Plan de Test

1. Introduction

- **Projet :** EasyBooking (Application de réservation de salles).
- **Objectif :** Garantir que l'application permet de s'inscrire, se connecter, visualiser et réserver des salles.

2. Périmètre des tests (Scope)

La stratégie de test couvre l'intégralité de la chaîne de valeur applicative, divisée en trois axes majeurs :

2.1. Axe Fonctionnel (Tests Système)

Validation du parcours utilisateur complet sur l'interface graphique :

1. **Gestion de Compte :** Inscription (création) et Authentification (connexion/déconnexion).
2. **Catalogue :** Consultation de la liste des salles et de leurs capacités.
3. **Réservation :** Prise de créneau avec validation des disponibilités (gestion des conflits).
4. **Espace Personnel :** Consultation de l'historique des réservations.

2.2. Axe Technique (Tests Automatisés)

Validation de la robustesse du code serveur (Backend Node.js/Express) :

- **Logique Métier (Unitaires) :** Vérification des règles de gestion (validation des dates, formats emails, nettoyage des entrées).
- **API REST (Intégration) :** Vérification des endpoints (Codes HTTP, structure JSON, communication avec la base de données PostgreSQL).

2.3. Axe Qualité (Non-Fonctionnel)

Validation des critères de mise en production :

- **Sécurité :** Tests de résistance aux injections (SQL, XSS), protection des données (Hashage MDP) et configuration serveur (Headers).

- **Performance :** Vérification de la tenue de charge et des temps de réponse API sous stress (50 utilisateurs simultanés).

3. Stratégie de Test

Types de Test	Objectif	Outils	Qui / Quand
Unitaires	Vérifier les petites fonctions isolées (ex: format email, date valide, hachage du mot de passe)	Jest	Automatisé / Code
Intégration	Vérifier que l'API et la Base de données discutent bien ensemble	Supertest + Jest	Automatisé
Performance	Vérifier la rapidité et la tenue de charge de l'API	K6	Automatisé
Sécurité	Vérifier la résistance aux attaques (Injections SQL etc ...)	Manuel / npm audit	Manuel

4. Environnement de Test

- **Frontend :** React.
- **Backend :** Node.js/Express.
- **Base de données :** PostgreSQL (Docker Container).

5. Critères d'acceptation (Definition of Done)

Le projet sera considéré comme valide si :

- Aucun bug critique (crash serveur, impossibilité de réserver).
- Le taux de succès des tests automatisés est de 100%.