Plan de Pruebas

1. Objetivo

El objetivo de este plan de pruebas es verificar que la nueva funcionalidad de pagos, que permite a los usuarios realizar pagos a travez de una nueva plataforma, funcione correctamente en lo entornos de QA y Produccion. La pruebas deben validar que el fluyo de pagos este estable. Tambien se asegura que la preubas automatizadas esten corriendo correctamente en el pipeline CI/CD

2. Alcance

Este plan de pruebas se enfocara en:

- Validar pagos exitosos y fallidos.
- Manejo de errores durante el processo de pago.
- Pruebas de compatiblidad en distintos navegadore y dispositivos.
- Comportamiento del sistema en los entornos QA y Produccion.
- Integracion de la nueva plataforma de pago en el pipeline de CI/CD.

3. Casos de Prueba

3.1. Casos Positivos

1. **Pago Exitoso:**
    - Precondiciones: El usuario tiene una tarjeta valida y saldo suficiente.
    - Test Steps:
        1. Seleccionar un producto.
        2. Agregar al carrito.
        3. Hacer click en checkout.
        4. Introducir los datos de la tarjeta de credito.
        5. Confirmar el pago
    - Resultado esperado: La transaccion es aprobada y el usuario se le muestra un mensaje de confirmacion. Tambien validar otras notificaciones como mail.
2. **Pago con tarjeta guardada:**
    - Precondiciones: El usuario tiene una tarjeta valida guardada en su perfil.

- Test Steps:
    1. Seleccionar un producto.
    2. Agregar al carrito.
    3. Hacer click en checkout.
    4. Elegir la tarjeta guardada del perfil.
    5. Confirmar el pago.
- Resultado espserado: La transaccion es aprobada y el usuario se le muestra un mensaje de confirmacion. Tambien validar otras notificaciones como mail.

3.2. Casos Negativos

1. **Pago con tarjeta invalida:**
    - Precondiciones: El usuario introduce una tarjeta de credito no valido.
    - Test Steps:
        1. Seleccionar un producto.
        2. Agregar al carrito.
        3. Hacer click en checkout.
        4. Introducir los datos de una tarjeta invalida.
        5. Confirmar el pago.
    - Resultado esperado: La transaccion es rechazada con un mensaje de error adecuado.
2. **Saldo Insuficiente:**
    - Precondiciones: La tarjeta no tiene salgo para completar el total del pago.
    - Test Steps:
        1. Seleccionar un producto.
        2. Agregar al carrito.
        3. Hacer click en checkout.
        4. Introducir una tarjeta con saldo insuficiente.
        5. Confirmar el pago.
    - Resultado esperado: La transaccion es rechazada con un mensaje de error adecuado.
3. **Pago con tarjeta guardada con datos des actualizados:**
    - Precondiciones: El usuario tiene una tarjeta guardada que esta desactualizada
    - Tests Steps:
        1. Seleccionar un producto.
        2. Agregar al carrito.
        3. Hacer click en checkout.

4. Seleccionar la tarjeta desactualizada.
5. Confirmar el pago.

- Resultado esperado: La transaccion es rechazada con un mensaje indicando que la tarjeta tiene que ser actualizada.

4. Estragegia de Preubas

4.1. Entorno de Prueba

- QA: Para la validacion inicial y preubas automatizada
- Produccion: Para pruebas finales, con un enfoque en la experiencia de usuario y smoke test.

4.2. Tipos de Pruebas

- **Funcionales**: Verificar las funciones principales de la plataforma de pagos.
- **Integracion**: Asegurar que la nueva funcionalidad se comunique bien con la plataforma de pagos.
- **Regresion**: Validar que las nuevas funcionalidades no afecto a otras areas de ecommerce.
- **Compatibilidad**: Verificar que el nuevo proceso de pago funcione en distintos navegadores y dispositivos.
- Pruebas de UI y Diseno

4.3. Herramientas

- **Postman:** Para las pruebas de API y validacion de respuesta.
- **Cypress**: para las purebas automatizadas de la UI.
- **Jenkins / Github Actions:** Para ejecutar las pruebas de automatizacion en el pipeline de CI/CD.
- **Browserstack**: Para las pruebas de compatibilidad.

5. Criterios de Aceptacion

1. Todas las pruebas funcionales deben ser exitosas.
2. Los mensajes de error deben ser claros y legibles par el usuario.
3. Las pruebas de regression deben pasar sin afectar funcionalidades previas.

4. La UI debe compir con el diseno proporcionado por el equipo de diseno.

6. Riesgos

1. Errores no detectados en produccion.
    - Mitigacion: Ejectuar preubas exhaustivas en QA antes de release.
2. Fallos en el pipeline CI/CD
    - Mitigacion: Implementar pruebas de automatizacion en cada push de codigo nuevo.
3. Problemas de compatibilidad en navegadores o depostivos.
    - Mitigacion: Ampliar la cobertura de pruebas y priorizar los despositivos populares de nuestros usuarios.
    -