***Testing and Inspection Report of Escape Rooms***

**

***A Sample Document for   
Generating Consistent Professional Reports***

***Prepared by***

***Nemil Shah, Wishy Parikh, Ruchit Patel and David Qiao***

**Table of Contents**

[I Project Description 4](#_Toc7277711)

[1 Project Overview 4](#_Toc7277712)

[2 Project Domain 4](#_Toc7277713)

[3 Relationship to Other Documents 5](#_Toc7277714)

[4 Naming Conventions and Definitions 5](#_Toc7277715)

[4a Definitions of Key Terms 5](#_Toc7277716)

[4b UML and Other Notation Used in This Document 6](#_Toc7277717)

[4c Data Dictionary for Any Included Models 6](#_Toc7277718)

[II Testing 7](#_Toc7277719)

[5 Items to be Tested 7](#_Toc7277720)

[6 Test Specifications 7](#_Toc7277721)

[7 Test Results 9](#_Toc7277722)

[8 Regression Testing 10](#_Toc7277723)

[III Inspection 10](#_Toc7277724)

[9 Items to be Inspected 10](#_Toc7277725)

[10 Inspection Procedures 10](#_Toc7277726)

[11 Inspection Results 11](#_Toc7277727)

[IV Recommendations and Conclusions 12](#_Toc7277728)

[V Project Issues 12](#_Toc7277729)

[12 Open Issues 12](#_Toc7277730)

[13 Waiting Room 13](#_Toc7277731)

[14 Ideas for Solutions 13](#_Toc7277732)

[15 Project Retrospective 14](#_Toc7277733)

[VI Glossary 15](#_Toc7277734)

[VII References / Bibliography 15](#_Toc7277735)

[VIII Index 16](#_Toc7277736)

# Project Description

## Project Overview

Escape room is puzzle game containing multiple rooms in the house. The goal of the player is to find the right key by solving puzzles, gathering useful tools in the inventory. There is hint or caption in every room located in the top of the screen. User can use this caption to get the right tool in wisely way to achieve the goal.

Language: JAVA

IDE: Eclipse

User Interface: Mouse Clicking.

## Project Domain

The domain of the project is associated with multiple JAVA files. Multiple Java files have been divided into four packages each:

1. Escape Room: Creation of World and pop up the main menu.
2. Multiple Rooms: Design and structure of each rooms.
3. Resource: Images and sounds to create the rooms relevant to the story defined.
4. Images: To create the image by converting image to a java object and other necessary manipulations.

Testing will be done in both ways:

1. Testing Individual Package: For example, testing only he package name as Multiple-Rooms
2. Testing the Entire project: For example, to test the entire project in whole. To make sure the correctness of functionality of all java files residing in each individual package.

New rooms, new images, and new sounds are the key domain to update the game.

## Relationship to Other Documents

The game planning contained multiple stages like pre development, development and post development stages.

1. Pre development: It mainly had the creation of story and script. We needed the good script for the user to navigate the real understanding of the game. Credits: Nemil Shah and Wishy Parikh
2. Development: It mainly had the creation of java files and packages to merge the story. Proper functionality, requirements and design goals were necessary part of this.

Credit: All

1. Post Development: To create the images on paint, edit, crop and filter images to make the visual appearance much better and intriguing. Sounds were added accurately to augment the feel of the game. Credit: David and Ruchit.

Merging all documents was very challenging task. For example, to create the Graphical User Interface java screen with mouse clicking as user interface to match the script of the story was the real goal.

Testing was done in two ways:

1. Test the functionality: Fix the bugs, errors, improve and optimize the java files residing in each package.
2. Test the story: Making sure the design of rooms was matching with the scripted story that was initialized in the pre development stage.

## Naming Conventions and Definitions

### Definitions of Key Terms

Names of packages and java files will let the client know what purpose that following files hold.

For example, file name ‘Room1.java’ specifies that this file has design and functionality available for Room-1.

When objects are created for images, their names have been consistent with the image names.

For example, dog image file, ‘dog.png’ is name with Buffuredim Dog;. Similarly, it is also for sounds too.

The separate variables are created as Boolean for all the objects which are clicked during the play of the game. For example, when the user clicks on dog, a variable name as ‘dogClicked’ is created.

Naming conventions are consistent in all the files whether it is variable name, file name, or package name.

### UML and Other Notation Used in This Document

For the graphical screen, we have used java swing components.

Graphical screen can be swapped with new screen when the user leaves from room 1 to room 2. This is being done with CardLayout style.

### Data Dictionary for Any Included Models

Data structures for each room has been created to maintain the functionality of each room during the game.

Each room contains the collection of images in the form of ‘list’ ( Data Structures). All images have separate data structured.

List of panel holder is created in file name ‘ImageInventoryPanel’ to control the inventory.

# Testing

## Items to be Tested

Rooms needs to be tested as the location of the click is determined in the rooms.

Images needs to be tested as the object in the image has to be in the same location as the click.

Sound needs to be tested as the sound clip plays at the right time of the click. And not overlap.

Main menu panel with the music.

## Test Specifications

**1 - Rooms**

**Description:** Rooms that contain objects and click points.

**Items covered by this test:** Organization of the room.

**Requirements addressed by this test:** Images should be added in the list according to the background. One by one.

**Environmental needs:** All the object that are used needs to be saved in the list.

**Intercase Dependencies:** Frame and panel test that comes in designing the main menu screen.

**Test Procedures:** Test by running it if the images were added in the same order.

**Output Specifications:** Perfectly lined up image as the room. For example dog in front and back wall in the background should not overlap. Dog should be seen on front.

**Pass/Fail Criteria:** Wrong overlapping image is a fail and perfect room is a pass.

**2 - Images**

**Description:** Images in the right location.

**Items covered by this test:** Organization of the room.

**Requirements addressed by this test:** Images needs to be of right size.

**Environmental needs:** Images needs to be in .png format. The background of the image needs to be transparent as we cant see white spaces in the backround wall.

**Intercase Dependencies:** Background of the room needs to be set up before creating images.

**Test Procedures:** Test by running it.

**Input Specifications:** Image file names needs to be same as in the code.

**Pass/Fail Criteria:** Wrong location of the image is a fail.

**3 - Sounds**

**Description:** Sound that plays on clicks and in background.

**Items covered by this test:** Audio of the rooms.

**Requirements addressed by this test:** Sound should play one by one without overlapping at the start

**Environmental needs:** Sound needs to be in the wav format.

**Intercase Dependencies:** Background of the room needs to be done before adding the sound.

**Test Procedures:** Test by running it

**Output Specifications:** Perfectly lined up sound the plays after ending of the previous sound

**Pass/Fail Criteria:** Wrong overlapping sound is a fail and perfect sound is a pass.

**4 – Main menu**

**Description:** General backbone of the code as the frame and panel is decided.

**Items covered by this test:** Organization of the room.

**Requirements addressed by this test:** Panel and frame size.

**Environmental needs:** java panel formats.

**Intercase Dependencies:** This is the first step of the testing.

**Test Procedures:** Test by running it if the images were added in the same order.

**Output Specifications:** Perfectly lined up image as the room. For example dog in front and back wall in the background should not overlap. Dog should be seen on front.

## Test Results

**1 - Rooms**

**Date(s) of Execution:** 12 April

**Staff conducting tests:** Wishy and Nemil

**Expected Results:** Clicks in the wrong place.

**Actual Results:** Figured out the clicks and places the coordinates.

**Test Status:** Generally pass.

**2 - Images**

**Date(s) of Execution:** 13 April

**Staff conducting tests:** Wishy and Nemil

**Expected Results:** Images in the wrong place.

**Actual Results:** Figured out the location so perfect location of image was obtained.

**Test Status:** Generally pass.

**3 - Sound**

**Date(s) of Execution:** 15 April

**Staff conducting tests:** Ruchit and David

**Expected Results:** Overlapping sound

**Actual Results:** Overlapping sound

**Test Status:** Made it work with sleeping the thread so sound does not overlap. Generally Pass.

**4 – Main menu**

**Date(s) of Execution:** 9 April

**Staff conducting tests:** Ruchit and David

**Expected Results:** panel not sized right.

**Actual Results:** perfect panel and frame size.

**Test Status:** Generally pass.

## Regression Testing

Images required testing again and again as the images did not line up correctly so we had to made the images in paint 3D to manually draw them and set the size.

Images needed to be background transparent so overlapping would not affect the background image.

Multiple tests were ran to shift the images and set them to the right location of the clicks.

Sound also required testing as overlapping sound and delayed sound was the failed test case.

# Inspection

## Items to be Inspected

* Images of room inspected while decorating room.
* Selecting objects are inspected for room, because clicking on object should not take to many clicking.
* While choosing back ground music. Must be song when right click.
* Choose java panel for Main menu.

## Inspection Procedures

* **Image of Rooms:**

Inspection by: Ruchit and David

Procedures: Make sure its meet requirements of rooms and project.

* **Objects of Rooms:**

Inspection by: Ruchit and David

Procedures: Check with rooms images and also have to crop images as need.

* **Back ground sound:**

Inspected by: Nemil and Wishy

Procedures: Check before putting inside program and its suit with rooms and another object.

* **Java panel for main menu:**

Inspected by: Nemil and Wishy

Procedures: Check java panel with color and frame size.

## Inspection Results

**Images of Rooms:**

**Date(s) of Execution:** 12 April

**Inspection by:** Ruchit and David

**Result of Inspection:** Some change pictures.

**Objects of Rooms:**

**Date(s) of Execution:** 13 April

**Inspection by:** Ruchit and David

**Result of Inspection:** Crop images and change selected objects.

**Background sounds:**

**Date(s) of Execution:** 14 April

**Inspection by:** Nemil and Wishy

**Result of Inspection:** Almost all pass in inspection, nothing much have to change.

**Java panel for main menu:**

**Date(s) of Execution:** 10 April

**Inspection by:** Nemil and Wishy

**Result of Inspection:** Mustchange couple of panels.

# Recommendations and Conclusions

In this project besides above we did many functions inspect and tested. Specifically, input and out put of functions.

# Project Issues

## Open Issues

Thoughts of adding the game to other devices, such as mobile or mac. Planning and implementing our code for such format is not complete. It is not a priority, but is a possible option.

Would VR (virtual reality) be an option? However, the first step would have to be recreating our game but in a 3D space. Both are options for the future, but no actual research has been done yet.

## Waiting Room

Updating the images used in game. Maybe the use of higher quality images and other graphical changes.

Multiple levels of difficulty could be added. Each with a unique room, and more complex puzzles. This would also create the need for more images, sound effects, and music.

Fleshing out our main menu to include multiple levels of graphics. This also includes creating higher end graphics.

## Ideas for Solutions

For upgrading graphics, switching from Java to Unity might allow more room for expansion. A more complex system could also allow for 3D gameplay.

## Project Retrospective

The use of Java created an easy environment to work with. It was simple and effective, however, limits our game's potential. Upgrading does not even require switching languages, we could probably start by using a different package of Java other than Swing.

The use of Paint3D was a clever and inventive idea for creating images, and will be a nice free option for future productions. This also removes the need for using copyrighted imagery, lessening issues to worry about.

# Glossary

So far, this coding project was wonderful. Everyone did best effort to finish project. From this project everyone learns how to work in team. How to get help or able to give help to any team member.

# References / Bibliography

|  |  |
| --- | --- |
| [1] | Robertson and Robertson, Mastering the Requirements Process. |
| [2] | A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013. |
| [3] | J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012. |
| [4] | M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004. |

# Index

**No index entries found.**