System Functional Test Plan

for

Knight

Table of Contents

1. Introduction

1.1. Test Objectives

The system test of the Knight-app should validate requirements like:

- The basic functionality of the application works without problems
- No problems with the GUI display
- No data display problems

1.2. *Scope of Testing*

The system testing will include functional, GUI testing.

1.3. System Overview

The task of the program is to find all variants of the board with the knight covering as many cells as possible from a given position.

Principle of the program: choose branches and check them.

1.4. References

- System specification Document
- Test standards

2. Approach

2.1. Constraints

Three days might not be enough time to test the entire system and then retest the system to find new defects due to fixes

2.2. Coverage

Test coverage will be measured by a matrix of testable requirements and test cases.

2.3. Software Components

System requirements weren't included in the Specification Document.

2.4. Test Type

The following types of testing will be performed during system functional testing:

- Functional testing, by performing test cases based on personal understanding of how the system should work
- GUI testing, based on a personal understanding of how the interface works
- Performance testing based on computational severity and impact on CPU load

3. Features to be Tested

3.1. Options:

- Changing view mode checker condition
- "Show" checkers work
- "Write" checkers work
- Changing lines num
- Changing cell size

3.2. *Selecting the starting position of the figure*

3.3. *Calculations. Calculation time; Affect on CPU load:*

- Calculations start
- Ability to stop calculations
- Ability to interact with another functions during calculations
- Choosing different number of decisions
- How calculations do affect on CPU loading
- Time that calculation will take

3.4. *Collapse / uncollapse the app:*

- Use the collapse function on system panel
- Trying to use Alt+Tab
- Use the collapse icon on app system panel

3.5. Graphic view of the calculations:

- Displaying calculations with numbers and a figure

4. Features Not to be Tested

4.1. Security of files that branches uses

5. Testing Procedures

5.1. Testing Execution

5.1.1. Test cases

For each system feature to be tested, the tester will execute a set of pre-defined test cases. If the observed results are equal to the expected results, a checkmark is placed in the “pass” column. If the observed results are not equal to the expected results, a checkmark is placed in the “fail” column.

5.2. Pass/Fail Criteria

To pass the system functional test, the following criteria must be met:

- Calculating function is working correctly
- All menu’s functions is working correctly
- The interface is displayed correctly with any type of interaction
- Input boundary values doesn’t broke functional of the app

5.3. Defect Management

The assignment and description of defect severity levels will be as follows:

- **Blocker** - Completion of test cases are impacted.
- **High** - Defects which prove to be detrimental to the system. Testing should not progress to the next build until corrective measures have been taken.
- **Medium** - Defects which provide invalid/incorrect information.
- **Low** - Defects are esthetic in nature. Functionality is NOT impacted.

6. Appendix

Bug reports list - List of bugs found in the process
Description.docx - Application Description