INSTITUTE SECURITY MANAGEMENT

TEST REPORT

Version *<1.0>*

<04/04/2016>

VERSION HISTORY

[Provide information on how the development and distribution of the Test Report was controlled and tracked. Use the table below to provide the version number, the author implementing the version, the date of the version, the name of the person approving the version, the date that particular version was approved, and a brief description of the reason for creating the revised version.]

Version #	Implemented By	Revision Date
1.0	12	<05/04/16>

Note to the Author

*[This document is a template of a **Test Report** document for a project. The template includes instructions to the author, example text, and fields that should be replaced with the values specific to the project.*

- *Blue italicized text enclosed in square brackets ([text]) provides instructions to the document author, or describes the intent, assumptions and context for content included in this document.*
- *Blue italicized text enclosed in angle brackets (<text>) indicates a field that should be replaced with information specific to a particular project.*
- *Text and tables in black are provided as examples of wording and formats that may be used or modified as appropriate to a specific project. These are offered only as suggestions to assist in developing project documents; they are not mandatory formats.*

When using this template for your project document, it is recommended that you follow these steps:

1. *Replace all text enclosed in angle brackets (e.g., <Project Name>) with the correct field values. These angle brackets appear in both the body of the document and in headers and footers.*
2. *Modify example text as appropriate to the specific project.*
3. *To add any new sections to the document, ensure that the appropriate header and body text styles are maintained. Styles used for the Section Headings are Heading 1, Heading 2 and Heading 3. Style used for boilerplate text is Body Text.*
4. *To update the Table of Contents, right-click and select "Update field" and choose the option- "Update entire table"*
5. *Before submission of the first draft of this document, delete this "Notes to the Author" page and all instructions to the author, which appear throughout the document as blue italicized text enclosed in square brackets.]*

Table of Contents

1.0 INTRODUCTION

1.1 PURPOSE

This Institute Security Management Test Report provides a summary of the results of test performed as outlined within this document.

2.0 TEST PLAN

The Framework used for testing is GoogleTest Framework developed by google for testing C/C++ projects. This framework is open source and is hosted here.

The tests performed here are in competence with the test provided in document itself. The google test project repository is cloned into the source root of project under test. An external build file (Makefile) is developed to reduce the redundant work of compiling whole project from scratch, thus it records what are the changes that we've done in code and compiles only the modified part of the project.

The test written include unit testing, as the code is simple we did manual as well as automatic testing on the software. Input for most test cases were manually feed into a file or were written in test_ism.cc file. The output then was compared to expected results. Also source code was checked for any coding errors.

test_ism.cc contains all the unit test with proper assertions and test cases.

3.0 TEST ASSESSMENT

Test assessment was thoroughly done. Source code was traversed manually to search for any errors present. Most of the test cases are checked for errors. Furthermore test cases can be tested for expected result to get all failure cases.

4.0 TEST RESULTS

[Summarize the test results. Include a detailed description of any deviations from the original test plan, design, test case, or expected results. Include any issues or bugs discovered during the test.]

4.1 UNIT/MODULE/SYSTEM TESTING

Unit, module, and system integration testing activities were performed during the development of the system build or release.

Test Case ID	Date Tested	Tester	Pass/Fail	Severity of Defect [Low/ Medium/ High]	Summary of Defect	Closed prior to Production Release?	Comments

1	3/4/16	Anmol	Pass	Medium	GuardPositioning.canSetNumericGuardPosition	<Yes> or <No>	
2	3/4/16	Anmol	fail	Medium	GuardPositioning.canSetProperNumericGuardPosition		
3	3/4/16	Anmol	fail	Medium	GuardPositioning.cannotSetAlphabeticGuardPosition		
4	3/4/16	Parag	Pass	medium	VehicleDatabaseEntry.setVehicleSlot		
5	3/4/16	parag	Pass	Medium	VehicleDatabaseExit.getFreeSlotAfterSetting		
6	3/4/16	parag	Fail	Medium	VehicleDatabaseExit.getFreeSlotBeforeSetting		Negative values of vehicles
7	3/4/16	Narender	pass	Medium	VehicleStatus.checkStatus		
8	3/4/16	Narender	Pass	Medium	VehicleMarked.checkIfMarked		
9	3/4/16	Narender	Pass	Medium	VehicleMarked.checkIfUnmarked		
10	3/4/16	Indraneel	Pass	medium	VehicleColor.checkColoredProperly		
11	3/4/16	Indraneel	Pass	Medium	VehicleDriverName.checkDriverName		
12	3/4/16	Indraneel	Pass	Medium	VehicleNumber.checkValidRegistrationNumber		
13	3/4/16	Indraneel	Fail	Medium	VehicleNumber.checkInValidRegistrationNumber		

4.2 SYSTEM TESTING

The table below summarizes the results of system testing:

Test Case ID	Date Tested	Tester	Pass/Fail	Severity of Defect [Low/ Medium / High]	Summary of Defect	Closed prior to Production Release?	Comments
1	29/03/16	Anmol	Pass	High	No defect		Vehicle entry
2	29/03/16	Narender	Fail	High	Exception handling on input was not used		Vehicle entry
3	30/03/16	Anmol	Pass	High	No defect		Vehicle exit
4	30/03/16	Parag	Fail	High	Out of bound error not handled for slot no.		Vehicle exit
5	3/04/16	Anmol	Fail	Medium	Include emergency condition within entry and exit		Vehicle entry/exit
5	4/04/16	Parag	pass	low	No defect		Guard allocation
6	4/04/16	Narender	Pass	Medium	No defect		Guard summary
7	4/04/16	Indraneel	Absent	Medium	Feature not implemented		Solution to guard absence
8	4/04/16	Indraneel	Absent	Medium	Feature not implemented		Emergency

4.3 USER ACCEPTANCE TESTING

The table below summarizes the test cases employed for user acceptance testing and the test results obtained for each test case:

Test Case ID	Date Tested	Tester	Pass/ Fail	Severity of Defect [Low/ Medium/ High]	Summary of Defect	Closed prior to Production Release?	Comments
1	30/03/16	Anmol	Fail	High	No error message for long input	<Yes> or <No>	At the time of vehicle entry
2	30/03/16	Indraneel	Fail	Low	Format for vehicle registration number not specified		Valid format should be shown to user
3	3/04/16	Narender	Fail	medium	No try catch implemented for exception handling		While taking string inputs

[If the test case failed, list the corresponding Test Incident ID in the Comments column.]

4.4 REGRESSION TESTING

The table below summarizes the test cases employed for regression testing and the test results obtained for each test case:

Test Case ID	Date Tested	Tester	Pass/Fail	Severity of Defect [Low/ Medium/ High]	Summary of Defect	Closed prior to Production Release?	Comments

						<Yes> or <No>	

[If the test case failed, list the corresponding Test Incident ID in the Comments column.]

4.5 PERFORMANCE TESTING

The table below summarizes the test cases employed for performance testing and the test results obtained for each test case:

Test Case ID	Date Tested	Tester	Pass/Fail	Severity of Defect [Low/ Medium/ High]	Summary of Defect	Closed prior to Production Release?	Comments
1	3/04/16	Anmol	pass	high	Time complexity of program very fast	<Yes> or <No>	Repose time of program is good
2	3/04/16	Parag	Fail	high	Huge database		Program can handle only 100 entries no dynamic database for handling huge dataset
3	3/04/16	Anmol	Fail	High	Password saved in plain text In source code also no hiding is used while entering it		Reveals confidential data

[If the test case failed, list the corresponding Test Incident ID in the Comments column.]

4.6 <TYPE OF TEST>

The table below summarizes the test cases employed for <type of test (e.g., unit/ module/ interface testing)> and the test results obtained for each test case:

Test Case ID	Date Tested	Tester	Pass/Fail	Severity of Defect [Low/ Medium / High]	Summary of Defect	Closed prior to Production Release? *[This will be filled by the dev team.]*	Comments
1	3/4/16	narender	Fail	High	Different set of improper inputs	<Yes> or <No>	No of vehicles inside becomes negative
2	3/4/16	parag	Fail	Medium	Whitespaces in input not handled		Registration number format not defined
3	3/4/16	Anmol	Fail	Low	Very large integer input not handled		Exceptions not handled while taking input

5.0 VARIANCES

- Mock testing of classes was not done because of improper project structure.
- GUI Testing could not be done due to non-implementation of GUI.
- Dataflow testing couldn't be done because of implementation of main driver program which was tested automatically by input redirection of input test files using input redirection on command line.
- No database management system was implemented so as to perform performance testing. Instead arrays were used for vehicle database which can be replaced by much more efficient data structures.

6.0 TEST INSTANCES

- Number of vehicles went into negative digits.
- No exception handling.
- No definition for entryReq() , ExitReq().

6.1 RESOLVED TEST INCIDENTS

[Identify all resolved test incidents and summarize their resolutions. Reference may be made to Test Incident Reports that describe in detail the unexpected results, problems, or defects reported during testing, along with their documented resolutions, which may be included as an appendix to this document.]

[This will be filled by the dev team.]

6.2 UNRESOLVED TEST INCIDENTS

[Identify all unresolved test incidents and provide a plan of action for their resolution. Reference may be made to Test Incident Reports that describe in detail the unexpected results, problems, or defects reported during testing, which may be included as an appendix to this document.]

[This will be filled by the dev team.]

7.0 RECOMMENDATIONS

- Interaction with program is command line based with menu driven as numeric input, instead a proper GUI can be used to drive the software.
- Program should contain multiple files with function declaration, definition, and driver program in different files. Generally the classes are defined using "virtual" modifier to ease testing. Program must be modular for ease of understanding and modification.
- Three global arrays of size 100 are declared at the start of the source code. Declaring global variables is not a good practice it should be avoided.
- No constructor is coded for the vehicle database this results in garbage values in database.
- Error while entering input are not handled properly. If we input a string instead of an integer the program fails.
- Instead of using arrays vectors can be used for vehicle database so as to let the database accommodate more than 100 data.

APPENDIX A: REFERENCES

The following table summarizes the documents referenced in this document.

Document Name	Version	Description
GoogleTest	Version 1.7.0	GoogleTest is a C/C++ testing framework
Makefile	Version 1.0	This is the build file used to generate the executable file from test_ism.cc
Input*	Version 1.0	Input test cases for command line testing

APPENDIX B: KEY TERMS

The following table provides definitions for terms relevant to this document.

Term	Definition
ISM	*The Project Under Test. Stands for Institute Security Management*
GT	*Google Test*
GM	*Google Mock, The mock classes part of GT framework*