VisualPro - Proposal

1st Given Edward Patch
Software Engineer Student (of BSc Year 3)
Independent Project
University of Wales Trinity St. Davids (of Mike Dacey)
Swansea, Wales
Student ID: 1801492

CONTENTS

***Abstract*—Documentation provides technical information to use the Programming Planner library. Technical information includes examples of executions and briefs the developer on using the library effectively.**

***Index Terms*—Documentation, Programming Planner, Development**

I. INTRODUCTION

Programming Planner generates different languages based on the user input. A provided Extensible Markup Language (XML) file, available on the VisualPro's GitHub Repository Add Link. The documentation guides the developer of ways to import and call the Programming Planner's Dynamic-Link Library (DLL) packages with explanations of DLL injection points and displays a series of examples of the Programming Planner's capabilities.

II. IMPORTING THE LIBRARY

Libraries are available for each language—explaining how to connect to Programming Planner's DLL packages. A web link with example code is attached to understand how a DLL library loads into each language listed. It is essential to mention that DLL files will not work on a Linux system, and a Shared Object (SO) library is unavailable. If the language is not displayed in the list below, then search on a Search Engine 'How to import a DLL file in *languageName*'. The code block below shows a communication example with two functions; The first function displays the string 'C++ says Helloworld', and the second function enables addition and subtraction from two numbers to support the examples in the following segments.

```
extern "C" _declspec(dllexport) const
    char* Helloworld();

// opt == true(Addition) and opt ==
    false(Subtraction)
extern "C" _declspec(dllexport) double
    Calculator(double x, double y, bool
    opt);
```

A. Language: C++

One method to import a DLL library in C++ is to import the 'windows.h' and 'tchar.h' libraries. To load the DLL library, the macro *LoadLibrary* and *GetProcAddress* function found in 'windows.h' include is required. The 'tchar.h' library casts a *const char** datatype to a *unicode* format. *FreeLibrary* closes the DLL library and clears the memory from the 'dll' variable name.

```
#include <iostream>
#include <windows.h>
#include <tchar.h>
typedef const char* (*Helloworld_t)();
typedef double (*Calculator_t)(double
    x, double y, bool opt);

int main() {
  auto dll = LoadLibrary(
    _T("DLL_Library.dll"));
  auto helloworld =
      (Helloworld_t)GetProcAddress(dll,
      "Helloworld");
  auto calculator =
      (Calculator_t)GetProcAddress(dll,
      "Calculator");

  if(dll) {
    std::cout << helloworld() << "\n";
    std::cout << "Add: " <<
        calculator(7, 3, true) << " |
        Subtraction: " << calculator(7,
        3, false);
  } else std::cout << "Not Found";

  FreeLibrary(dll);
  return 0;
}
```

List of terminologies used in this document:-

- XML - Extensible Markup Language.
- DLL - Dynamic-Link Library.
- SO - Shared Object.
- IDE - Integrated Development Environment.