**Project Documentation**

VSS Replacement with Git

Summer Internship 2020

Content

[Overview 1](#_Toc49334446)

[Prerequisites 1](#_Toc49334447)

[Installation Guide 1](#_Toc49334448)

[Project Description 2](#_Toc49334449)

[Batch Files 3](#_Toc49334450)

[Git Based C++ functions 3](#_Toc49334451)

[GUI made for this API 5](#_Toc49334452)

[a. Shortcuts and Buttons 5](#_Toc49334453)

[b. Dialogs 6](#_Toc49334454)

# Overview

**Supervisors:** Csóka Lóránd, Szabó Zoltán, Jakab Enikő

**Period:** 20th July – 28th August

**Creators:** Horváth Áron, Magyari Zsuzsanna

**Language:** C++

**Description:** create an application interface identical to the Visual SourceSafe API, which provides the same functionalities, but in the background operates with GIT.

# Prerequisites

Git needs to be installed on the computer.

# Installation Guide

The project can be found on the innersource.accenture webpage

* <https://innersource.accenture.com/users/szabo.zoltan/repos/vss_to_git/browse/VssToGit>

This repository contains the following:

* The “VssToGit” folder contains the source code. (the C++ DLL, the batch files and the GUI).
* The “release” folder contains the executable file with all the DLLs
* “Documentation” contains the PPT

To use this GUI you need to follow these steps:

1. Download the “release” folder from the innersource
2. Open the folder and run the VssToGit.exe file

**Notices: 1.)** Do not change the directory, because it also contains the batch files and the DLL. Without those folders, the application won’t work!

**2.)** When running the GUI windows defender may block it because it has noticed a suspicious file. You need to click “Run Anyway”. Those suspicious files are the batch files, there is nothing to be worried about.

# Project Description

By using this application, the user will be able to use the GIT version control system under their project.

The project is separated into three parts:

* *Batch Files*: these files are running the git functions in the background
* *C++ DLL library*: here are the functions which are calling the batch files
* *GUI*: an implementation of the library functions in the graphical user interface

## Batch Files

For making the git commands accessible by the library, we needed to implement them in batch files to automate some processes.

**Notice:** every batch file has the same name and functionality as it’s C++ function. For further information about the batch files check below.

## Git Based C++ functions

This contains the basic functions which are required by the version control system. Every function calls a batch file. It’s containing the following functions:

* + **void exportProject** (string pathtoWorkingDirectory, string projectName);

Save the git repository to the local computer. The compressed file can be found under the working directory.

* + **void exportFolder** (string pathtoWorkingDirectory, string folderName);

Save a specific folder from the git repository to the local computer. The compressed file can be found under the selected folder.

* + **void gitInit** (string pathtoWorkingDirectory);

Initializes a git repository under the selected folder.

* + v**oid deleteFile** (string pathtoWorkingDirectory, string fileName, string& errorMessage);

Delete the selected file/files/directory from the repository.

* + **void renameFile** (string pathtoWorkingDirectory, string oldName, string newName, string& errorMessage);

Rename the selected file/files/directory.

* + **bool isFileExisting** (string pathtoWorkingDirectory, string fileName, string& errorMessage);

Check if a file exists in any version in the git repository.

* + **bool checkoutFile** (string pathtoWorkingDirectory, string fileName, string& message);

Get the latest version of a file from the repository (the same as “pull” in Git)

* + **void checkInFile** (string pathtoWorkingDirectory, string commitMessage, string& errorMessage);

Save the changes of the current file. Basically, make a new copy of the current file (same as the commit in Git).

* + **vector<string> getFolder** (string pathtoWorkingDirectory, string folderName, bool onlyFiles, string& errorMessage);

Return the files under the selected folders.

* + **void getFile** (string pathtoWorkingDirectory, string fileName, string& errorMessage);

Print the current file.

**Parameter description:**

* + pathtoWorkingDirectory: this parameter takes a path, where the user can work with the current files or folders.
  + fileName: takes the name of a file
  + errorMessage: in this parameter the errorMessage will be returned which is produced by the git functions
  + commitMessage: the commit message will be returned in this parameter
  + folderName: takes a name of a directory.

## GUI made for this API

This GUI is similar to the old VSS’s GUI. The user can navigate between the files and folders, which are in the git repository. This GUI replaces the git command line commands and it give us a great user experience. The user can:

* + Set a working folder: a git repository will be initialized under this folder (does not need to exist, it can be created in the GUI)
  + Add files: adding existing files or folder from the computer to the git repository
  + Edit/view files: every file can be viewed or edited in the QT’s basic editor.
  + Export: the whole project or a subfolder can be exported to the computer. The zip file will be placed under the selected folder
  + Delete/rename: the files can be deleted or renamed in the git repository
  + Check-in/Check-out: a file or a folder can be checked in with a message. Or check out

### a. Shortcuts and Buttons

We put **shortcuts** and **buttons** for easy access to all the functionality in the GUI, to provide a better user experience.



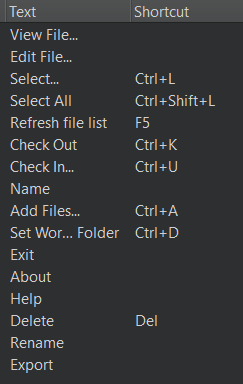
The buttons functionalities (from left to right):

* + - 1. Set working directory
      2. Add files to the working directory
      3. Check In
      4. Check Out
      5. View file
      6. Edit file
      7. Refresh files
      8. Help

**Notice**: the buttons and shortcuts will not always be accessible; the following conditions need to be fulfilled:

* If a working directory is not chosen, nothing can be done
* If a file is not checked out it can’t be edited or checked in
* If a file is checked out, you need to make a check in before the next check out

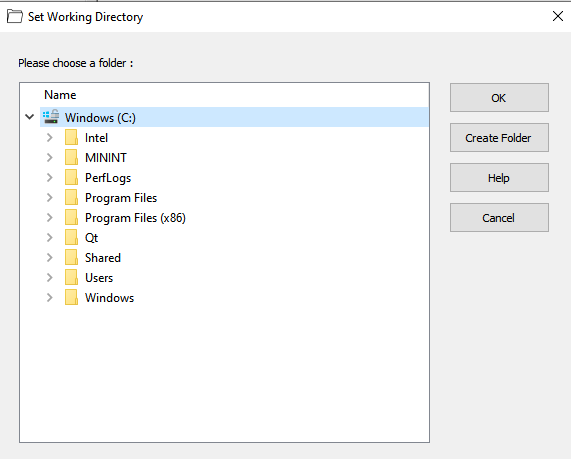
The shortcuts are the following:



### b. Dialogs

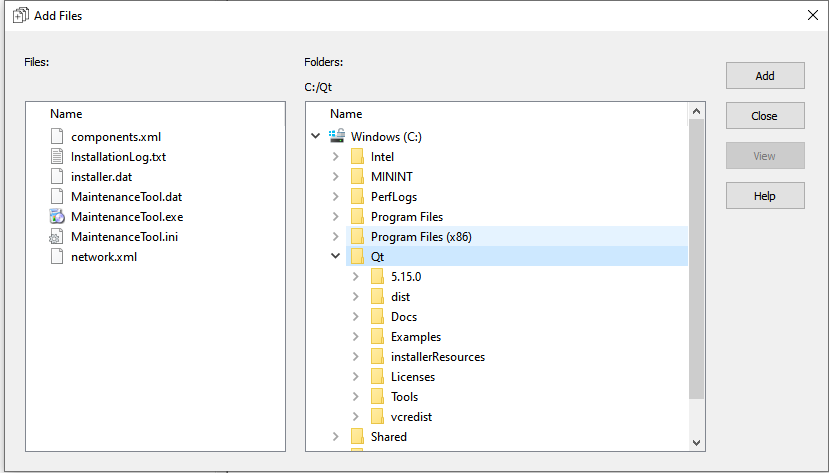
To make this GUI more user friendly we used dialogs for doing some action. Every dialog has a Help button which contains the basic information.

1. Set working directory dialog



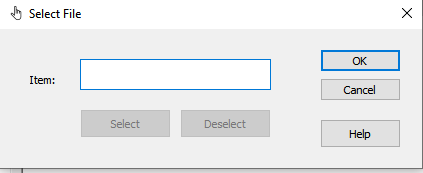
This dialog will appear when the user pressed the “Set Working Folder” button or “CTRL+D”. It allows them to choose one folder from the computer where a git repository will be initialized.

1. Add files dialog



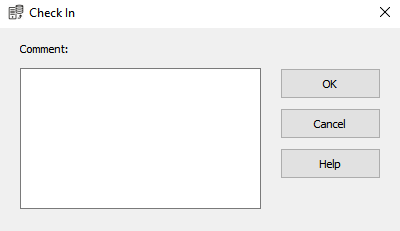
Will appear when the user pressed the “Add files” button or “CTRL+A”. It allows them to add files or folders in the repository. The available folders will be displayed on the right and the available files will be visible at the left.

1. Select dialog



Will appear when the user pressed “Select” from the menu bar or pressed “CTRL+L”. It allows them to search for a specific file from the working directory.

1. Check In dialog



Will appear when the user pressed the “Check In” button, pressed “CTRL+U” or it can be accessed with the help of the context menu (right click). It allows them to check in a file/folder in the working directory.