John Munguia

CS 499

July 27, 2025

CS 405 Module Four – Exceptions Activity: Process Summary

This activity focused on identifying unsafe failure points in a banking web application and applying secure C++ exception handling to prevent the program from crashing without warning. The goal was to catch all exceptions, handle them properly, and ensure the program could continue or exit safely with clear user messages.

To begin, I created a custom exception class called CustomException that inherits from std::exception and provides a specific error message. I also simulated a failure in the nested logic using a std::runtime\_error, and wrote a divide() function that checks for divide-by-zero errors and throws std::invalid\_argument if one occurs. I also added a check for possible floating-point overflow using std::numeric\_limits<float>::max() to prevent undefined behavior.

All potentially dangerous code is wrapped in try-catch blocks that catch specific exceptions first, followed by a general std::exception handler. At the top level in main(), I included a final catch (...) block to handle any uncaught or unknown exceptions as a last resort. All error messages are printed using the what() method, and logging is clearly labeled with tags like [INFO] and [ERROR]. Output is directed to std::cerr where appropriate for secure logging.

One challenge I ran into was organizing the exception handlers in the correct order. I needed to make sure custom exceptions were caught before general ones to avoid them being missed. I also had to ensure every possible failure was anticipated without relying too much on catch-all handlers. These issues were resolved through testing and following best practices for exception safety.

As seen in the screenshot, the final version of the program successfully catches multiple exception types and exits cleanly with a return code of zero. The design avoids exposing sensitive information and demonstrates secure handling, clear error messaging, and reliable program behavior.

A screenshot of a computer

AI-generated content may be incorrect.