**Plan de Pruebas para RedSalud**

**1. Introducción**

* **Objetivo del Documento:** Describir el enfoque, recursos y cronograma de las actividades de prueba para garantizar la correcta funcionalidad del sistema RedSalud, que utiliza tecnologías como React.js, Django, Amazon RDS y JWT.
* **Alcance:** Este plan de pruebas cubre todas las funcionalidades principales de la aplicación, incluyendo la autenticación mediante JWT, el backend en Django con integración de base de datos en Amazon RDS, y la interfaz de usuario en React.js.
* **Referencias:**
  + Documentación de requisitos funcionales y no funcionales del sistema.
  + Documentación técnica de las tecnologías utilizadas (React.js, Django, JWT, Amazon RDS).

**2. Estrategia de Pruebas**

* **Tipos de Pruebas a Realizar:**
  + **Pruebas Unitarias:** Verificar las funciones individuales en el backend (Django) y los componentes en el frontend (React.js).
  + **Pruebas de Integración:** Validar la interacción entre React.js y Django, y entre Django y Amazon RDS, asegurando la correcta comunicación y manejo de datos.
  + **Pruebas Funcionales:** Probar que el sistema cumpla con los requisitos funcionales, incluyendo la autenticación basada en JWT.
  + **Pruebas No Funcionales:** Evaluar el rendimiento de la base de datos en Amazon RDS, la usabilidad de la interfaz en React.js, y la eficiencia general del sistema.
  + **Pruebas de Seguridad:** Asegurar que el sistema sea seguro, especialmente en lo relacionado con la autenticación mediante JWT y la protección de los datos en Amazon RDS.
* **Enfoque de Pruebas:** Se utilizarán frameworks como Jest para pruebas unitarias en React.js y Django, y herramientas de pruebas de seguridad como OWASP ZAP para evaluar vulnerabilidades.
* **Criterios de Aceptación/Rechazo:**
  + Todas las funcionalidades deben cumplir con los requisitos establecidos.
  + El rendimiento del sistema debe ser óptimo bajo carga, especialmente en la interacción con la base de datos de Amazon RDS.
  + La seguridad debe garantizar la protección de los datos personales y la autenticación JWT debe ser efectiva.

**3. Elementos a Probar**

* **Módulos/Componentes:**
  + Frontend en **React.js**: Comprobación de la visualización correcta de la interfaz, interacción de formularios y botones.
  + Backend en **Django**: Verificación de las rutas, lógica de negocio, y las interacciones con la base de datos en **Amazon RDS**.
  + Sistema de **Autenticación JWT**: Validación de la seguridad de la autenticación, incluyendo la gestión de tokens y expiración.
* **Funcionalidades Clave:**
  + Ingreso y gestión de citas médicas.
  + Sistema de autenticación de usuarios.
  + Visualización y generación de reportes en el área administrativa.

**4. Elementos No Incluidos**

* **Exclusiones:**
  + Componentes no relacionados con la interfaz de usuario, como servicios externos no críticos o módulos de administración de servidores, que no están cubiertos por este plan de pruebas.

**5. Recursos y Responsabilidades**

* **Equipo de Pruebas:**
  + **Desarrolladores**: Responsable de crear y ejecutar las pruebas unitarias y de integración.
  + **QA Tester**: Responsable de ejecutar las pruebas funcionales y no funcionales.
  + **Especialistas en Seguridad**: Encargados de realizar las pruebas de seguridad en el sistema.
* **Herramientas de Prueba:**
  + **Frontend**: Jest, Enzyme.
  + **Backend**: Pytest para pruebas de Django.
  + **Base de Datos**: Amazon RDS para pruebas de integración con la base de datos.
  + **Autenticación JWT**: Postman para probar los endpoints de autenticación.
  + **Seguridad**: OWASP ZAP, Burp Suite.

**6. Cronograma**

* **Planificación de Actividades:**
  + **Fase de Pruebas Unitarias**: 06/11/2024 - 10/11/2024
  + **Fase de Pruebas de Integración**: 11/11/2024 - 15/11/2024
  + **Fase de Pruebas Funcionales**: 16/11/2024 - 20/11/2024
  + **Fase de Pruebas de Seguridad**: 21/11/2024 - 25/11/2024

**7. Entorno de Pruebas**

* **Configuración del Hardware/Software:**
  + Servidores de prueba en Amazon Web Services con **Amazon RDS**.
  + Entorno de desarrollo para **React.js** y **Django** en servidores locales.
  + Herramientas de prueba como **Postman** y **OWASP ZAP** para la validación de seguridad.

**8. Gestión de Riesgos**

* **Identificación de Riesgos:**
  + Posibles errores de integración entre React.js y Django.
  + Problemas de rendimiento bajo carga debido a consultas a Amazon RDS.
  + Fallos en la autenticación JWT si el token no se maneja correctamente.
* **Mitigación:**
  + Realizar pruebas de integración de manera temprana y frecuente.
  + Realizar pruebas de carga y estrés para identificar posibles cuellos de botella.
  + Validar los mecanismos de autenticación en cada fase del desarrollo.

**9. Procedimientos de Comunicación**

* **Reportes:**
  + Los informes de avance se generarán semanalmente, incluyendo resultados de pruebas y problemas encontrados.
* **Reuniones:**
  + Reuniones de revisión de resultados de pruebas programadas al final de cada fase de prueba.

**10. Control de Cambios**

* **Procedimientos para Solicitudes de Cambio:**
  + Cualquier cambio que afecte el sistema debe ser aprobado y probado nuevamente para asegurar que no afecte otras partes del sistema.