niveau 4 - Tests Non Fonctionnels

Performance

Tests prioritaires :

- Temps de réponse endpoints authentication (< 200ms)
- Comportement sous charge modérée (50 req/s)
- Stabilité mémoire et CPU

Sécurité

Tests prioritaires :

- Tokens invalides/expirés/manipulés
- Tentatives d'injection NoSQL
- Brute force sur login (rate limiting)
- Accès aux endpoints sans authentication
- Élévation de privilèges (user → admin)

6.6 Automatisation des Tests

Niveau de test	Auto.	Outil principal	Fréquence
Tests unitaires	✓ 100%	Jest	Chaque commit
Tests d'intégration	✓ 100%	Supertest	Chaque commit
Tests E2E	✓ 90%	Chai / Newman	Chaque push main
Tests de sécurité	✓ 80%	Scripts custom	Quotidien
Tests de performance	✓ 70%	loadtest/autocannon	Hebdomadaire
Linting	✓ 100%	ESLint	Chaque commit
Analyse de couverture	✓ 100%	Jest coverage	Chaque push
Pipeline CI/CD	✓ 100%	GitHub Actions	Chaque push

✓ Bénéfices de l'Automatisation

- Exécution rapide et répétable
- Détection précoce des régressions
- Feedback immédiat aux développeurs
- Réduction des coûts de test manuel
- Traçabilité et historique des résultats

6.7 Suivi et Contrôle de l'Avancement

6.7.1 Métriques Globales du Projet

Métrique	Formule / Description	Objectif	Source
Nombre total de tests	Count(tests)	≥ 100	Jest + Xray
Taux global de réussite	$(\text{Réussis} / \text{Total}) \times 100$	$\geq 90\%$	CI/CD
Couverture de code	Coverage global	$\geq 80\%$	Jest
Nombre de défauts	Count(bugs)	Trend ↓	Jira
Temps pipeline CI/CD	Durée end-to-end	$< 10 \text{ min}$	GitHub
Taux d'automatisation	$(\text{Auto} / \text{Total}) \times 100$	$\geq 80\%$	Manuel

6.7.2 Métriques par Niveau de Test

Tests Unitaires (Jest)

Métrique	Objectif	Outil de collecte
Nombre de tests unitaires	≥ 50	Jest reporter
Taux de réussite unitaire	$\geq 95\%$	Jest output
Couverture statements	$\geq 85\%$	Jest coverage
Couverture branches	$\geq 80\%$	Jest coverage
Couverture functions	$\geq 90\%$	Jest coverage
Temps d'exécution total	< 30 secondes	Jest -verbose

Tests d'Intégration

Métrique	Objectif	Outil de collecte
Nombre de scénarios	≥ 30	Supertest reporter
Taux de réussite	$\geq 90\%$	CI/CD logs
Temps moyen réponse API	$< 200\text{ms}$	Supertest response time
Défaillances MongoDB	0	Error logs
Tests flaky (instables)	$< 5\%$	CI/CD history

Tests E2E

Métrique	Objectif	Outil de collecte
Scénarios E2E exécutés	≥ 15	Newman/Postman reports
Taux de réussite E2E	$\geq 85\%$	Postman/Chai output
Temps moyen parcours	< 2 secondes	Newman timing
Défaillances en chaîne	0	Analyse manuelle

Tests de Performance

Métrique	Objectif	Outil de collecte
Temps moyen de réponse (p50)	$< 150\text{ms}$	autocannon/loadtest
Temps p95	$< 300\text{ms}$	autocannon
Temps p99	$< 500\text{ms}$	autocannon
Requêtes par seconde	≥ 50 req/s	loadtest
Taux d'erreurs sous charge	$< 1\%$	loadtest errors
Utilisation CPU/Mémoire	$< 70\%$	Monitoring système

6.7.3 Fréquence de Collecte et Reporting

Type de rapport	Fréquence	Destinataires
Rapport quotidien CI/CD	Quotidien	Équipe technique
Dashboard métriques	Temps réel	Test Lead
Rapport hebdomadaire	Hebdomadaire	Management + QA
Rapport de couverture	Chaque push	Développeurs
Rapport final de clôture	Fin de projet	Toutes parties prenantes

6.8 Gestion de Configuration

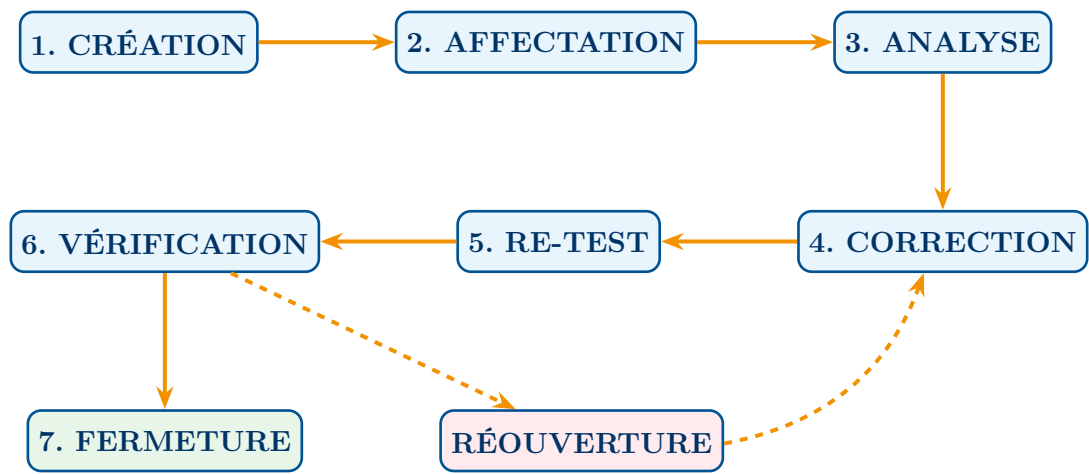
Élément	Outil	Localisation	Versionnement
Code source	Git / GitHub	Repository principal	Semantic Versioning
Tests automatisés	Git / GitHub	/tests/	Avec le code
Pipeline CI/CD	Git / GitHub	.github/workflows/	Avec le code
Cas de test Xray	Jira / Xray	Projet TJAA	Par sprint
Documentation	Git / GitHub	/docs/ et README.md	Markdown versionné
Données de test	Git / GitHub	/tests/data/	JSON versionné

i Règles de Gestion

- Toute modification du code source nécessite un commit explicite
- Les tests doivent être mis à jour en même temps que le code
- Le pipeline CI/CD est déclenché automatiquement à chaque push sur main
- Les rapports sont archivés pour chaque build
- Backup quotidien du repository GitHub

6.9 Gestion des Défauts

6.9.1 Cycle de Vie d'un Défaut

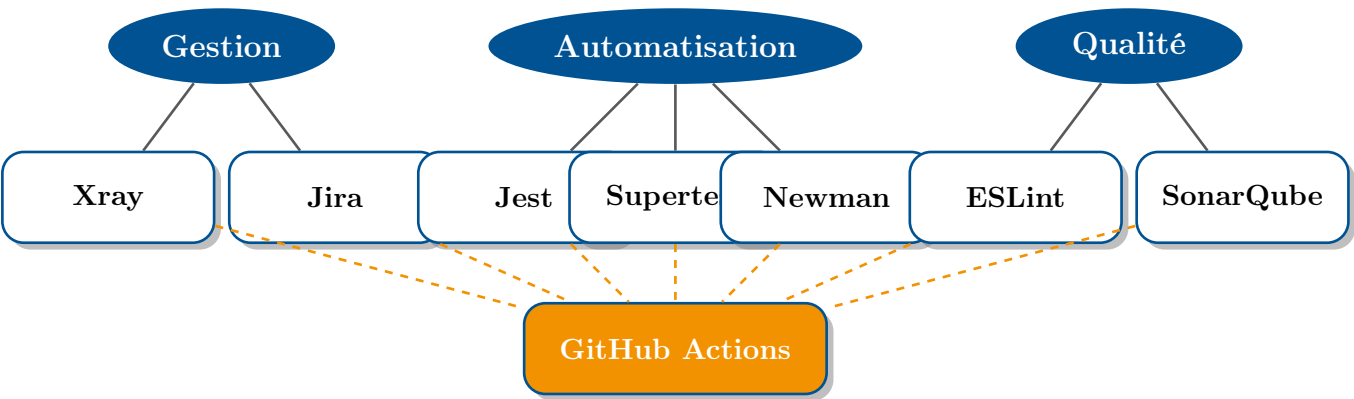


6.9.2 Criticité des Défauts

Niveau	Définition	Exemple	Délai
Bloquant	Empêche toute utilisation	Impossibilité de se connecter	< 24h
Majeur	Fonctionnalité critique défaillante	Token non validé	< 48h
Mineur	Dysfonctionnement non bloquant	Message d'erreur peu clair	< 1 semaine
Trivial	Problème cosmétique	Faute d'orthographe	Backlog

6.10 Outils de Test Utilisés

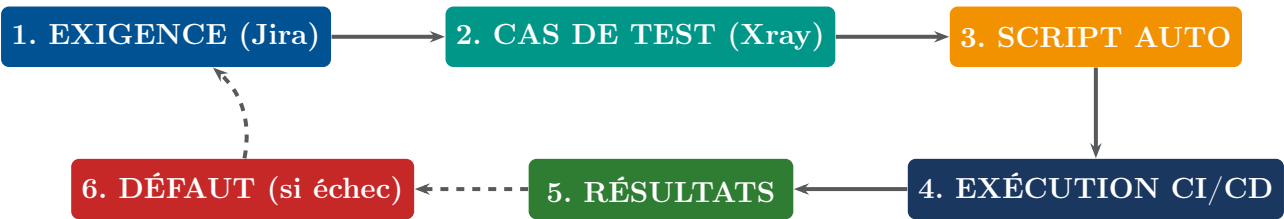
6.10.1 Vue d'ensemble des Outils



6.10.2 Détail des Outils

Outil	Type	Utilisation	Intégration
Xray	Gestion tests	Cas de test, campagnes	Jira
Jira	Gestion défauts	Suivi anomalies, workflow	Xray, GitHub
Jest	Tests unitaires	Fonctions isolées, couverture	GitHub Actions
Supertest	Tests API	Tests d'intégration HTTP	Jest
Chai	Assertions	Tests E2E avancés	Postman/Newman
Newman	Tests E2E auto	Exécution collections	GitHub Actions
ESLint	Analyse statique	Qualité code, conventions	GitHub Actions
SonarQube	Qualité code	Code smells, vulnérabilités	Optionnel

6.10.3 Traçabilité Complète



6.11 Gestion de la Qualité

i Principes de Qualité Appliqués

- Traçabilité totale** Chaque exigence a au moins un test associé
- Revue par les pairs** Tout code est revu avant merge
- Critères d'acceptation** Chaque user story a des critères mesurables
- Amélioration continue** Rétrospectives après chaque sprint
- Standards de codage** Respect des conventions ESLint

6.11.1 Audits Qualité Planifiés

Type d'audit	Fréquence	Responsable
Revue de code	Chaque Pull Request	Pair developer
Audit des tests	Hebdomadaire	Test Lead
Revue de couverture	Chaque push	Automatique (CI/CD)
Audit sécurité	Fin de sprint	Test Engineer
Validation finale	Avant déploiement	Encadrant

7

CHAPITRE

BESOINS EN ENVIRONNEMENTS

ID	Environnement	Description	Configuration
ENV-01	Local Development	Postes de travail	Node.js v18+, MongoDB, npm, Git
ENV-02	CI/CD GitHub Actions	Intégration continue	Ubuntu latest, Node.js v18
ENV-03	MongoDB Test	Base de données	MongoDB 6.0+, collections test
ENV-04	Staging (optionnel)	Pré-production	Environnement miroir

Variables d'Environnement Requises

Variable	Description	Exemple
MONGODB_URI	URL connexion MongoDB	mongodb://localhost:27017/testdb
JWT_SECRET	Clé secrète JWT	mySecretKey123
JWT_EXPIRATION	Durée validité token	86400 (24h)
PORT	Port serveur Express	3000
NODE_ENV	Environnement Node	test / production

8

CHAPITRE

BESOINS EN RESSOURCES ET FORMATION

8.1 Ressources Humaines

Rôle	Nom	Charge	Période	Compétences
Test Lead	Rana ROMDHANE	50% (4h/j)	01/10 - 15/12/2025	Jest, Xray, Jira
Test Engineer	Oulimata SALL	50% (4h/j)	01/10 - 15/12/2025	Supertest, CI/CD
Encadrant	Hela AOUADI	10% (conseil)	01/10 - 15/12/2025	Validation, revue

Total effort estimé : 8 jours/personne

8.2 Ressources Matérielles

Ressource	Quantité	Usage
Postes de travail	2	Développement et tests locaux
Accès GitHub	2 comptes	Repository et CI/CD
Licence Jira/Xray	2 utilisateurs	Gestion tests et défauts
MongoDB Atlas (opt.)	1 cluster	BD cloud pour tests

8.3 Besoins en Formation

Thème	Public	Durée	Date prévue
Xray / Jira pour les tests	O. SALL & R. ROMDHANE	2h	05/10/2025
Jest avancé	O. SALL & R. ROMDHANE	3h	10/10/2025
Sécurité API (OWASP)	R. ROMDHANE	2h	15/10/2025
GitHub Actions CI/CD	R. ROMDHANE	2h	20/10/2025
Techniques de test API	O. SALL	2h	12/10/2025

Total formation : 11 heures

9

CHAPITRE

TÂCHES DE TEST ET RESPONSABILITÉS

9.1 Rôles et Responsabilités

Rôle	NOM Prénom	Entité	Responsabilités principales
Test Lead	ROMDHANE Rana	Équipe QA	<ul style="list-style-type: none"> ➤ Planification des tests ➤ Création des cas de test ➤ Coordination avec l'encadrant ➤ Analyse des résultats
Test Engineer	SALL Oulimata	Équipe QA	<ul style="list-style-type: none"> ➤ Automatisation (Jest, Super-test) ➤ Configuration CI/CD ➤ Tests de sécurité ➤ Maintenance du pipeline
Encadrant	AOUADI Hela	ENICAR	<ul style="list-style-type: none"> ➤ Validation du plan de test ➤ Revue des livrables ➤ Approbation finale

9.2 Préparation des Tests

Description	Responsable	Charge	Date fin	Livrables
Analyse des exigences	R. ROMD-HANE	0,5 j	10/10/2025	Document d'analyse
Rédaction cas de test	O. SALL	1 j	25/10/2025	50+ cas dans Xray
Installation outils	Équipe	0,5 j	20/10/2025	Env. opérationnel
Configuration CI/CD	O. SALL	0,5 j	17/11/2025	Pipeline fonctionnel
Création jeu de données	R. ROMD-HANE	0,5 j	25/10/2025	Fichiers JSON
Rédaction Plan de Test	O. SALL	1 j	01/12/2025	Ce document

Total préparation : 4 jours

9.3 Exécution des Tests

Description	Responsable	Charge	Date fin	Livrables
Tests unitaires Jest	R. ROMD-HANE	0,5 j	01/11/2025	Rapport Jest + couverture
Tests endpoints API	O. SALL	1 j	02/11/2025	Rapport Supertest
Scénarios E2E	R. ROMD-HANE	1 j	02/11/2025	Collection Postman
Tests de sécurité	O. SALL	1 j	05/11/2025	Rapport sécurité
Tests performance	R. ROMD-HANE	0,5 j	05/11/2025	Rapport performance
Analyse des résultats	O. SALL	0,5 j	20/11/2025	Liste défauts Jira
Correction défauts	Équipe	1 j	30/11/2025	Code corrigé
Re-tests	O. SALL	0,5 j	05/12/2025	Rapport re-test
Rapport Final	O. SALL	1 j	15/12/2025	Rapport de clôture

Total exécution : 7 jours

10 CHAPITRE

CRITÈRES D'ARRÊT ET CONDITIONS DE REPRISE DES TESTS

10.1 Critères d'Arrêt (Suspension des Tests)

Les tests seront suspendus immédiatement si l'une des conditions suivantes est remplie :

Critère	Description	Action corrective
Taux d'échec critique	Plus de 20% des tests critiques échouent	Analyse d'impact + correction urgente
Environnement instable	MongoDB indisponible ou CI/CD en panne	Restauration environnement
Bloqueur technique	Bug bloquant empêchant l'exécution	Correction prioritaire du bug
Données corrompues	Impossible de réinitialiser la BD	Recréation des données
Pipeline non fonctionnel	GitHub Actions ne démarre plus	Debug du workflow YAML
Ressources indisponibles	Absence prolongée d'un testeur clé	Réaffectation des tâches

10.2 Conditions de Reprise des Tests

Les tests peuvent reprendre uniquement si **toutes** les conditions suivantes sont satisfaites :

Condition	Vérification	Responsable
Anomalies résolues	Tous les défauts bloquants sont fermés	Test Lead
Environnement validé	MongoDB + CI/CD opérationnels	Test Engineer
Données restaurées	Jeu de données initial recréé et vérifié	Test Engineer
Tests critiques validés	Au moins 90% des tests critiques passent	Test Lead
Approbation formelle	Encadrant valide la reprise	Encadrant

Processus de Reprise

1. Correction des défauts bloquants
2. Validation environnement (smoke tests)
3. Exécution tests critiques uniquement
4. Si succès > 90% → reprise complète
5. Sinon → nouvelle analyse

11 CHAPITRE

LIVRABLES DU TEST

Cette section identifie l'ensemble des livrables produits durant les activités de test.

11.1 Cas de Test

Description

Ensemble complet des cas de tests fonctionnels, techniques, d'intégration, de sécurité et E2E créés dans Xray.

Contenu :

- Cas de test inscription (signup) - env. 8 scénarios
- Cas de test connexion (signin) - env. 10 scénarios
- Cas de test gestion des rôles - env. 12 scénarios
- Cas de test endpoints protégés - env. 10 scénarios
- Cas de test sécurité JWT - env. 8 scénarios
- Cas de test performance (basiques) - env. 5 scénarios

Total estimé : 50+ cas de test

Localisation :

- 📁 Xray – Projet Jira : TJAA
- Exemple : [TJAA-35] Campagne de Test - Sprint 1

Format : Cas de test structurés Xray avec préconditions, étapes, résultats attendus

11.2 Scripts d'Automatisation (Jest / Supertest)

Description

Scripts Node.js permettant l'exécution automatique des tests unitaires et d'intégration.

Tests unitaires :

- Validation des données (`validators.test.js`)
- Hashing et cryptographie (`crypto.test.js`)
- Génération JWT (`jwt.test.js`)

Tests d'intégration :

- Endpoints d'authentification (`auth.integration.test.js`)
- Middleware JWT (`middleware.test.js`)
- Accès protégé (`protected.routes.test.js`)

Localisation :

-  GitHub Repository : <https://github.com/RanaRomdhane/node-js-jwt-auth-testing>
-  Dossier `/tests/`

Format : Fichiers `.js` versionnés avec Git

11.3 Rapports de Couverture

Description

Rapports générés automatiquement par Jest après chaque exécution CI/CD.

Contenu :

- Couverture globale (statements, branches, functions, lines)
- Couverture par module/fichier
- Fichiers non couverts
- Rapport HTML interactif

Métriques cibles :

Métrique	Objectif
Statements	$\geq 80\%$
Branches	$\geq 80\%$
Functions	$\geq 80\%$
Lines	$\geq 80\%$

Localisation :

- GitHub Actions → Artifacts → coverage-report
- Localement : /coverage/index.html
- Format JSON : /coverage/coverage-final.json

Format : HTML, JSON, LCOV

11.4 Rapports de Tests E2E

Description

Rapports issus des tests End-to-End exécutés avec Postman/Newman/Chai.

Scénarios E2E couverts :

1. Inscription → Connexion → Accès ressource user
2. Inscription admin → Connexion → Accès ressource admin
3. Connexion → Refresh token → Nouvelle requête
4. Tentative accès sans token → Erreur 401
5. Tentative accès rôle insuffisant → Erreur 403

Localisation :

- 📁 Workspace Postman (collections exportées)
- 🌐 GitHub Repository : /reports/e2e/
- 💎 Xray : Campagnes d'exécution E2E

Format : HTML, JSON

11.5 Rapport de Performance

Description

Résultats des tests de charge effectués avec autocannon / loadtest.

Métriques principales :

- Transactions par seconde (TPS)
- Latence moyenne, p95, p99
- Taux d'erreur (HTTP 4xx, 5xx)
- Débit (throughput) en req/s

Objectifs de performance :

Métrique	Objectif
Temps de réponse moyen	< 150ms
p95	< 300ms
p99	< 500ms
Taux d'erreur	< 1%
Support charge	50 req/s

Format : TXT, JSON, graphiques PNG

11.6 Rapport d'Anomalies

Description

Liste exhaustive des défauts détectés pendant les phases de test.

Structure du rapport :

1. Résumé exécutif (nombre total, répartition par criticité)
2. Défauts bloquants (liste détaillée)
3. Défauts majeurs (liste résumée)
4. Défauts mineurs/triviaux (liste)
5. Défauts reportés (backlog futur)
6. Tendances et recommandations

Localisation : ♦ Jira – Projet TJAA

Format : Jira (online), Excel (export)

11.7 Rapport Final de Test (Rapport de Clôture)

Description

Synthèse finale des activités de test conformément au modèle Certilog.

Contenu :

- **Résumé exécutif** : Objectifs atteints, taux de réussite, défis rencontrés
- **Résultats globaux** : Nombre de tests, taux de réussite, couverture, défauts
- **Analyse qualité** : Conformité fonctionnelle, sécurité, performance
- **Métriques CI/CD** : Stabilité pipeline, temps d'exécution, automatisation
- **Défauts restants** : Défauts ouverts, risques acceptés, plan futur
- **Recommandations** : Améliorations, tests à ajouter, optimisations
- **Leçons apprises** : Bonnes pratiques, difficultés rencontrées

- **Conclusion** : Avis final, recommandation de mise en production

Localisation :

- 📄 Document Word : Rapport_de_cloture_v1.0.docx
- 📁 GitHub Repository : /documents/Rapport_Final_Test.pdf

Date de livraison : 15/12/2025

11.8 Données de Test

Description

Jeux de données MongoDB utilisés pour les tests automatisés.

Utilisateurs de test :

JSON

```
1 {  
2   "username": "testuser",  
3   "email": "test@example.com",  
4   "password": "hashed_password",  
5   "roles": ["user"]  
6 }
```

Rôles disponibles :

- user - accès basique
- moderator - accès intermédiaire
- admin - accès total

Tokens de test :

- Tokens valides avec différentes expirations
- Tokens expirés
- Tokens mal formés (pour tests sécurité)

Localisation :

- 📁 GitHub Repository : /tests/data/
- 📄 Script d'initialisation : /tests/setup/seed-test-db.js

Format : JSON

11.9 Outils de Support

Description

Outils et scripts utilisés pour faciliter et exécuter les tests.

Mocks et Stubs Jest :

- Mock MongoDB : `__mocks__/mongoose.js`
- Mock JWT : `__mocks__/jsonwebtoken.js`
- Stub bcrypt : `__mocks__/bcryptjs.js`

Configuration CI/CD :

- `.github/workflows/main.yml` (pipeline principal)

Scripts utilitaires :

- `tests/utils/db-helper.js` (connexion/déconnexion DB)
- `tests/utils/token-generator.js` (génération tokens)
- `tests/utils/reset-db.js` (réinitialisation DB)

Collections Postman :

- `postman/JWT-Auth-API.postman_collection.json`
- `postman/test-environment.json`

Format : JavaScript, JSON, YAML

12 CHAPITRE

RISQUES ET CONTINGENCES

12.1 Tableau des Risques

ID	Description	Impact	Prob.	Contingence	Resp.
RISK-01	Instabilité MongoDB	Fort	● Élevée	MongoDB Memory Server + retry logic	R. ROMD-HANE
RISK-02	Token JWT expiré pendant tests	Moyen	● Moyenne	Tokens longue durée + refresh auto	O. SALL
RISK-03	Pipeline CI/CD en panne	Fort	● Faible	Exécution locale + docs manuelles	R. ROMD-HANE
RISK-04	Défauts bloquants multiples	Fort	● Moyenne	Suspension + session debug prioritaire	Test Lead
RISK-05	Indisponibilité testeur	Moyen	● Moyenne	Cross-training + backup plan encadrant	Test Lead
RISK-06	Données test corrompues	Moyen	● Moyenne	Versioning Git + script reset auto	R. ROMD-HANE
RISK-07	Couverture code < 80%	Moyen	● Moyenne	Sprint dédié tests + priorisation	O. SALL
RISK-08	Problèmes compatibilité	Faible	● Faible	Lock versions + nvm	R. ROMD-HANE

Légende Probabilité

- > ● Élevée (> 50%)
- > ● Moyenne (20-50%)
- > ● Faible (< 20%)

13 CHAPITRE

CRITÈRES DE PASSAGE OU ÉCHEC

13.1 Critères de Succès (GO)

Le projet de test est considéré comme **RÉUSSI** si **TOUS** les critères suivants sont satisfaits :

✓ Critères de Succès

Critère	Seuil min.	Mesure
Taux de réussite tests	$\geq 90\%$	$(\text{Tests réussis} / \text{Total}) \times 100$
Couverture de code	$\geq 80\%$	Jest coverage
Défauts critiques	0 ouvert	Jira - criticité = Bloquant
Défauts majeurs	≤ 2 ouverts	Jira - criticité = Majeur
Pipeline CI/CD	$\geq 95\%$ succès	GitHub Actions - 10 derniers runs
Tests de sécurité	100% validés	Scénarios OWASP passés
Tests performance	Objectifs atteints	Temps $< 300\text{ms}$ (p95)
Documentation	100% livrée	Tous livrables présents



MISE EN PRODUCTION AUTORISÉE

13.2 Critères d'Échec (NO-GO)

Le projet de test est considéré comme **ÉCHOUÉ** si **AU MOINS UN** des critères suivants est présent :

✖ Critères d'Échec

Critère bloquant	Seuil alerte	Impact
Taux de réussite	< 70%	Qualité insuffisante
Défaut critique ouvert	≥ 1 défaut	Fonctionnalité majeure défaillante
Couverture de code	< 60%	Zones non testées importantes
Pipeline CI/CD	Échec constant	Qualité continue impossible
Tests sécurité échoués	≥ 1 vulnérabilité	Risque sécurité inacceptable
Performance dégradée	Temps > 1s (p95)	Expérience utilisateur inacceptable



MISE EN PRODUCTION REFUSÉE

13.3 Critères d'Acceptation Conditionnelle

Si les critères suivants sont présents, une **mise en production conditionnelle** peut être envisagée :

⚠ Acceptation Conditionnelle

Situation	Condition	Plan d'action requis
Taux de réussite	Entre 70% et 90%	Analyse détaillée + correction sous 48h
Défauts majeurs	Entre 3 et 5 défauts	Plan correction priorisé + hotfix prévu
Couverture code	Entre 60% et 80%	Sprint dédié tests + monitoring renforcé
Performance	p95 entre 300-500ms	Optimisation planifiée + monitoring

**MISE EN PRODUCTION SOUS RÉSERVE**

13.4 Processus de Validation Finale

1. Auto-évaluation (Test Lead)



2. Revue par les pairs



3. Présentation encadrant



4. Décision formelle

GO / NO-GO / Conditionnel



5. Documentation



Date de validation finale : 15/12/2025

14 CHAPITRE

INFORMATIONS COMPLÉMENTAIRES

14.1 Contexte Académique

Ce projet s'inscrit dans le cadre d'un **projet académique ENICAR** visant à :

- Appliquer les méthodologies de test professionnelles
- Maîtriser les outils modernes de test et CI/CD
- Développer des compétences en assurance qualité logicielle
- Préparer à des certifications de test (ISTQB)

i Encadrement

Le projet est supervisé par **Mme Hela AOUADI**, enseignante à l'ENICAR, spécialiste en génie logiciel et qualité.

14.2 Contraintes Spécifiques

Type de contrainte	Description
Réglementaire	Aucune contrainte légale particulière (projet académique)
Budgétaire	Budget limité : utilisation d'outils gratuits/open-source uniquement
Temporelle	Deadline stricte : 15/12/2025 (fin du semestre)
Technologique	Stack : Node.js, MongoDB, JWT, GitHub Actions
Pédagogique	Respect du template Certilog + présentation orale obligatoire

14.3 Spécificités Techniques

14.3.1 Architecture du Projet

Structure du projet

```
1 node-js-jwt-auth-testing/  
2 |-- src/  
3 |   |-- controllers/      (logique metier)  
4 |   |-- models/           (schemas MongoDB)  
5 |   |-- middlewares/      (auth, validation)  
6 |   |-- routes/           (endpoints API)  
7 |   +-- utils/            (helpers)  
8 |-- tests/  
9 |   |-- unit/             (tests Jest)  
10 |   |-- integration/      (tests Supertest)  
11 |   |-- e2e/              (tests Chai/Newman)  
12 |   |-- data/             (fixtures)  
13 |   +-- utils/            (test helpers)  
14 |-- .github/  
15 |   +-- workflows/        (CI/CD)  
16 +-- coverage/             (rapports)
```

14.3.2 Technologies Utilisées



14.4 Points d'Attention Particuliers

Points Critiques

Sécurité JWT Validation rigoureuse des tokens (expiration, signature, payload) - tests de sécurité prioritaires

Gestion des rôles Vérification stricte des permissions - tests d'autorisation exhaustifs

Performance MongoDB Indexation correcte des collections - tests de perfor-

mance sur requêtes

🔗 **Traçabilité** Chaque exigence → test → résultat → défaut doit être lié dans Xray

14.5 Livrables Finaux du Projet Global

En plus des livrables de test, le projet académique nécessite :

- ✓ Code source complet (GitHub)
- ✓ Documentation technique (README, API docs)
- ✓ Plan de Test (ce document)
- ✓ Rapport Final de Test
- ✓ Présentation PowerPoint (soutenance)
- ✓ Démonstration live (API fonctionnelle + tests)

Date de soutenance prévue : 20/12/2025

14.6 Critères d'Évaluation Académique

Critère	Pondération
Qualité du Plan de Test	15%
Couverture des tests	20%
Automatisation et CI/CD	20%
Qualité du code (ESLint, best practices)	15%
Rapport Final de Test	15%
Présentation et démonstration	15%
TOTAL	100%

14.7 Contacts et Support

Support Technique

- > 🐙 **GitHub Issues** : <https://github.com/RanaRomdhane/node-js-jwt-auth-testing/issues>
- > ✉ **Email équipe** :
 - > oulimata.sall@enicar.ucar.tn
 - > rana.romdhane@enicar.ucar.tn

Support Pédagogique

>  Encadrante : hela.boukhriss@enicar.ucar.tn

15 CHAPITRE

GLOSSAIRE

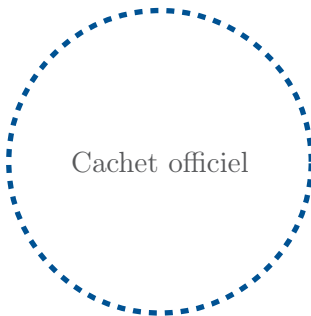
Terme	Définition
API	Application Programming Interface - interface permettant la communication entre systèmes logiciels
Artifact	Fichier produit par un pipeline CI/CD (rapports, logs, binaires)
Bcrypt	Algorithme de hachage cryptographique pour sécuriser les mots de passe
Branch	Branche conditionnelle dans le code (if/else)
CI/CD	Continuous Integration / Continuous Deployment - pratiques d'automatisation
Coverage	Couverture de code - pourcentage de code exécuté par les tests
E2E	End-to-End - tests de bout en bout simulant le parcours utilisateur
ESLint	Outil d'analyse statique de code JavaScript
Flaky test	Test instable donnant des résultats aléatoires (succès/échec)
GitHub Actions	Plateforme CI/CD intégrée à GitHub
Jest	Framework de test JavaScript développé par Facebook
Jira	Outil de gestion de projet et de suivi des défauts
JWT	JSON Web Token - standard d'authentification basé sur des tokens
Lint / Linting	Analyse statique du code pour détecter erreurs et violations
Mock	Objet simulé remplaçant une dépendance réelle dans les tests
MongoDB	Base de données NoSQL orientée documents
Newman	Outil CLI pour exécuter les collections Postman
Node.js	Environnement d'exécution JavaScript côté serveur
OWASP	Open Web Application Security Project - référence en sécurité web
p95 / p99	Percentile 95/99 - mesure de latence

Terme	Définition
Pipeline	Chaîne automatisée d'étapes CI/CD (build, test, deploy)
Postman	Outil de test d'API avec interface graphique
RACI	Responsible, Accountable, Consulted, Informed - matrice de responsabilités
REST	Representational State Transfer - architecture d'API web
Smoke test	Test rapide vérifiant les fonctionnalités de base
SonarQube	Plateforme d'analyse de qualité de code
Sprint	Période de développement itérative (généralement 1-4 semaines)
Stub	Version simplifiée d'un composant pour les tests
Supertest	Bibliothèque Node.js pour tester les API HTTP
TDD	Test-Driven Development - développement guidé par les tests
TPS	Transactions Per Second - nombre de transactions par seconde
Xray	Extension Jira pour la gestion avancée des tests

SIGNATURES ET APPROBATIONS

VALIDATION DU DOCUMENT

Rôle	Nom	Signature	Date
Rédigé par	Rana ROMDHANE & Oulimata SALL	Rana.R Oulimata.S	01/12/2025
Vérifié par	Rana ROMDHANE	Rana.R	01/12/2025
Approuvé par	Hela AOUADI		---/--- / 2025



Document conforme au template Certilog v1.0.1

ENICAR - École Nationale d'Ingénieurs de Carthage

Année académique 2025-2026

Plan de Test

Node.js JWT Authentication

Référence : TJAA-PT-001

Version : 1.0

Date : 01/12/2025

Préparé par

Oulimata SALL & Rana ROMDHANE

Équipe QA - ENICAR

 github.com/RanaRomdhane/node-js-jwt-auth-testing

 rana.romdhane@enicar.ucar.tn

 oulimata.sall@enicar.ucar.tn