EXCEPTION IN C#

Overview of exceptions:

What are exceptions in C#

Why we use exceptions

Where exception classes come from

How to respond to runtime errors

Signal runtime error conditions

Define custom exception classes

By the end of this topic you will be able to :

UNDERSTAND THE IMPORTANCE OF ERROR HANDLING

UNDERSTAND THE EXCEPTION CLASS HIERARCHY AND EXCEPTION CONSTRUCTORS

GETTING STARTED WITH EXCEPTIONS

CATCHING,THROWING AND RETHROWING EXCEPTIONS

CREATING AND USING CUSTOM EXCEPTIONS.

WRITING AUTOMATED TESTS FOR EXCEPTION THROWING CODE

In C#, **exceptions** are like alerts that the computer sends when your program runs into unexpected issues. They're like alarms that ring when things go wrong, similar to a smoke alarm alerting you if your oven starts overheating while baking a cake.

In programming, using exceptions is like having a plan for when things don't go as planned. They provide a structured way to handle errors, helping you avoid a complete program crash. Without exceptions, it's like a frozen phone that needs a restart. But with exceptions, your program can handle problems gracefully and keep running, much like how your phone might close a misbehaving app.

Furthermore, exceptions act like notes that tell you where problems occurred. They help you understand issues and fix them, like clues in a detective's notebook. This is especially handy when you're troubleshooting and correcting problems in your code.

Additionally, exceptions protect users' work. Think of using a computer program to create something important, and suddenly the program crashes, causing you to lose everything. Exceptions prevent this disaster. If something bad is about to happen, the program can catch it, warn you, and possibly allow you to save your work before things go downhill.

In a nutshell, exceptions in C# serve as safety nets for your program. They prevent unexpected crashes, aid in identifying issues, and enable you to fix problems before they escalate. Just as alarms, safety measures, and detectives maintain control in different scenarios, exceptions do the same for your program.

UNDERSTANDING THE IMPORTANCE OF ERROR HANDLING

Prevent Program Crashes: Error handling prevents your program from crashing unexpectedly, ensuring it continues to run smoothly even when unexpected issues arise.

Maintain User Experience: Proper error handling provides users with clear and helpful error messages, maintaining a positive and user-friendly experience.

Protect Data and Work: Handling errors safeguards user data and prevents loss or corruption of important information in case of unexpected situations.

Improve Debugging: Effective error handling provides valuable insights into what went wrong and where, making it easier to identify and fix issues during development and maintenance.

Enhance Program Reliability: By managing errors, you create a more reliable program that can handle problems gracefully, reducing disruptions and increasing overall stability.