WINDOWS APPLICATION THRASHING TOOL (WATT)

Test Plan Document

Joseph Caton, My Nguyen, Brooks Olney, and Thomas Pannozzo

November 25, 2018

Version 1.0.1

REVISION HISTORY			
DATE	**VERSION**	**DESCRIPTION**	**AUTHOR**
10/05/2018	1.0.0	Original document	Joseph Caton, My Nguyen, Brooks Olney, Thomas Pannozzo
11/25/2018	1.0.1	Updated user story	My Nguyen

TABLE OF CONTENTS

INTRODUCTION

This Test Plan Document is derived from the Technical Requirements Document, Design Review, and Specifications Document that describes the appropriate strategies, process, workflows, and methodologies used to plan, organize, execute, and manage testing of WATT.

PURPOSE

The Windows Application Thrashing Tool (WATT) is designed to detect potential thrashing on the Windows operating system (OS) and send alerts. This tool will monitor and poll all the processes on the OS to predict potential thrashing. Once thrashing has occurred, WATT will integrate its machine learning program to predict future thrashing and send out an alert to the IT System Administrator. Thrashing for the purposes of these requirements will be tested in a virtual machine by overloading memory and CPU with processes prior to integrating into the business's (SunView Software) environment.

SCOPE

WATT is designed to help the IT department in detecting potential thrashing and sending alerts to the IT System Administrators to take further actions. The test scope of WATT includes the following:

- Testing of all polling, monitoring, and reporting functions
- Testing of the machine learning components

QUALITY OBJECTIVE

PRIMARY OBJECTIVE

The primary objective of testing WATT is to ensure that the system meets the full requirements while maintaining the quality of the product. At the end of the product development cycle, the user should find that the detection tool has met or exceeded all their expectations as detailed in the requirements.

SECONDARY OBJECTIVE

The secondary objective is to identify and expose all issues and associated risks and ensure that all issues/risks are addressed before the project deadline.

ROLES AND RESPONSIBILITIES

The roles and responsibilities are listed below for each team player.

DEVELOPER

The developer is responsible for developing the detection tool and developing test cases and requirements.

TESTER
The tester is responsible testing all the test cases and requirements.

DEBUGGER
The debugger is responsible for fixing any bugs or issues that the tester has found.

DOCUMENTS

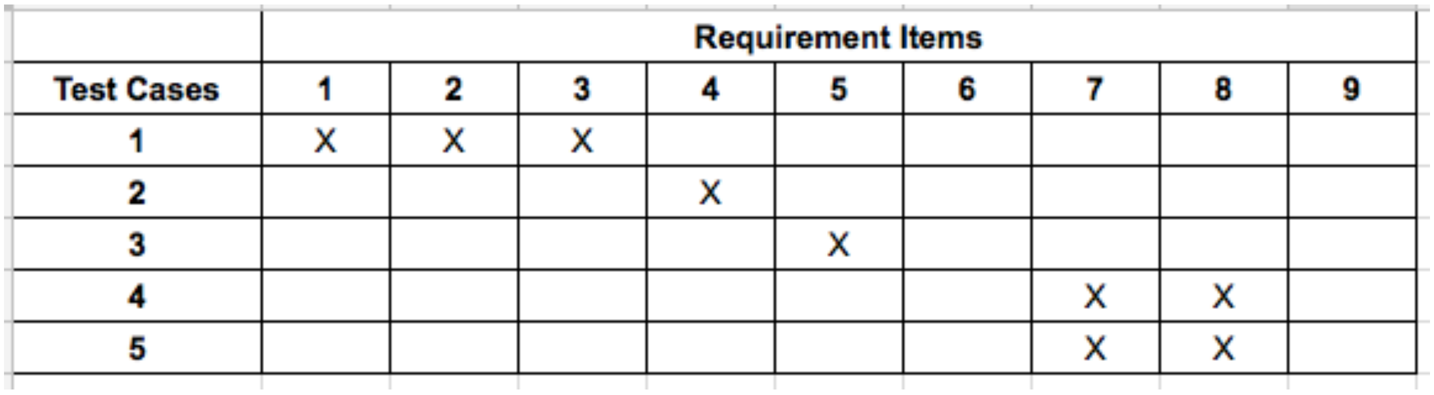
The following documents are used to track the content of this project and the deliverables:

- Business Requirements Document Version 2.05
- Specification Document Version 1.0.3
- WATT Design Review
- Functionalities of WATT i.e. tasks/roles, user stories, sprints

PURPOSE OF TEST PLAN
WATT is expected to detect thrashing based on the monitoring and polling results of page faults, low CPU usage, and high memory usage. The purpose of test plan document is to:

- specify the approach that testing will use to test the product and the deliverables;
- break down the product into distinct areas and identify the features that are being tested;
- identify testing risks;
- and indicate the tools used to test the product.

TRACEABILITY MATRIX
The traceability matrix is used to link the test scenarios to the requirements and test cases. The requirements traceability is defined as the ability to describe and follow the life of a requirement.



	Requirement Items								
Test Cases	1	2	3	4	5	6	7	8	9
1	X	X	X						
2				X					
3					X				
4							X	X	
5							X	X	

OPERATING ENVIRONMENT
WATT's algorithm will be written in C# and Python with the implementation of machine learning for thrashing. WATT will have the potential to be integrated into the business's (SunView Software) environment.

ASSUMPTIONS / DEPENDENCIES
The machine learning portion of WATT not producing the correct output. WATT falsely identifies thrashing and sends an alert too early.

REQUIREMENTS OF EXTERNAL INTERFACE

USER INTERFACES
Users will utilize the command line to run the program. If the program detects the OS thrashing it will alert the user through a dialog window. Upon successful functionality of the program, integration will be considered with the business's (SunView Software) Web platform. In this scenario, the program will alert the user through the API.

HARDWARE INTERFACES
WATT is a software tool that will integrate into the business's (SunView Software) machines either on the client or server.

SOFTWARE INTERFACES
WATT's algorithm is written in C# using a .NET library and PyTorch is used for the machine learning portion of WATT. WATT will integrate into the Windows operating system to utilize the resources i.e. page faults, low CPU usage, high memory usage to monitor potential thrashing. Based on monitoring and polling, WATT will send an alert if potential thrashing will occur.

TEST CASES

Test Case ID	1.0	**Test Priority**	Low
Test Case Description	Track system resource usage i.e. page faults, physical memory usage, and CPU usage.		
Pre-Requisite	A running operating system	**Post-Requisite**	A CSV (Excel) file with all the data

Steps	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comments**
1	Run the C# program	.NET diagnostic libraries	Consumed system resources (hard page			

Test Case ID	2.0		**Test Priority**	High		
Test Case Description	Measure total hard faults.					
Pre-Requisite	A running operating system and RAM usage is above 85%		**Post-Requisite**	Hard page faults increase		
Steps	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comments**
1	Run the C# program	.NET diagnostic libraries	Hard page faults			

Test Case ID	3.0		**Test Priority**	Low		
Test Case Description	Package statistics in a CSV format and integrated it with an API.					
Pre-Requisite	A running operating system		**Post-Requisite**	Data stored in a CSV format		
Steps	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comments**
1	Run the C# program	System resources (hard page faults, CPU usage, RAM usage, etc.)	Successful integration with business's (SunView Software) Web platform			

Test Case ID	4.0		**Test Priority**	Medium		
Test Case Description	Interaction between Python machine learning model and C# polling and monitoring function.					
Pre-Requisite	A running operating system		**Post-Requisite**	Binary yes or no		
Steps	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comments**
1	Run the C# program	Test vectors from polling function (CSV data)	Binary yes or no			

Test Case ID	5.0		**Test Priority**	Medium		
Test Case Description	Send a warning or alert when signs of thrashing are occurring on the system.					
Pre-Requisite	A running operating system		**Post-Requisite**	Dialog box yes or no		
Steps	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comments**
1	Run the C# program	Binary yes or no	Dialog box yes or no			