Disaster Reactor

Software Testing Document
CSC 4330

Team Members:
Amanda Alfaro, Craig Jones, Khaleel Harris,
Sam Shrestha, Zachary Smith, Nealan Vettivelu

Table of Contents_Toc449477873

1 Revision History

Name	Date	Reason For Changes	Version
Sam Shrestha	4/25/2016	Completed 1st Draft	0.5.0
Amanda Alfaro	4/25/2016	Updated Approach Section	0.5.1
Amanda Alfaro	4/25/2016	Updated Summary Section	0.5.1
Nealan Vettivelu	4/25/2016	Added Test Cases for Simulation and Print	0.6.0
W. Craig Jones	4/25/2016	Added Input Test Cases	0.6.5
Khaleel Harris	4/25/2016	Added Test Cases for GUI	0.7.0
Nealan Vettivelu	4/26/2016	Added Non Functional Requirements	0.7.1
Sam Shrestha	4/26/2016	Updated Overall Approach	0.7.3
W. Craig Jones	4/26/2016	Added Output Test Cases	0.8.0
W. Craig Jones	4/26/2016	Updated Test Cases Layout	0.8.1
Zachary Smith	4/26/2016	Updated Test Cases for GUI	0.8.5
Amanda Alfaro	4/26/2016	Added Deliverables Section	0.9.0
Khaleel Harris	4/26/2016	Removed Technical References to Code	0.9.1
W. Craig Jones	4/26/2016	Formatting and Final Version	1.0.0

2 Test Plan Identifier

The plan defined in this document will be referred to as the Disaster Reactor Test Plan. It is the Master test plan upon which all test cases involved in Disaster Reactor are listed. When we talk about test planning, we are referring to this document.

This document is to be kept dynamic as the implementation phase is ongoing, should there be a new feature added or a feature deleted, the appropriate test cases are generated and added to this document or located and deleted from this document.

3 Introduction

3.1 Purpose

This document will describe the scope, approach, resources, and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, testing task assignments, and any risks requiring contingency planning.

3.2 Summary

Disaster Reactor will be tested on multiple items and features to provide a complete and functional program. The Software Testing Document is divided into four testing sections with various subsections. The sections of the Software Testing Document are:

3.2.1 Graphical User Interface

- Modes: This section provides testing on switching between inspection mode and Paint mode in the application
- Load Map Button: This section provides testing on the "Load Map" button
- Paint Mode: This section provides testing on the feature that allows user to paint Damage, Difficulty, and Value on the map.
- Inspect Mode: This section provides testing on the functions of Inspect mode including: Clear Spawn Points, Increase Radius, Create Spawn Points, Decrease Radius, and Sample Position
- Run Sim: This section provides testing on the "Run Sim" button

3.2.2 System Features

- Input Stage: This section provides testing on the input functions of the system such as the two modes of the system: Paint and Inspect Mode
- Load Map: This section provides testing on the "Load Map" feature which allows the user to load a map via the "Load Map" button of their selected terrain
- Simulation Stage: This section provides testing on the Simulation Stage where the simulation runs based off of the user inputs
- Output Stage: This section provides testing on the "Print" feature which allows the user to print the output to a pdf file

3.2.3 Nonfunctional Features

- Performance tests: This section will provide testing and target metrics for the software’s performance.
- Safety tests: This section will provide outline and provide all safety routines which the software will conform to.
- Software Quality tests: This section will provide testable standards which the software will conform to which relate to inherent user quality.

3.3 References

- Disaster Reactor Software Requirement Specification Document
    - Version 1.1, Last Updated on 2016-04-25
- Disaster Reactor Software Design Document
    - Version 1.1, Last Updated on 2016-04-25

4 Graphical User Interface Tests

Test Case ID:	GUI-Test-1
Test Item:	Load Map Button
Test Priority:	High
Pass/Fail Criteria:	If the correct button is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Press the 'Load Map' button
Expected Results:	A file dialog should appear that allows the user to load a map into the program
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	GUI-Test-2
Test Item:	Create Spawn Point Key
Test Priority:	High
Pass/Fail Criteria:	If the correct key is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure map has been loaded into program 2. Ensure the cursor is hovered over the simulation area. 3. Press the 'S' hotkey on keyboard
Expected Results:	Visual image for agent spawn point should appear at the position of the user's cursor.
Special Procedural Requirements:	N/A
Intercase Dependencies:	Input-Test-0

Test Case ID:	GUI-Test-3
Test Item:	Increase Radius Key
Test Priority:	High
Pass/Fail Criteria:	If the correct key is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure that the map has been loaded into the program. 2. Ensure the cursor is hovered over the simulation area. 3. Press the "R" hotkey on keyboard.
Expected Results:	Radius of the circle that represents the desired paint area increases.
Special Procedural Requirements:	N/A
Intercase Dependencies:	Input-Test-0

Test Case ID:	GUI-Test-4
Test Item:	Sample Position Key
Test Priority:	High
Pass/Fail Criteria:	If the correct key is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure map has been loaded into program 2. Ensure the cursor is hovered over the simulation area 3. Press the 'Space' hotkey on keyboard
Expected Results:	RGB Color of sampled position should appear in the bottom-left corner of the Inspect Mode Panel
Special Procedural Requirements:	N/A
Intercase Dependencies:	Input-Test-0

Test Case ID:	GUI-Test-5
Test Item:	Decrease Radius Key
Test Priority:	High
Pass/Fail Criteria:	If the correct key is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure that the map has been loaded into the program. 2. Ensure the cursor is hovered over the simulation area. 3. Press the "T" hotkey on keyboard.
Expected Results:	Radius of the circle that represents the desired paint area decreases.
Special Procedural Requirements:	N/A
Intercase Dependencies:	Input-Test-0

Test Case ID:	GUI-Test-6
Test Item:	Inspect Mode Key
Test Priority:	High
Pass/Fail Criteria:	If the correct key is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure that the map has been loaded into the program. 2. Press the "I" hotkey on keyboard.
Expected Results:	GUI will switch to inspect mode.
Special Procedural Requirements:	N/A
Intercase Dependencies:	Input-Test-0

Test Case ID:	GUI-Test-7
Test Item:	Paint Damage Radio Button
Test Priority:	High
Pass/Fail Criteria:	If the correct radio button is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure map has been loaded into program. 2. Click the Radio Button to the left of the "Damage (Red)" text.
Expected Results:	Color of user's cursor should change to red.
Special Procedural Requirements:	N/A
Intercase Dependencies:	Input-Test-0

Test Case ID:	GUI-Test-8
Test Item:	Paint Difficulty Radio Button
Test Priority:	High
Pass/Fail Criteria:	If the correct radio button is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure map has been loaded into program. 2. Click the Radio Button to the left of the "Difficulty (Blue)" text.
Expected Results:	Color of user's cursor should change to blue.
Special Procedural Requirements:	N/A
Intercase Dependencies:	Input-Test-0

Test Case ID:	GUI-Test-9
Test Item:	Paint Value Radio Button
Test Priority:	High
Pass/Fail Criteria:	If the correct radio button is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure map has been loaded into program. 2. Click the Radio Button to the left of the "Value (Green)" text.
Expected Results:	Color of user's cursor should change to green.
Special Procedural Requirements:	N/A
Intercase Dependencies:	Input-Test-0

Test Case ID:	GUI-Test-10
Test Item:	Run Sim Button
Test Priority:	High
Pass/Fail Criteria:	If the correct button is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure that the map has been loaded into the program. 2. Press the "Run Sim" button.
Expected Results:	Radius of the circle that represents the desired paint area decreases.
Special Procedural Requirements:	N/A
Intercase Dependencies:	Input-Test-0

Test Case ID:	GUI-Test-11
Test Item:	Toggle Agents Button
Test Priority:	High
Pass/Fail Criteria:	If the correct toggle button is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure 'Run Sim' button has been pressed. 2. Click the Toggle Button to the left of the "Toggle Agents" text.
Expected Results:	Agents in simulation area should become invisible if they are visible and vice versa.
Special Procedural Requirements:	N/A
Intercase Dependencies:	GUI-Test-10

Test Case ID:	GUI-Test-12
Test Item:	Toggle Spawns Button
Test Priority:	High
Pass/Fail Criteria:	If the correct toggle button is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure 'Run Sim' button has been pressed. 2. Click the Toggle Button to the left of the "Toggle Spawns" text.
Expected Results:	Agent spawn points in the simulation area should become invisible if they are visible and vice versa.
Special Procedural Requirements:	N/A
Intercase Dependencies:	GUI-Test-10

Test Case ID:	GUI-Test-13
Test Item:	Toggle Environment Button
Test Priority:	High
Pass/Fail Criteria:	If the correct toggle button is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure 'Run Sim' button has been pressed. 2. Click the Toggle Button to the left of the "Toggle Environment" text.
Expected Results:	Map in simulation area should become invisible if it is visible and vice versa.
Special Procedural Requirements:	N/A
Intercase Dependencies:	GUI-Test-10

Test Case ID:	GUI-Test-14
Test Item:	Toggle Trails Button
Test Priority:	High
Pass/Fail Criteria:	If the correct toggle button is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure 'Run Sim' button has been pressed. 2. Click the Toggle Button to the left of the "Toggle Trails" text.
Expected Results:	Agent trails in simulation area should become invisible if they are visible and vice versa.
Special Procedural Requirements:	N/A
Intercase Dependencies:	GUI-Test-10

Test Case ID:	GUI-Test-15
Test Item:	Print Button
Test Priority:	High
Pass/Fail Criteria:	If the correct button is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure that the map has been loaded into the program. 2. Ensure 'Run Sim' button has been pressed. 3. Press the "Print" button.
Expected Results:	A print PDF function should be called.
Special Procedural Requirements:	N/A
Intercase Dependencies:	Input-Test-0, GUI-Test-10

Test Case ID:	GUI-Test-16
Test Item:	Quit Button
Test Priority:	High
Pass/Fail Criteria:	If the correct button is registered as input, the test passes. Otherwise, the test fails.
Test Steps:	1. Ensure that the map has been loaded into the program. 2. Ensure 'Run Sim' button has been pressed. 3. Press the "Quit" button.
Expected Results:	The program should stop and close.
Special Procedural Requirements:	N/A
Intercase Dependencies:	Input-Test-0, GUI-Test-10

5 System Features Tests

5.1 Input Tests

Test Case ID:	Input-Test-0
Test Item:	Load Map
Test Priority:	High
Pass/Fail Criteria:	The user is able to load an OSM file into the application.
Preconditions:	N/A
Test Steps:	1. Click the "Load Map" button on the Input Stage of the application. 2. Navigate to an OSM file and select it. 3. Click ok.
Expected Results:	The map should load and the simulation pane should be populated with a moderate difficulty in all areas that do not have roads.
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Input-Test-1
Test Item:	Paint Damage
Test Priority:	High
Pass/Fail Criteria:	The user is able to increase the damage component of an area centered around the cursor.
Preconditions:	Input-Test-0
Test Steps:	1. Select the "Damage" radio button. 2. Hold left click on the simulation pane and move the mouse.
Expected Results:	The areas of the simulation pane under the cursor have their damage component increased, as represented by the red component of the image.
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Input-Test-2
Test Item:	Paint Value
Test Priority:	High
Pass/Fail Criteria:	The user is able to increase the value component of an area centered around the cursor.
Preconditions:	Input-Test-0
Test Steps:	1. Select the "Value" radio button. 2. Hold left click on the simulation pane and move the mouse.
Expected Results:	The areas of the simulation pane under the cursor have their value component increased, as represented by the green component of the image.
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Input-Test-3
Test Item:	Paint Difficulty
Test Priority:	Medium
Pass/Fail Criteria:	The user is able to increase the difficulty component of an area centered around the cursor.
Preconditions:	Input-Test-0
Test Steps:	1. Select the "Difficulty" radio button. 2. Hold left click on the simulation pane and move the mouse.
Expected Results:	The areas of the simulation pane under the cursor have their difficulty component increased, as represented by the blue component of the image.
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Input-Test-4
Test Item:	Create Spawn Point
Test Priority:	High
Pass/Fail Criteria:	The user is able to add spawn points by hovering over a location and pressing S on the keyboard.
Preconditions:	None
Test Steps:	1. Hover over an area in the simulation pane and press the "S" key.
Expected Results:	A spawn point should be created at that location.
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Input-Test-5
Test Item:	Clear Spawn Points
Test Priority:	Low
Pass/Fail Criteria:	The user is able to remove all set spawn points by pressing C on the keyboard.
Preconditions:	Input-Test-4
Test Steps:	1. Press the "C" key
Expected Results:	All spawn points are removed.
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Input-Test-6
Test Item:	Increase Cursor Radius
Test Priority:	Low
Pass/Fail Criteria:	The user is able to increase the radius of the cursor by pressing the R key on the keyboard.
Preconditions:	
Test Steps:	1. Press the "R" key while the cursor is over the Simulation Pane.
Expected Results:	The radius of the cursor increases.
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

5.2 Simulation Tests

Test Case ID:	Simulation-Test-1
Test Item:	Calculate Aid
Test Priority:	High
Pass/Fail Criteria:	Aid calculation passes if a float returns that has incorporated the Trails as well as the environmental data Fails if the calculation does not.
Preconditions:	Gui-Test-3
Test Steps:	1. Pass in predefined value with an expected value of 0,255 and 122.5 2. Confirm that values exist at these points 3. Pass in values with an expected value of 255.1 and -0.1 4. Clamps values to ensure inside the correct range
Expected Results:	Float between the value of 0 and 255
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Simulation-Test-2
Test Item:	Calculate Repair
Test Priority:	High
Pass/Fail Criteria:	Repair calculation passes if a float returns that has incorporated the Trails as well as the environmental data Fails if the calculation does not return.
Preconditions:	Gui-Test-3
Test Steps:	1. Pass in predefined value with an expected value of 0,255 and 122.5 2. Confirm that values exist at these points 3. Pass in values with an expected value of 255.1 and -0.1 4. Clamps values to ensure inside the correct range
Expected Results:	Float between the value of 0 and 255
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Simulation-Test-3
Test Item:	Reading Damage
Test Priority:	High
Pass/Fail Criteria:	Successfully read the Damage value in a pixel
Preconditions:	Input-Test-1
Test Steps:	1. Create a blank map 2. Add damage to a single point 3. Create agent 4. Run simulation and output the damage reading
Expected Results:	Return red value from environment pixel
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Simulation-Test-4
Test Item:	Reading Value
Test Priority:	High
Pass/Fail Criteria:	Successfully read the Value value in a pixel
Preconditions:	Input-Test-2
Test Steps:	1. Create a blank map 2. Add value to a single point 3. Create agent 4. Run simulation and output the value reading when agent is on top of the value coloring
Expected Results:	Return green value from environment pixel
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Simulation-Test-5
Test Item:	Reading Difficulty
Test Priority:	High
Pass/Fail Criteria:	Successfully read the Difficulty value in a pixel
Preconditions:	Input-Test-3
Test Steps:	1. Create a blank map 2. Add difficulty to a single point 3. Create agent 4. Run simulation and output the difficulty reading when agent is on top of the difficulty coloring
Expected Results:	Return blue value from environment pixel
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Simulation-Test-6
Test Item:	Calculate L1 Vector
Test Priority:	High
Pass/Fail Criteria:	Calculate L1 Vector passes if a vector2f is returned that is dependent on each of the pixels within its region
Preconditions:	
Test Steps:	5. Create a blank map 6. Add damage to a single point 7. Add spawn points surrounding it 8. Run simulation
Expected Results:	All agents should converge in the center of the damage point
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Simulation-Test-7
Test Item:	Calculate L0 Vector
Test Priority:	High
Pass/Fail Criteria:	Repair calculation passes if a float returns that has incorporated the Trails as well as the environmental data Fails if the calculation does not return.
Preconditions:	Simulation-Test-4
Test Steps:	1. Create a blank map 2. Add damage to a single point 3. Add spawn points surrounding it 4. Run simulation
Expected Results:	All agents should converge in the center of the damage point
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Simulation-Test-8
Test Item:	Calculate Path
Test Priority:	High
Pass/Fail Criteria:	Pass if a path is returned that is determined by comparing local optimal and global optimal.
Preconditions:	Simulation-Test-5
Test Steps:	1. Create a blank map 2. Add damage to a single point 3. Add spawn points surrounding it 4. Run simulation
Expected Results:	All agents should converge in the center of the damage point
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

5.3 Output Tests

Test Case ID:	Output-Test-1
Test Item:	PDF Printing
Test Priority:	Medium
Pass/Fail Criteria:	The application can write to a pdf file with images and descriptions provided.
Preconditions:	Input-Test-0
Test Steps:	1. Left Click "Print" located on the bottom left of the application window
Expected Results:	File called output.pdf will be created with the following images in the following order: 1. Initial Conditions, Environment and Spawn 2. Showing Environment, Trails, Spawns and Agents 3. Showing Environment, Trails, and Spawns 4. Showing Environment and Trails 5. Showing Trails 6. Showing Position Heatmap 7. Showing Repair Heatmap 8. Showing Aid Heatmap
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Output-Test-2
Test Item:	PDF output
Test Priority:	Medium
Pass/Fail Criteria:	All 8 images are successfully saved to a PDF document called output.pdf
Preconditions:	Simulation-Test-1
Test Steps:	1. Left Click “Print” located on the bottom left of the application window
Expected Results:	File called output.pdf will be created with the following images in the following order: 1. Initial Conditions, Environment and Spawn 2. Showing Environment, Trails, Spawns and Agents 3. Showing Environment, Trails, and Spawns 4. Showing Environment and Trails 5. Showing Trails 6. Showing Position Heatmap 7. Showing Repair Heatmap 8. Showing Aid Heatmap
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

6 Non Functional Tests

Test Case ID:	Non-Fuctional-Test-1
Test Item:	Performance
Test Priority:	Medium
Pass/Fail Criteria:	Maintain 5 frames per second at Windows 7+, 3.0GHz Quad Core, 16GB RAM
Preconditions:	Simulation-Test-5
Test Steps:	1. Run Simulation for 10 seconds 2. Find difference in number of frames from when started and finished
Expected Results:	Number of frames should have increased by 50
Special Procedural Requirements:	N/A
Intercase Dependencies:	N/A

Test Case ID:	Non-Fuctional-Test-2
Test Item:	Quality
Test Priority:	Medium
Pass/Fail Criteria:	Client is satisfied with the product
Preconditions:	All
Test Steps:	1. Provide client with the software 2. Incorporate feedback loop for next development of software
Expected Results:	Client should be more satisfied with new iteration of software
Special Procedural Requirements:	Client must be on-site whilst the application is in beta to ensure that they will be able to point out features directly to the developers

7 Features Not to Be Tested

Due to our limited resources, at this time we are unable to test the accuracy of the paths determined by our application. This feature will be tested thoroughly before any official releases of the application.

8 Approach

8.1 Overall Approach to Testing

First, the testers will test the functions of the GUI. If the GUI is not functioning properly, the testers will inform all parties responsible for the GUI via Facebook messenger. Once the GUI has passed all tests, the testers will test the features of the Input Stage. If any of these features are not functioning properly, the testers will inform all parties responsible for the Input Stage via Facebook messenger. Once the Input Stage has passed all tests, the testers will test the Simulation Stage. If the Simulation Stage fails the tests, the testers with inform all parties responsible for the Simulation Stage via Facebook messenger. Once the Simulation Stage passes all tests, the testers will test the Output Stage. If the Output Stage fails any tests, the testers will inform all parties responsible for the Output Stage via Facebook messenger. Once the Output Stage passes all tests, testing will be complete for this version of Disaster Reactor.
See section 11 below for responsible parties

8.2 Four Major Features

- GUI – Each tester will go through the interface to make sure 1. each button properly works 2. button does the actions it's supposed to.
- Input – The tester assigned to this will go through each specific input to work on the input test cases and will then report outcome, whether it has errors or not, to the group.
- Simulation – The tester assigned to this will load a map into the system, provide input, and run the simulation to make sure the system as a whole works properly.
- Output – The tester assigned to this will ensure that a PDF file is properly created.

8.3 More Information on the Specs of Testing

There are no special techniques that we need to complete our tests. The only tool needed to run the tests is a computer with a Windows 7 or later operating system.

The minimum degree of comprehensiveness is that each test case is tested a total of 2 times. Each test uses black box testing, where we will test valid and invalid inputs leading to a pass/fail test result therefore there are no special techniques needed.

In order for a test criteria to be completed it must meet the demands of our updated Software Requirements Specification document. As we stated before, the technique to be used to trace the requirements will be to reference our SRS document to ensure features work as stated. The deadline for all testing to be complete is the day the code for Disaster Reactor project in CSC 4330 at Louisiana State University is due on April 29th.

9 Suspension Criteria and Resumption Requirements

The criteria for suspending testing activity is as follows:

- the feature is not completed for testing. This can be due to features targeted for a future version of the product or focus areas due to time constraints.
- A critical system feature that other test cases in rely on fails testing
    - Testing will resume upon correction of the error

10 Test Deliverables

The following will be produced as the result of testing. Testing will complete when the following deliverables have been produced:

10.1 Acceptance test plan

Acceptance testing is to be conducted after the output paths can be tested for accuracy. The goal of this prototype is to introduce the product and generate interest in it.

10.2 System and Integration test plan

The integration test will produce pdf files displaying a series of output maps.

10.3 Unit Test Plans and Turnover Documentation

Unit tests will be conducted on an individual basis, individuals responsible for each specific unit is located in Section 11.

10.4 Report Mock-ups

Disaster Reactor will not generate mockup reports instead we will use our testing logs to produce any reports we may have.

10.5 Defect and Incident Reports

These reports will be reported to the team members. We are all equally responsible for addressing these incidents until the issue is resolved.

10.6 Test Logging and Reporting

All of out test passes and fails will be logged in Section 4 where we list the pass/fail criteria, test steps, and expected results. From there our logs will be reported to all the team members.

11 Testing Tasks

In order to prepare for and perform testing, a tester must have access to Windows 7 OS or later versions, and map data on which to perform testing. There are no other requirements or necessary preparations to begin testing Disaster Reactor
There are no task interdependencies nor any special skills required to begin testing on Disaster Reactor.

12 Environmental Needs

12.1 Physical

- Access to a computer with a working mouse and keyboard
- Access to a computer with Windows 7 or later OS
- Program installed on computer
- Map data in .osm format
- Facebook messenger for communications
- GitHub

13 Responsibilities

As outlined in the *Disaster Reactor Software Design Document*, Sam Shrestha and Amanda Alfaro are responsible for the testing and quality assurance of Disaster Reactor. Any bugs found by Sam and Amanda will be immediately reported to the team members according to their responsibilities which are as follows:

- Bug fixes pertaining to the GUI:
  - Khaleel Harris and Zachary Smith
- Bug fixes pertaining to the Simulation:
  - W. Craig Jones and Nealan Vettivelu
- Bug fixes required for the Input and Output Stages:
  - W. Craig Jones, Khaleel Harris, and Zachary Smith

Once testers complete multiple tests and log them they will report back to the group to update section 4 of our Software Testing Document.
All team members are responsible for getting and maintaining access to any and all environment needs, as these resources are readily available for students across the campus of Louisiana State University.

14 Staffing and Training Needs

There are no special staffing or training needs as one of the main goals of Disaster Reactor is to be made to be used by persons of any skill level. There is a minimum knowledge of operation and using a computer required to run Disaster Reactor.

There is a user manual and video tutorial included with the software. This will be sufficient for training users to run the program.

15 Schedule

15.1 Artificial Intelligence Testing

- Start Date: 4/26/2016
- End Date: 4/27/2016

15.2 Graphical Interface Testing

- Start Date: 4/26/2016
- End Date: 4/27/2016

15.3 Integration Testing

- Start Date: 4/27/2016
- End Date: 4/28/2016

15.4 System Testing

- Start Date: 4/27/2016
- End Date: 4/28/2016

15.5 Milestones

15.5.1 MS1

- Date: 4/27/2016
- Description: Completion of the Graphical User Interface

15.5.2 MS2

- Date: 4/28/2016
- Description: Completion of System Integration and Testing of the Graphical User Interface and the System features

15.5.3 Alpha Build

- Date: 4/28/2016
- Description: Completion of System and Integration testing

15.5.4 Beta Build

- Date: To be determined once product can be tested for path accuracy
- Description: Conclusion of performance and safety testing and documentation to come.

15.5.5 Release

- Date: To be determined once product can be tested for path accuracy
- Description: All testing will be concluded. Documentation and deliverables will be finalized and packaged.

16 Risks and Contingencies

16.1 Risk of Time

The biggest risks for the project is the amount of time the team has to complete the project and to then test it. The team has an advantage with being the maximum size of 6. Should the time run out of time to complete the implementation or testing by certain dates, functionality of various features will be removed. The necessary features will be completed and tested first. No attempt will be made to bypass any part of the review and testing processes. To reiterate, the base functionality of the program will be implemented and tested before continuing with additional features.

17 Approvals

To be filled out upon completion of testing. Approval is granted for the final build by each developer prior to release.

Name	Date	Signature of Approval
Amanda Alfaro		
Khaleel Harris		
W. Craig Jones		
Sam Shrestha		
Zachary Smith		
Nealan Vettivelu		