MICHIGAN STATE UNIVERSITY

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

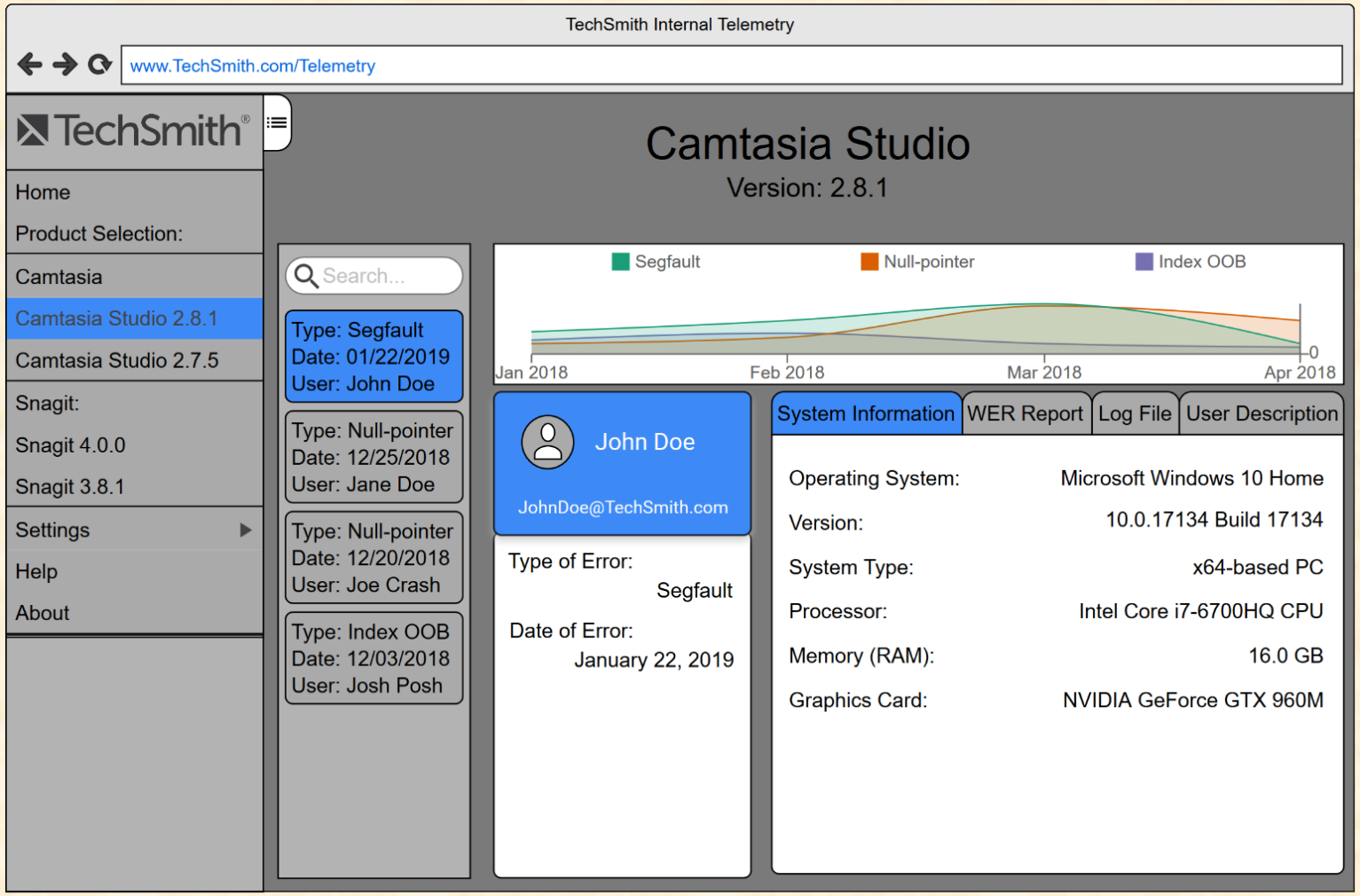
Functional Specifications

- 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.

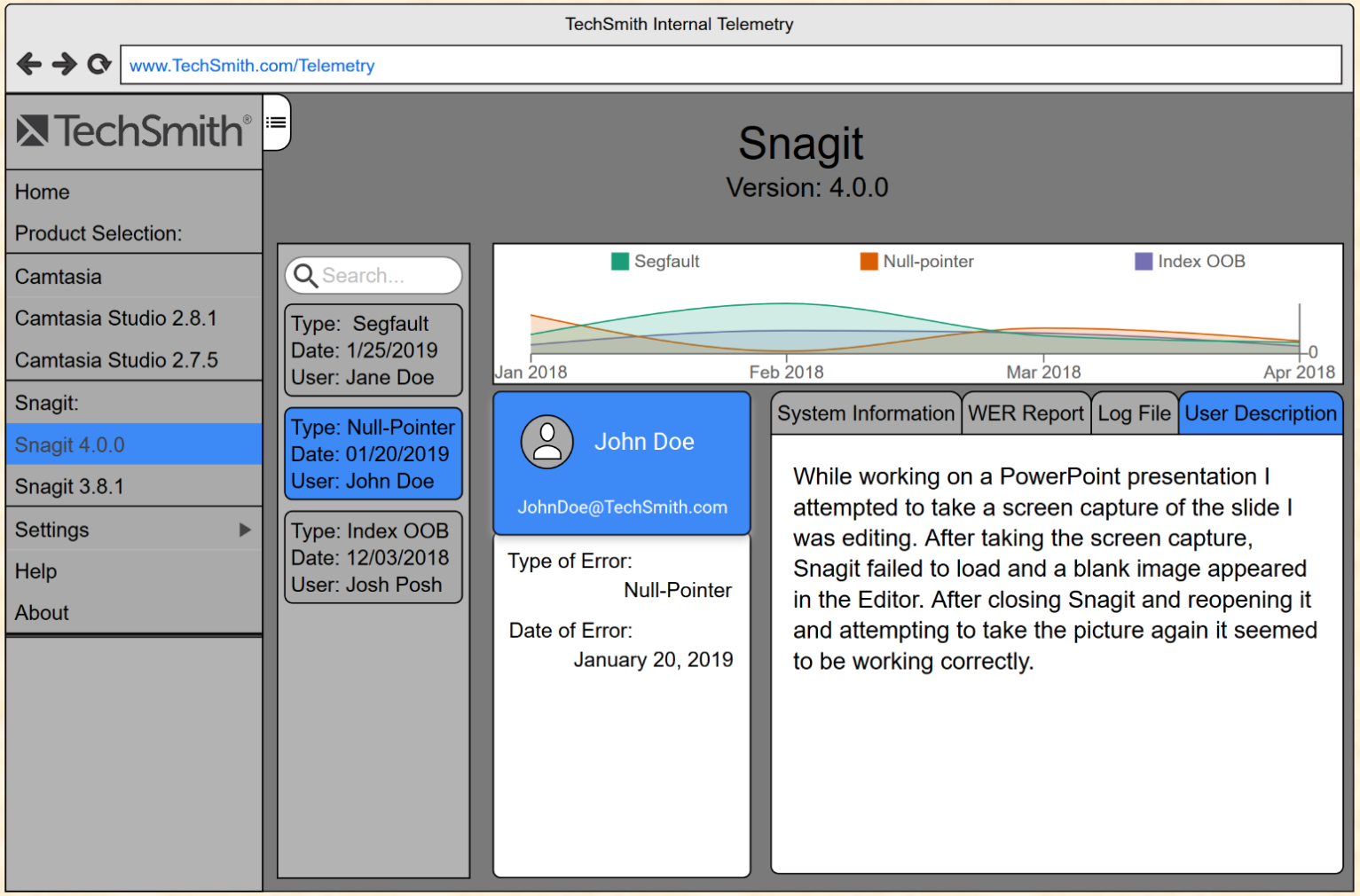
Design Specifications

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

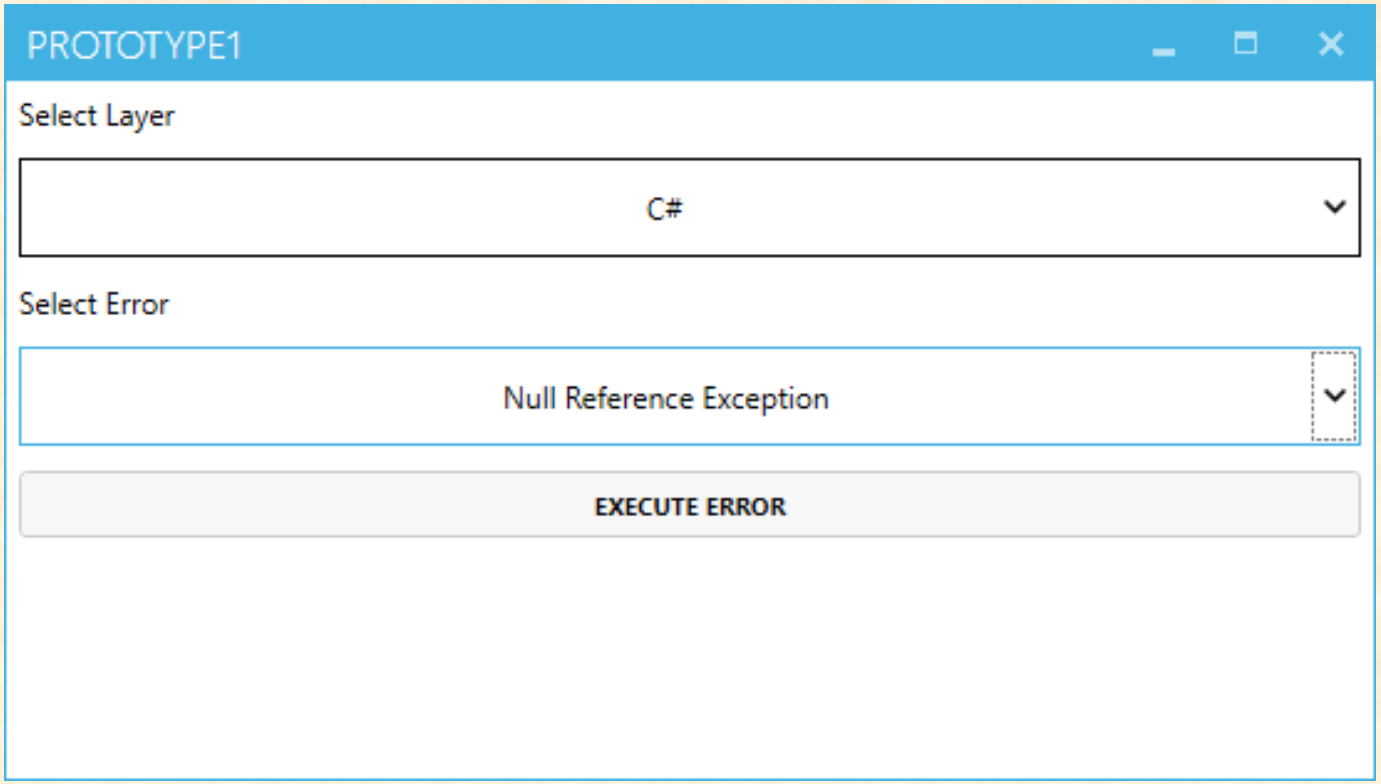
Screen Mockup: Web Portal



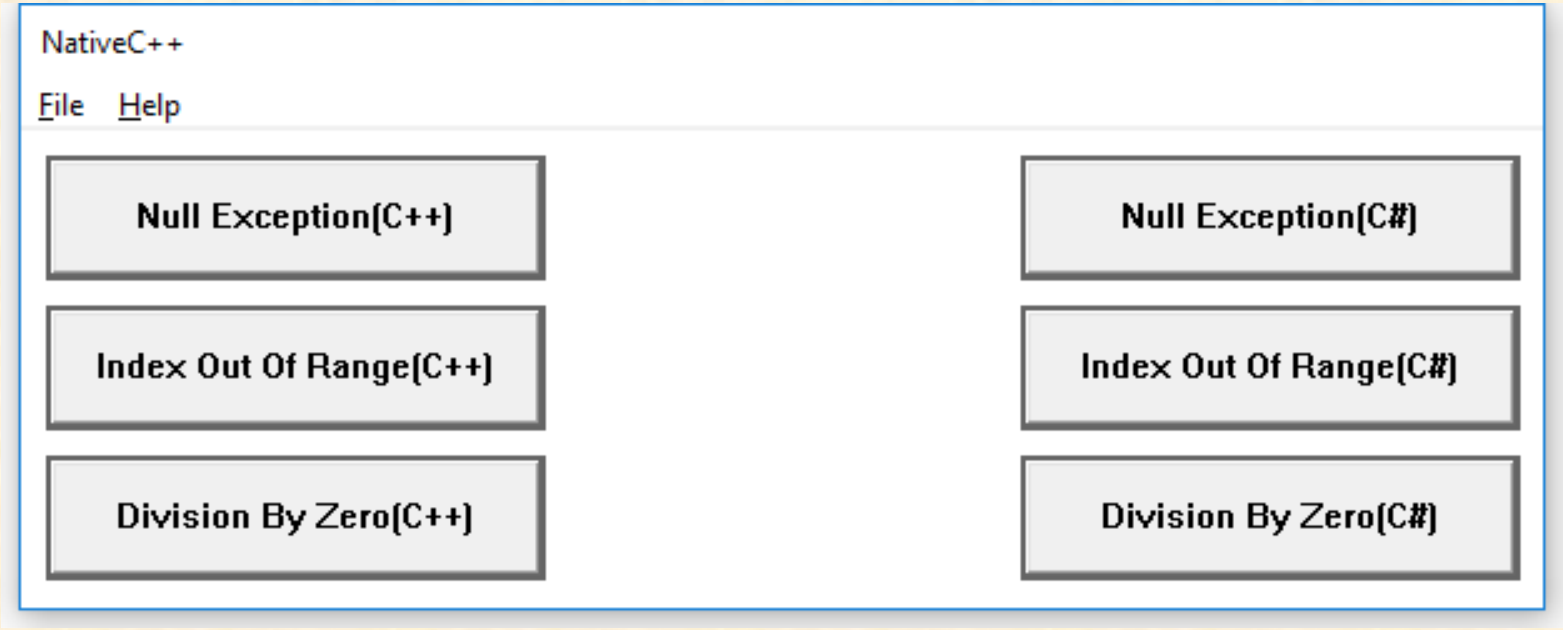
Screen Mockup: Web Portal



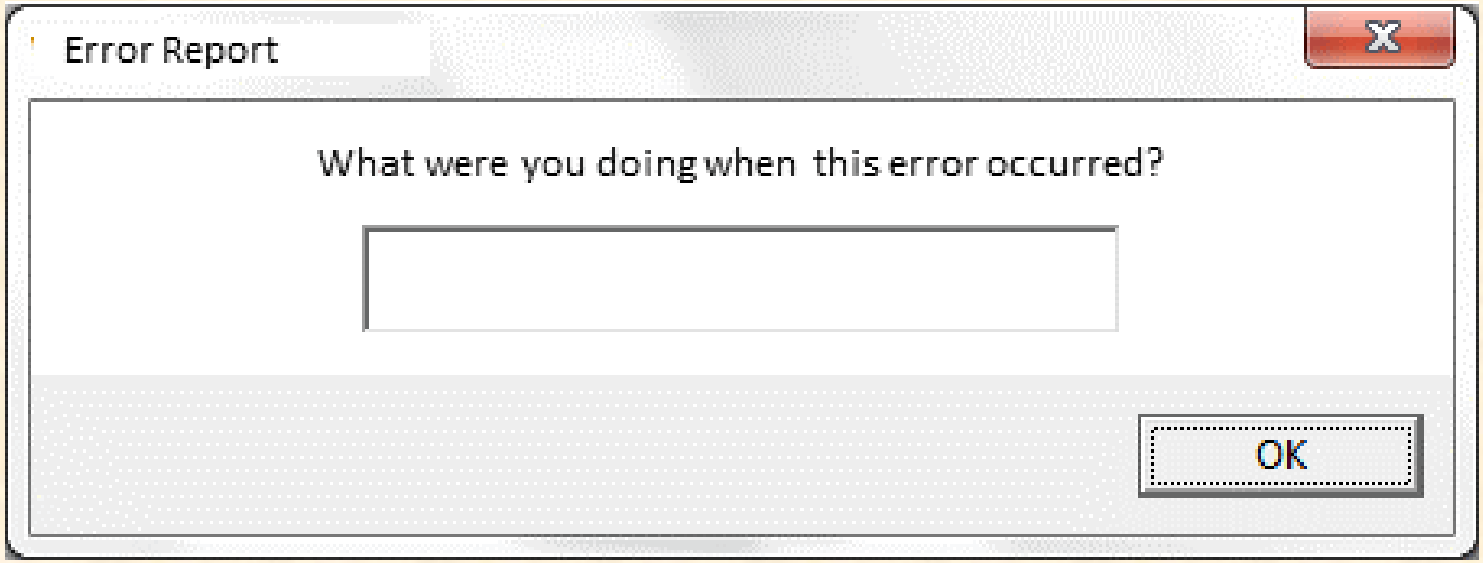
Screen Mockup: C# to C++



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++.

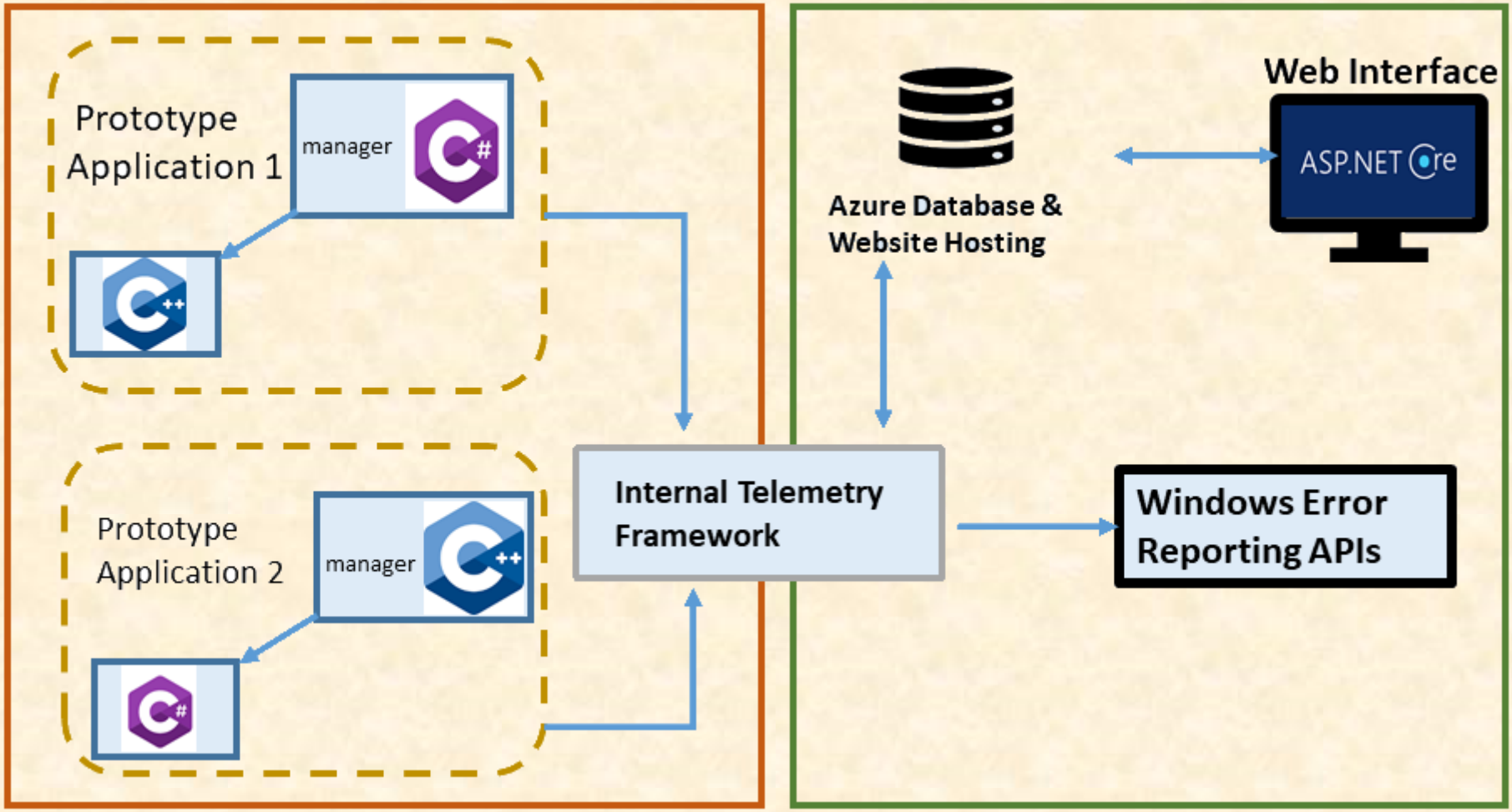
C++: Prototype Application:

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?