Los flaky tests son pruebas de software cuyos resultados son inconsistentes, es decir, pasan o fallan de manera impredecible sin cambios en el código subyacente. A diferencia de las pruebas confiables, que producen resultados consistentes, las pruebas inestables crean incertidumbre y presentan desafíos para los equipos de desarrollo de software. En resumen, un flaky test es una prueba automatizada con resultados no deterministas, que puede pasar o fallar de manera intermitente, independientemente de los cambios en el código. Es fundamental abordar estos problemas para garantizar la confiabilidad de las pruebas y la calidad del software.

Los flaky tests pueden ser problemáticos para los programadores por varias razones:

- Inconsistencia en los resultados: Los flaky tests generan resultados impredecibles. A veces pasan y otras veces fallan sin que haya cambios en el código subyacente. Esto dificulta la confianza en las pruebas y puede llevar a errores en la detección de problemas reales.
- Tiempo de depuración: Cuando un flaky test falla, los programadores deben investigar si se trata de un problema real o simplemente una inconsistencia temporal. Esto consume tiempo y recursos valiosos.
- Falsos positivos/negativos: Los flaky tests pueden dar falsos positivos (indicando un problema que no existe) o falsos negativos (no detectando un problema real). Esto puede afectar la calidad del software y la confianza en las pruebas automatizadas.
- Impacto en la productividad: Si los programadores deben lidiar constantemente con pruebas inestables, su productividad disminuye. En lugar de centrarse en resolver problemas reales, dedican tiempo a depurar pruebas.

En resumen, los flaky tests pueden afectar la eficiencia, la confiabilidad y la calidad del software, lo que representa un desafío para los programadores. Es importante abordar estos problemas mediante buenas prácticas de prueba y depuración.

1. Flaky tests dependientes del orden: Estas pruebas generan resultados distintos según el orden en que se ejecutan. Por ejemplo, si una prueba A depende de la salida de otra prueba B, el orden de ejecución puede afectar los resultados. Este problema es especialmente problemático en sistemas distribuidos o paralelos.
2. Flaky tests no dependientes del orden: Estas pruebas producen resultados inconsistentes sin importar el orden de ejecución. Pueden estar asociadas

con condiciones de carrera, sincronización incorrecta o problemas en la lógica de la prueba.

3. Flaky tests dependientes del entorno externo: Estas pruebas son sensibles a factores externos, como la red, la carga del sistema o la disponibilidad de recursos. Por ejemplo, una prueba que verifica una API puede fallar si el servidor está sobrecargado o si hay problemas de conectividad.
4. Flaky tests dependientes del tiempo: Estas pruebas generan resultados diferentes según el momento en que se ejecutan. Por ejemplo, una prueba que verifica un temporizador puede fallar si se ejecuta justo antes o justo después de que expire el tiempo.

Son pruebas de software que generan resultados inconsistentes. Pueden ser dependientes del orden, no dependientes del orden, dependientes del entorno externo o dependientes del tiempo. Para detectarlos, se pueden utilizar análisis estadísticos, reejecución de pruebas y anotaciones personalizadas.

Para abordar pruebas inestables en la suite de pruebas, es crucial:

Reproducir localmente el problema y aislar su causa raíz, revisando el código de la prueba y sus dependencias. Añadir más información a los registros para identificar patrones inesperados. Reejecutar la prueba en diversos entornos para determinar su especificidad. Priorizar la corrección de pruebas inestables para mantener la confiabilidad de la suite. Implementar un monitoreo continuo para detectar pruebas inestables tempranamente y prevenir su introducción en el futuro. Podemos usar varios tipos de objetos simulados ya que nos ayudan a controlar la inestabilidad a costa de limitar el realismo de prueba.

Para las pruebas de interfaces web podemos hacer las pruebas basándolas en eventos por naturaleza en lugar de los eventos basados en tiempo o peso. Además, nos centramos más en los widgets existentes en nuestra interfaz más que en la ubicación exacta de estos ya que introduce debilidad al cambiar de web a aplicación