

MICHIGAN STATE

U N I V E R S I T Y

Project Plan

Internal Telemetry for TechSmith Products

The Capstone Experience

Team TechSmith

Ryan Ciffin
Zackary Schreur
Zhuolun Xia
Ben Hickmott
Dakota Locklear

Department of Computer Science and Engineering
Michigan State University

Spring 2019



*From Students...
...to Professionals*

Functional Specifications

- Standalone Internal error monitoring application
- Its main purpose is to compile an error report containing information from TechSmith applications crashes.
- The information collected will consist of hardware information, product log file information, and a windows error report.
- Once collected it will be organized and displayed on a web portal where developers can research a solution.



Design Specifications

- Internal Telemetry Framework
- Prototype Applications
 - C# Application that calls into native C++
 - C++ Application that calls into C#
- Web Portal
- Azure SQL Database



Screen Mockup: Web Portal

The screenshot displays a web portal for 'TechSmith Internal Telemetry'. The browser address bar shows 'www.TechSmith.com/Telemetry'. The main header features the TechSmith logo and a navigation menu with items: Home, Product Selection, Camtasia (selected), Camtasia Studio 2.8.1 (highlighted), Camtasia Studio 2.7.5, Snagit, Snagit 4.0.0, Snagit 3.8.1, Settings, Help, and About.

The central content area is titled 'Camtasia Studio Version: 2.8.1'. It includes a search bar and a list of error reports:

- Type: Segfault, Date: 01/22/2019, User: John Doe
- Type: Null-pointer, Date: 12/25/2018, User: Jane Doe
- Type: Null-pointer, Date: 12/20/2018, User: Joe Crash
- Type: Index OOB, Date: 12/03/2018, User: Josh Posh

A line chart shows error trends from Jan 2018 to Apr 2018 for Segfault (green), Null-pointer (orange), and Index OOB (blue). The Segfault count peaks in March 2018.

User profile for John Doe (JohnDoe@TechSmith.com) is shown, with error details: Type of Error: Segfault, Date of Error: January 22, 2019.

System Information is displayed in a table:

System Information	WER Report	Log File	User Description
Operating System:	Microsoft Windows 10 Home		
Version:	10.0.17134 Build 17134		
System Type:	x64-based PC		
Processor:	Intel Core i7-6700HQ CPU		
Memory (RAM):	16.0 GB		
Graphics Card:	NVIDIA GeForce GTX 960M		



Screen Mockup: Web Portal

TechSmith Internal Telemetry

www.TechSmith.com/Telemetry

TechSmith®

Snagit

Version: 4.0.0

Home

Product Selection:

- Camtasia
 - Camtasia Studio 2.8.1
 - Camtasia Studio 2.7.5
- Snagit:
 - Snagit 4.0.0**
 - Snagit 3.8.1
- Settings
- Help
- About

Search...

Type: Segfault
Date: 1/25/2019
User: Jane Doe

Type: Null-Pointer
Date: 01/20/2019
User: John Doe

Type: Index OOB
Date: 12/03/2018
User: Josh Posh

Jan 2018 Feb 2018 Mar 2018 Apr 2018

Legend: Segfault (Green), Null-pointer (Orange), Index OOB (Blue)

John Doe
JohnDoe@TechSmith.com

Type of Error: Null-Pointer
Date of Error: January 20, 2019

System Information | WER Report | Log File | **User Description**

While working on a PowerPoint presentation I attempted to take a screen capture of the slide I was editing. After taking the screen capture, Snagit failed to load and a blank image appeared in the Editor. After closing Snagit and reopening it and attempting to take the picture again it seemed to be working correctly.



Screen Mockup: C# to C++

The image shows a software window titled "PROTOTYPE1" with standard window controls (minimize, maximize, close). The window contains two dropdown menus and a button. The first dropdown menu is labeled "Select Layer" and has "C#" selected. The second dropdown menu is labeled "Select Error" and has "Null Reference Exception" selected. Below these is a button labeled "EXECUTE ERROR".

PROTOTYPE1

Select Layer

C#

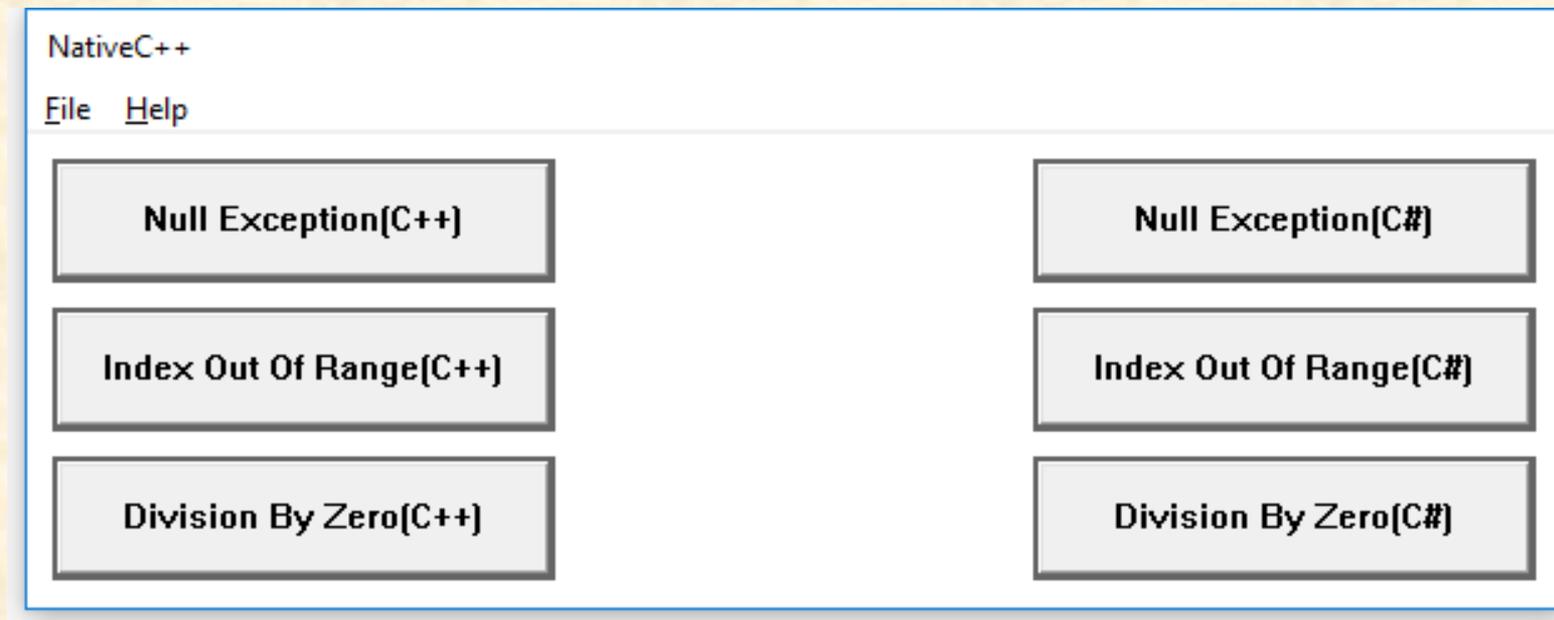
Select Error

Null Reference Exception

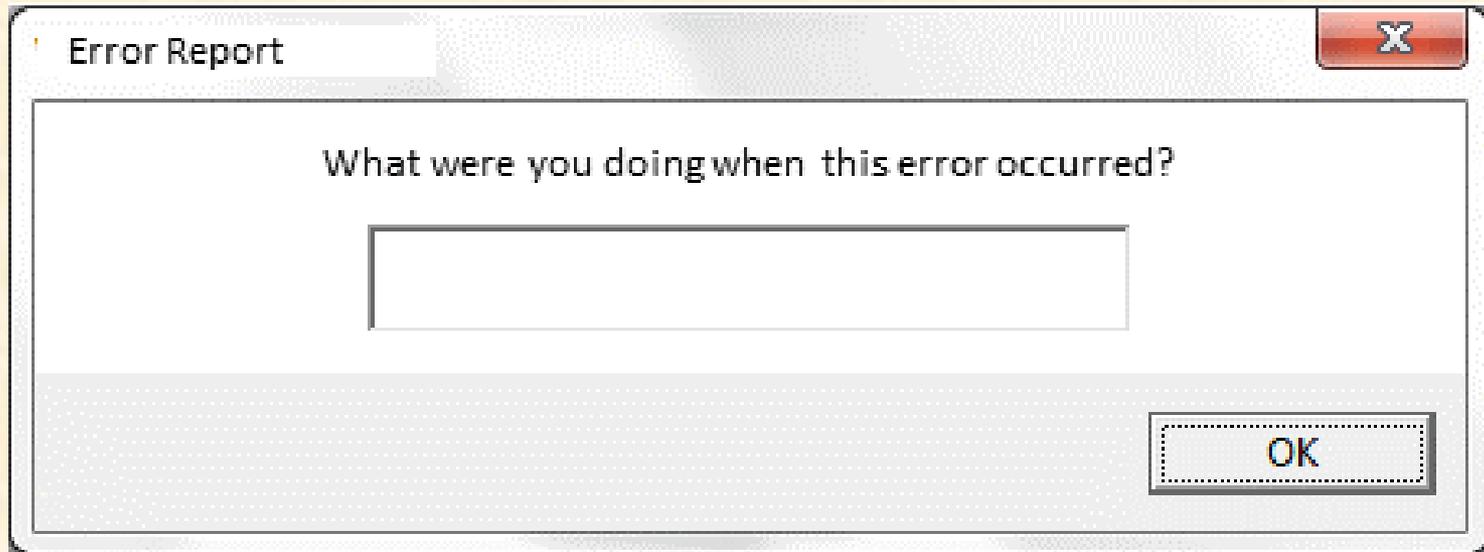
EXECUTE ERROR



Screen Mockup: C++ to C#



Screen Mockup: Error Dialog



Technical Specifications

Software Technologies:

Windows Error Reporting API:

The Windows Error Reporting API will be used to construct error reports when our Internal Framework detects that an application crashed

Microsoft Store Analytics API:

The Microsoft Store analytics API lets you programmatically retrieve analytics data from applications

DxDiag:

Windows application for collecting hardware information

Development Environments:

ASP .NET Core: Web Interface:

Our web interface for viewing application errors, crashes and hangs will be developed in ASP .NET Core

C#: Prototype Application:

One prototype application will be written in C# with an interop layer to call into native C++.

C++: Prototype Application:

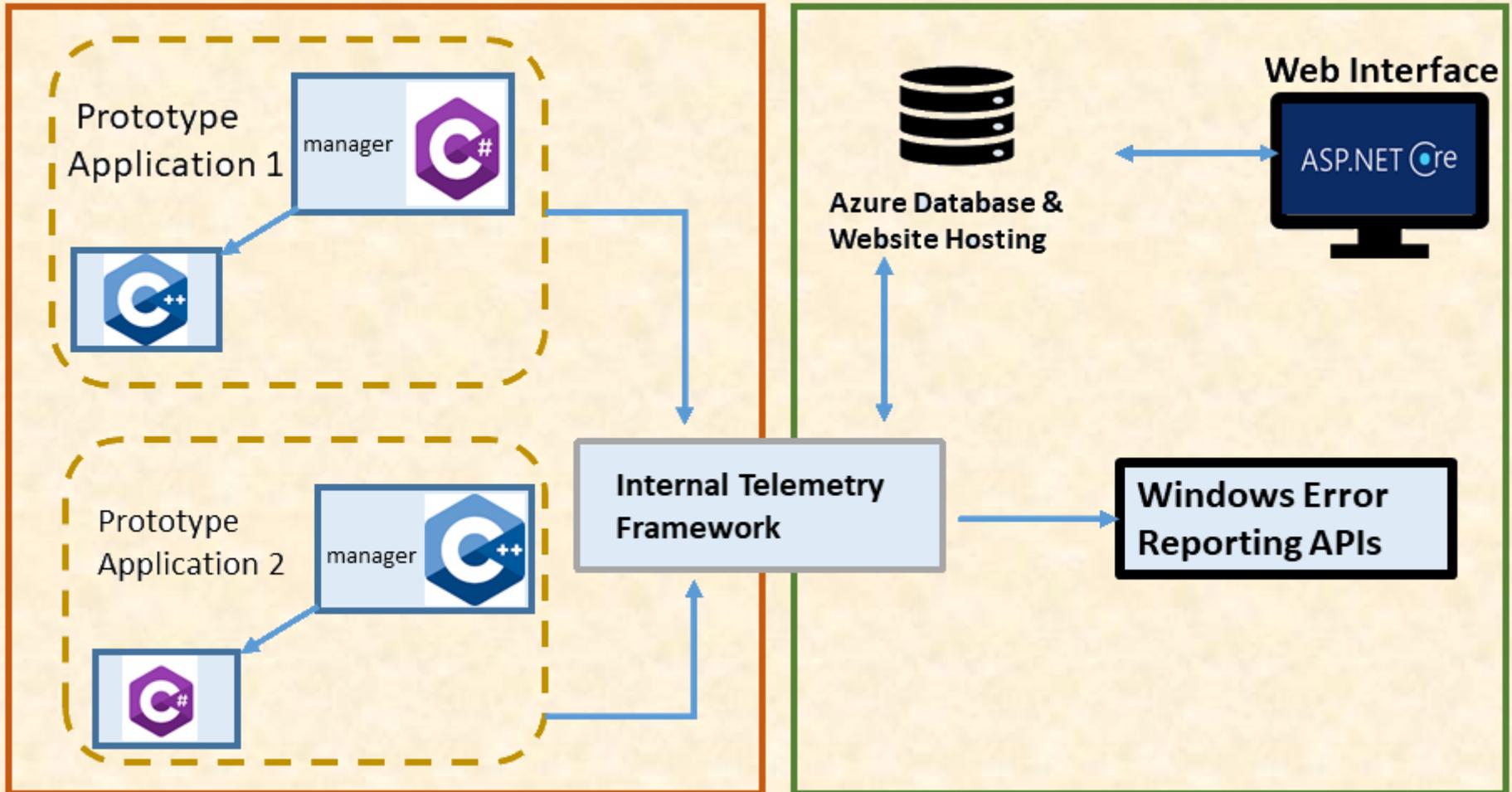
A second prototype application will be written in C++ with an interop layer to call C#.

C++: Internal Telemetry Framework:

Our Internal Telemetry Framework will be a DLL developed in C++



System Architecture



System Components

- Hardware Platforms
 - Azure Database Hosting
- Software Platforms / Technologies
 - VMWare Fusion 10
 - Windows 10
 - Visual Studio 2017
 - Microsoft Azure



Risks

Risk 1: Internal telemetry that can be incorporated into any Win32 application

Description: Universal telemetry application that can be incorporated regardless of the underlying architecture.

Mitigation: We are developing the internal telemetry as stand alone framework that collects information from a location that is universal between applications.

Risk 2: How to tell when an application crashes and where to find WER file

Description: When there is a crash Windows Error Reporting must know that the crash has occurred and what information is to be collected.

Mitigation: We have caused various application crashes and observed unique characteristics to watch for and browsed the resulting archives to find the WER file.

Risk 3: Getting hardware information using DxDiag in C++

Description: DirectX diagnostics will be used to gather system information relating to the crashed application's hardware.

Mitigation: We have researched how to use DxDiag in the command line and designed a basic application that implements this to collect the hardware information.

Risk 4: Implementing the interop layer between C++ and C#

Description: The two prototype applications must have the same functionality but be built on different architectures.

Mitigation: We have researched different implementations and examples of the C++/CLI layer and incorporated this layer into the prototype applications.



Questions?

?

?

?

?

?

?

?

?

?

