SE 3XA3: Test Report
Ratava

Team 9, Makiam Group
Aidan McPhelim mcpheima
Alexie McDonald mcdona16
Illya Pilipenko pilipeni

December 6, 2018

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
2018-09-19	0.0	Initial draft created
2018-09-03	0.1	Team info updated

This document provides a report of the testing done on the Ratava project.

1 Functional Requirements Evaluation

We had to review some of our functional requirements, but the rest were satisfied by our manual and automated testing.

2 Nonfunctional Requirements Evaluation

We had to review some of our non-functional requirements, but the rest were satisfied by our manual and automated testing.

2.1 Usability

The program was tested for usability by having users who have previously had no interaction with the system use it. The metrics were very informal, but provided feedback for improving the UI. In the end, the program was deemed user-friendly.

2.2 Performance

The program takes very little computational resources to run, and so all of the performance metrics were satisfied. The program was deemed to succeed well enough against its performance expectations that no data was collected.

3 Unit Testing

*__NOTE:__ not all unit tests are shown here due to redundancy.

Test:	Hash list returned is correct
Input:	-
Output:	Available hash list
Expected:	Available hash list
Result:	PASS

Test:	Hash object corresponds with name
Input:	Hash name
Output:	Corresponding hash object
Expected:	Corresponding hash object
Result:	PASS

Test:	Hash generates same output with same input
Input:	Test string
Output:	Two identical hash values
Expected:	Two identical hash values
Result:	PASS

Test:	Hex value properly split into colours
Input:	Hex corresponding to the colour white
Output:	White colour
Expected:	White colour
Result:	PASS

Test:	Hash value is split into colours
Input:	Hash value
Output:	List of colours
Expected:	List of colours
Result:	PASS

Test:	Hash is used to find random int in range
Input:	Hash value
Output:	Int in range
Expected:	Int in range
Result:	PASS

Test:	Template pixel array corresponds to stored pixel array
Input:	Template choice
Output:	Exact array of pixels corresponding to the template
Expected:	Exact array of pixels corresponding to the template
Result:	PASS

Test:	Output image file type is .jpg
Input:	Hash string input
Output:	.jpg image file
Expected:	.jpg image file
Result:	PASS

Test:	Image file name is the name that was specified
Input:	test
Output:	test.jpg
Expected:	test.jpg
Result:	PASS

4 Changes Due to Testing

5 Automated Testing

All of our unit testing for the GenerateHash, UseHash, TemplateDraw, and GraphicsDraw was automated.

6 Trace to Requirements

7 Trace to Modules

8 Code Coverage Metrics

For test coverage, we made an automated unit test case for every function that was able to be automated. For the rest, we did manual system testing.