**System Test Plan**

**For**

**<****Renju Game System >**

Written by:

陈俊哲 19301141

白远希 19301139

周光彪 19301136

邱 潼 19301155

范星宇 19301145

潘炎灵 19301067

Date: 2022.06.21

Table of Contents

[1. Introduction 2](#_Toc106788338)

[1.1 Purpose 2](#_Toc106788339)

[1.2 Objectives 2](#_Toc106788340)

[2. Functional Scope 2](#_Toc106788341)

[2.1 Login and Register 3](#_Toc106788342)

[2.2 Local Battle Model 3](#_Toc106788343)

[2.3 Man vs. Machine 3](#_Toc106788344)

[2.4 Finished game playback 4](#_Toc106788345)

[3. Overall Strategy and Approach 4](#_Toc106788346)

[3.1 Testing Strategy 4](#_Toc106788347)

[3.2 System Testing Entrance Criteria 4](#_Toc106788348)

[3.3 Testing Types 5](#_Toc106788349)

[3.4 Suspension Criteria and Resumption Requirements 5](#_Toc106788350)

[3.5 Test Data 6](#_Toc106788351)

[4. Execution Plan 10](#_Toc106788352)

[4.1 Execution Plan 10](#_Toc106788353)

[4.2 Test Cases 10](#_Toc106788354)

[5. Defect Reporting 30](#_Toc106788355)

[5.1 Defect Tracking 30](#_Toc106788356)

[5.2 Defect Reporting and Reports 30](#_Toc106788357)

[5.3 Defect Management Process 31](#_Toc106788358)

[5.4 Defect Severity Definitions 31](#_Toc106788359)

[6. Environment 32](#_Toc106788360)

[6.1 Hardware Environment 32](#_Toc106788361)

[6.2 Software Environment 32](#_Toc106788362)

[7. Test Schedule 32](#_Toc106788363)

[8. Assumptions 33](#_Toc106788364)

[9. Risks and Contingencies 34](#_Toc106788365)

[9.1 Demand risk 34](#_Toc106788366)

[9.2 Test case risk 34](#_Toc106788367)

[9.3 Code quality risk 34](#_Toc106788368)

[9.4 Test environment risk 34](#_Toc106788369)

[10. Appendices 34](#_Toc106788370)

# Introduction

## Purpose

This document is a test plan for Renju Game System Testing, produced by the System Testing team. It describes the testing strategy and approach to testing the team will use to verify that the application meets the established requirements of the business prior to release.

## Objectives

* Meets the requirements, specifications and the Business rules.

The specification of this test submission document is:

1. Management document, document status: completed and approved. Document number shall be indicated. The document contains the following types: plan, summary, report, meeting minutes, memos, applications, etc. All documents must be stored in the repository and the corresponding VSS library. The documents put into the knowledge base shall be put into the Department Assistant uniformly, and the documents must be approved and approved.

2. Technical documents, document status: completed, approved and approved. Document number shall be indicated. The documents include the following types: design documents, requirements documents, test design documents, interface prototype software, user manual, installation manual, technical white paper, source code, software products, etc. All documents must be stored in the repository and the corresponding VSS library. The documents put into the knowledge base shall be put into the Department Assistant uniformly, and the documents must be approved and approved.

* Supports the intended business functions and achieves the required software standards.

The expected business function of this test is to realize the Gobang Game system, which should realize login and registration, local games, human machine strategies, winning judgment, game storage, game replay, game reload, leaderboard. All functional and non functional requirements in the product specification should be realized.

* Satisfies the Entrance Criteria for User Acceptance Testing.

1) Functional requirements must be available.

2) Application code should be fully developed.

3) Unit test, integration test and system test should be completed.

4) There are no obstacles, high or medium defects in the system integration testing phase.

5) Regression test should be completed without major defects.

6) The traceability matrix of all tests shall be completed.

# Functional Scope

The Modules in the scope of testing for the Gobang Game System Testing are mentioned in the document attached in the following path:

## Login and Register

### Login

User can click “登录” button to login his account to start game, and participate in the score ranking.

### Register

If the user doesn’t have an account, he can click “注册” button to get his account.

## Local Battle Model

### Start New Game

User can click “双人对战” button to create new game. If there is a unfinished game stored in the game, the system will ask user whether continue the game.

### Place Chess Pieces

During the game, two players can place pieces one by one, and the color of the pieces will black and white alternate. Click the chessboard, the pieces will place on correct place. The system do not use Internet, so two players should use one computer and mouse to play it.

### Winning Judgment

Every time user places a piece on the board, system will judge whether the game is done. Winning of Renju game is that:

Black plays first, and players alternate in placing a stone of their color on an empty intersection. The winner is the first player to get an unbroken row of five stones horizontally, vertically, or diagonally.

So, the system will check pieces around the piece placed just now and judge whether the game is done.

## Man vs. Machine

### Start New Game

User can click “人机对战” button to create new game. If there is a unfinished game stored in the game, the system will ask user whether continue the game.

### Place Chess Pieces

During the game, the player can place pieces with computer, and the color of the pieces will black and white alternate. Click the chessboard, the pieces will place on correct place. The system do not use Internet, so the player should use one computer and mouse to play it.

### Winning Judgment

Every time user places a piece on the board, system will judge whether the game is done. Winning of Renju game is that:

Black plays first, and players alternate in placing a stone of their color on an empty intersection. The winner is the first player to get an unbroken row of five stones horizontally, vertically, or diagonally.

So, the system will check pieces around the piece placed just now and judge whether the game is done.

## Finished game playback

If there are some finished game stored in the system, players can click” Demonstration” button to make last finished game playback.

# Overall Strategy and Approach

## Testing Strategy

Renju Game System Testing will include testing of all functionalities that are in scope (Refer Functional Scope Section) identified. System testing activities will include the testing of new functionalities, modified functionalities, screen level validations, work flows, functionality access, testing of internal & external interfaces.

## System Testing Entrance Criteria

In order to start system testing, certain requirement must be met for testing readiness. The readiness can be classified into:

* The developer has finished coding and completed the unit test.
* The functions specified in the requirements specification or submitted by developers have been realized.
* The basic flow of the system under test can go through, and the functions on the interface are realized, which are in line with the functions specified in the design documents.
* The test scope is consistent with the function of the requirement specification. If not, the requirement specification shall be updated and the change application form shall be provided.
* The requirements document is approved and the review document is submitted.

## Testing Types

|  |  |
| --- | --- |
| Testing Type | Test Content |
| Black-Box Test | Test samples with various critical points and capabilities are selected to test, and each test sample represents a category |
| White-Box Test | Ensure that all kinds of test samples are covered and the test scope is comprehensive |
| Unit Test | The person in charge of each module uses JUnit for unit test |
| Integration Test | All the modules are merged one by one and tested to ensure the normal use of the functional interfaces |
| System Test | The overall test of the system to ensure that the work is normal, performance standards |

### Usability Testing

User interface attributes, cosmetic presentation and content will be tested for accuracy and general usability. The goal of Usability Testing is to ensure that the User Interface is comfortable to use and provides the user with consistent and appropriate access and navigation through the functions of the application (e.g., access keys, consistent tab order, readable fonts etc.)

### Functional Testing

The objective of this test is to ensure that each element of the component meets the functional requirements of the business as outlined in the:

* Business / Functional Requirements
* Business rules or conditions
* Other functional documents produced during the course of the project i.e. resolution to issues/change requests/feedback

## Suspension Criteria and Resumption Requirements

This section will specify the criteria that will be used to suspend all or a portion of the testing activities on the items associated with this test plan.

### Suspension Criteria

Testing will be suspended if the incidents found will not allow further testing of the system/application under-test. If testing is halted, and changes are made to the hardware, software or database, it is up to the Testing Manager to determine whether the test plan will be re-executed or part of the plan will be re-executed.

* One third of the test cases cannot be executed;
* When the test achieves the milestone target (such as unit test), it needs to be adjusted;
* Accidental damage of hardware or objective environment;
* The test practice cannot meet the test plan

### Resumption Requirements

Resumption of testing will be possible when the functionality that caused the suspension of testing has been retested successfully.

* Debug and optimize the module that causes pause, and recover after passing the test;
* When the milestone is completed, the manager summarizes and arranges to enter the next stage;
* The manager quickly coordinates and solves the influence of objective factors, and recovers when ready;
* Change and optimize the test plan, and resume the test according to the new plan.

## Test Data

Test data requirements are drawn up based on the functional requirements that are due for testing. The testing team will identify test cases that can be grouped into test scenarios and detail the data required to complete the testing activities.

### Login and Register Module

|  |
| --- |
| Test Module 1: Login and Register Mo |
| Test Purpose:  Check whether the user can login or register an account; whether the user is prompted for wrong input in login registration; whether the system is entered after successful login;  Test Point1: Whether the user can register an account;  Test Point2: Whether the user's registered account is repeatable；  Test Point3: Whether the user is given a correct prompt when the input does not meet the requirements when logging in or registering;  Test Point4: Whether the system can feedback the error message after the user's login password is wrong;  Test Point5: Whether the user enters the main interface of the game after successfully entering the login information and clicking login. |
| Test Condition:  Describe the corresponding data clearly, including general value, boundary value and abnormal value. |
| Test Tools:  CUnit  Visual Studio 2019  .exe File |
| Test Process:  Click Register, enter the correct and appropriate password for your account, confirm your password by entering a different password and see if you are prompted with an error message.  Change to a consistent password, click Register and see if it jumps to the login screen. Enter the wrong format of account password, see if there is an error message.  Enter the correct format but not registered account, click Login and see if you are logged in and if there is a message indicating that you are not registered.  Enter the registered account number and incorrect password, see if you log in, and see if there is a message indicating wrong password.  Enter the correct account password and see if the login is successful. |
| Test Expectation:  Right Login and register and reasonable error alerts |

### Game Play Module

|  |
| --- |
| Test Module 2: Local Game Play Module |
| Test Purpose:  Check whether the user can start, play;  Test Point1: Whether the user can start a game;  Test Point2: Whether the user can put correct color piece on the checkerboard; |
| Test Condition:  Describe the corresponding data clearly, including general value, boundary value and abnormal value. |
| Test Tools:  CUnit  Visual Studio 2019  .exe File |
| Test Process:  The user clicks to start the game and drops the pieces separately to see if the color of each adjacent drop is different. |
| Test Expectation:  Right color and could drop . |

|  |
| --- |
| Test Module 3: Winning Judgment Module |
| Test Purpose:  Check that whether system can identify the game is over when there are 5 some color pieces in a line. Every condition should be checked in.  Test Point 1: An unbroken row of five stones horizontally.  Test Point 2: An unbroken row of five stones vertically.  Test Point 3: An unbroken row of five stones diagonally.  Test Point 4: An unbroken row of five stones near the boundary(Decision at the board boundary).  (Because the whole chessboard is full and the winner is not decided, it is not easy to realize, so it is not considered.) |
| Test Condition:  Describe the corresponding data clearly, including general value, boundary value and abnormal value. |
| Test Tools:  CUnit  Visual Studio 2019  .exe File |
| Test Process:  In this test module, we should create every condition that game should be over: An unbroken row of five stones horizontally, vertically or diagonally. We can use three test case to check it.  Finally, we use one test case to test whether the judgment near the chessboard boundary is valid. |
| Test Expectation:  System can identify the finish of the game and display which player is winning. |

|  |
| --- |
| Test Module 4: Reload and Replay Module |
| Test Purpose:  Check that whether system can store the unfinished when user wants to leave and store the game. When user wants to continue the game, system should reload the game from the file.  Check that whether system can store the finished when a game is over. The game can be queried in ranking list and show playback when user click on it.  Test Point 1: Whether the system can store unfinished game.  Test Point 2: Whether the system can reload unfinished game from file.  Test Point 3: Whether the system can store finished game in the rank file.  Test Point 4: Whether the system can reload and show playback of any finished game. |
| Test Condition:  Describe the corresponding data clearly, including general value, boundary value and abnormal value. |
| Test Tools:  CUnit  Visual Studio 2019  .exe File |
| Test Process:  Click Playback to see if the game is played back in full. Click Start game, drop but do not connect into 5 pieces, exit, click Reload and see if the game reverts to the game before you exited. |
| Test Expectation:  System can store and reload the information of unfinished and finished games. |

### Man vs. Machine Module

|  |
| --- |
| Test Module 5: Man vs. Machine Module |
| Test Purpose:  Check whether the machine will make a move in a reasonable position when the human makes a move against the machine  Test Point 1: Weather the time and step of the machine drop correcte  Test Point 2: Whether the color of the machine drop is correct.  Test Point 3: Whether the position of the machine is reasonable |
| Test Condition:  Describe the corresponding data clearly, including general value, boundary value and abnormal value. |
| Test Tools:  CUnit  Visual Studio 2019  .exe File |
| Test Process:  Click on the human-machine match, the human player will make a move and see if the machine makes a reasonable move. |
| Test Expectation:  Reasonable machine drop |

### Leaderboard

|  |
| --- |
| Test Module 6: Leaderboard Module |
| Test Purpose:  Check if the ranking is sorted correctly and if the user can reorder correctly after the game .  Test Point 1: Weather the leaderboard is sorted correctly  Test Point 2: Whether the user can reorder correctly after the game |
| Test Condition:  Describe the corresponding data clearly, including general value, boundary value and abnormal value. |
| Test Tools:  CUnit  Visual Studio 2019  .exe File |
| Test Process:  Complete a game, click on the leaderboard and see if the sorting is correct |
| Test Expectation:  The leaderboard is sorted correctly. |

# Execution Plan

## Execution Plan

The execution plan will detail the test cases to be executed. The Execution plan will be put together to ensure that all the requirements are covered. The execution plan will be designed to accommodate some changes if necessary, if testing is incomplete on any day. All the test cases of the projects under test in this release are arranged in a logical order depending upon their inter dependency.

## Test Cases

### Login and Register

#### Login-1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Scenario ID | | Login-1 | | | Test Case ID | | Login-1B | |  |
| Test Case  Description | | Login-Negative test  case | | | **Test Priority** | | High | |
| Pre-Requisite | | NA | | | **Post-Requisite** | | NA | |
| Test Execution Steps: | | | | | | | | | |
| S.No | **Action** | | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | | **Test Result** | **Test Comments** |
| 1 | Launch the application | | hit the application | home | home | Google Chrome | | Pass | Launch successfully |
| 2 | Enter invalid account number, any password and hit login button | | Id: admiredsfsd  Password: \*\*\*\*\* | The account id does not match any account. | The account id does not match any account. | Google Chrome | | Pass | Invalid login attempt |
| 3 | Enter valid account number, incorrect password and hit login button | | Id: admire  Password: \*\*\*\*\* | The password is incorrect | The password is incorrect | Google Chrome | | Pass | Invalid login attempt |
| 4 | Enter valid account number, incorrect verification code and hit login button | | Id: admire  Verification code :\*\*\*\*\* | The verification code is incorrect | The verification code is incorrect | Google Chrome | | Pass | Invalid login attempt |



#### Login-2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Scenario ID | | Login-2 | | | Test Case ID | | Login-2B | |  |
| Test Case  Description | | Login-Positive test  case | | | **Test Priority** | | High | |
| Pre-Requisite | | NA | | | **Post-Requisite** | | NA | |
| Test Execution Steps: | | | | | | | | | |
| S.No | **Action** | | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | | **Test Result** | **Test Comments** |
| 1 | Launch the application | | hit the application | home | home | Google Chrome | | Pass | Launch successfully |
| 2 | Enter valid account number, correct password and hit login button | | Id: admire  Password: \*\*\*\*\* | The account id and password match the account. | The account id and password match the account. | Google Chrome | | Pass | Valid login attempt |
| 3 | Enter valid account number, correct verification code and hit login button | | Id: admire  Verification code: \*\*\*\*\* | The verification code is correct. | The verification code is correct. | Google Chrome | | Pass | Valid login attempt |

#### Register-1

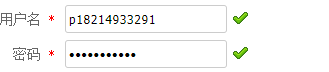
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Scenario ID | | Rsgister-1 | | | Test Case ID | | Register-1B | |  |
| Test Case  Description | | Login-Negative test  case | | | **Test Priority** | | High | |
| Pre-Requisite | | NA | | | **Post-Requisite** | | NA | |
| Test Execution Steps: | | | | | | | | | |
| S.No | **Action** | | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | | **Test Result** | **Test Comments** |
| 1 | Launch the application | | hit the application | home | home | Google Chrome | | Pass | Launch successfully |
| 2 | Enter invalid username with illegal character and hit register button | | Username: @@ | The username is invalid | The username is invalid | Google Chrome | | Pass | Invalid register attempt |
| 3 | Enter empty username and hit register button | | Username:/ | The username is empty | The username is empty | Google Chrome | | Pass | Invalid register attempt |
| 4 | Enter valid username, incorrect password and hit register button | | Username:  p18214933291  Password: \*\*\*\* | The password is invalid | The password is invalid | Google Chrome | | Pass | Invalid register attempt |





Register-2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | Rsgister-2 | | | **Test Case ID** | | Register-2B | |  |
| **Test Case**  **Description** | | Login-Positive test  case | | | **Test Priority** | | High | |
| **Pre-Requisite** | | NA | | | **Post-Requisite** | | NA | |
| Test Execution Steps: | | | | | | | | | |
| **S.No** | **Action** | | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | | **Test Result** | **Test Comments** |
| 1 | Launch the application | | hit the application | home | home | Google Chrome | | Pass | Launch successfully |
| 2 | Enter valid username, valid password and hit register button | | Username:  p18214933291  Password: \*\*\*\*\*\* | The username and password are valid | The username and password are valid | Google Chrome | | Pass | Valid register attempt |
| 3 | Enter valid phone number, valid verification code and hit register button | | Phone:  p18214933291  Password: \*\*\*\* | The phone and verification code are valid. | The phone and verification code are valid. | Google Chrome | | Pass | Valid register attempt |



### Retreat and Demonstration Test Case

#### Retreat

Time for Retreat:new game; gaming; over game;

New Games:

Test Scenario: Retreat-1

Test Description: retreat after new game

Test Priority: high

Pre-Requisite: NA

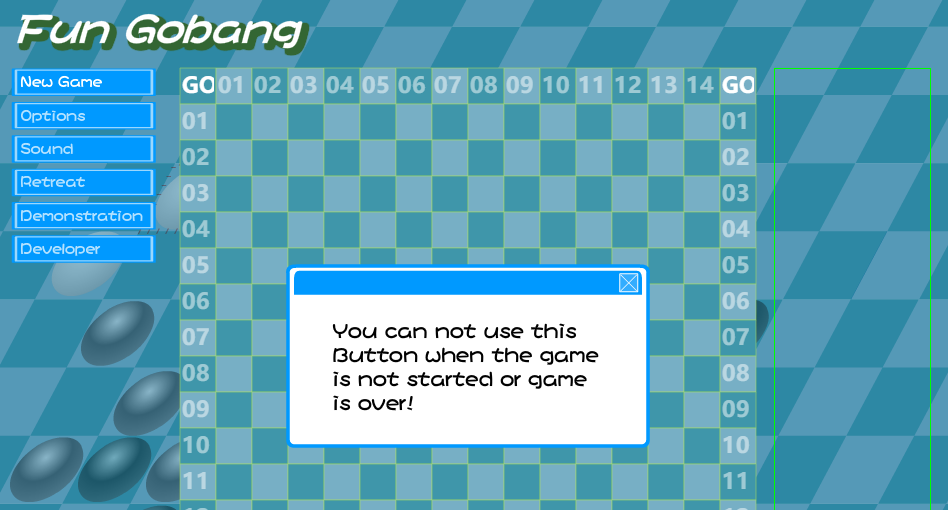
Post-Requisite: NA

Action: retreat

Inputs: Click the button of retreat

Expected Output: failure

Actual Output:



Test Result: pass

Gaming:

Test Scenario: Retreat-2

Test Description: retreat in game

Test Priority: high

Pre-Requisite: NA

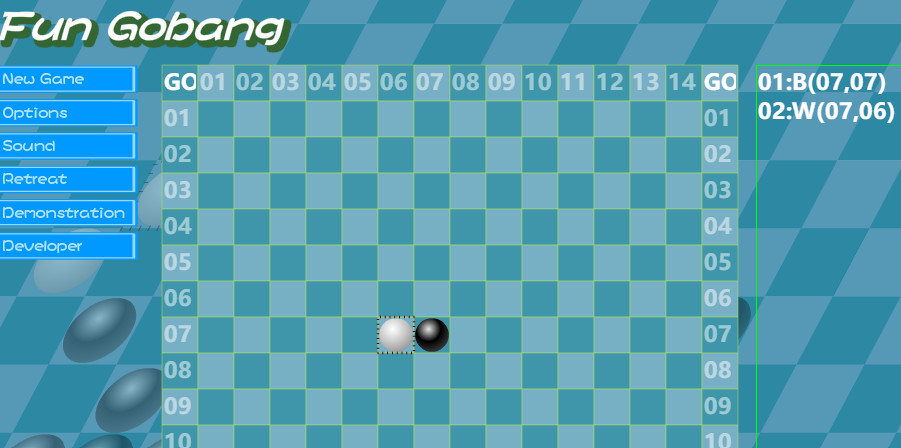
Post-Requisite: NA

Action: retreat

Inputs: Click the button of retreat

Expected Output: success

Actual Output:





Test Result: pass

Over Game:

Test Scenario: Retreat-3

Test Description: retreat after game

Test Priority: high

Pre-Requisite: NA

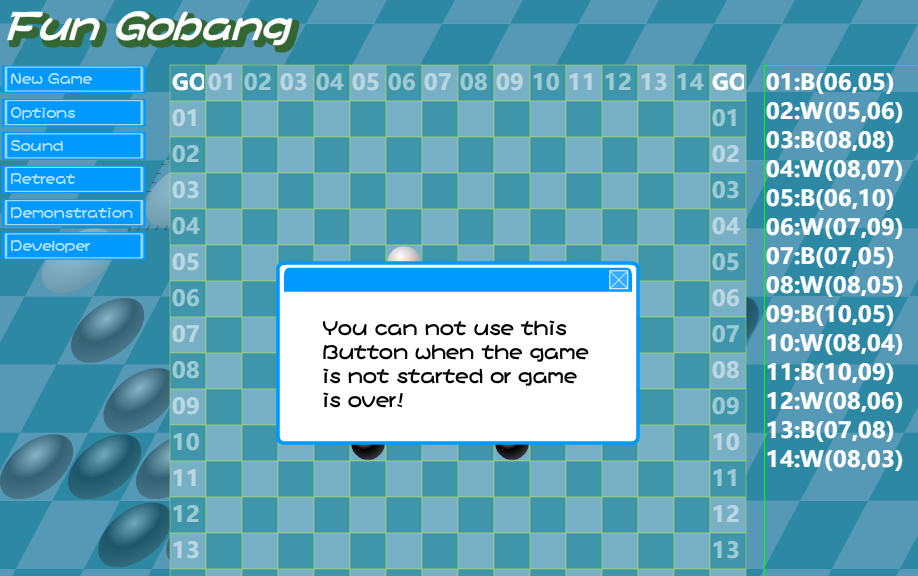
Post-Requisite: NA

Action: retreat

Inputs: Click the button of retreat

Expected Output: failure

Actual Output:



Test Result: pass

#### Demonstration

Time for Retreat:new game; gaming; over game;

New Game:

Test Scenario: Demonstration -1

Test Description: demonstration after new game

Test Priority: high

Pre-Requisite: NA

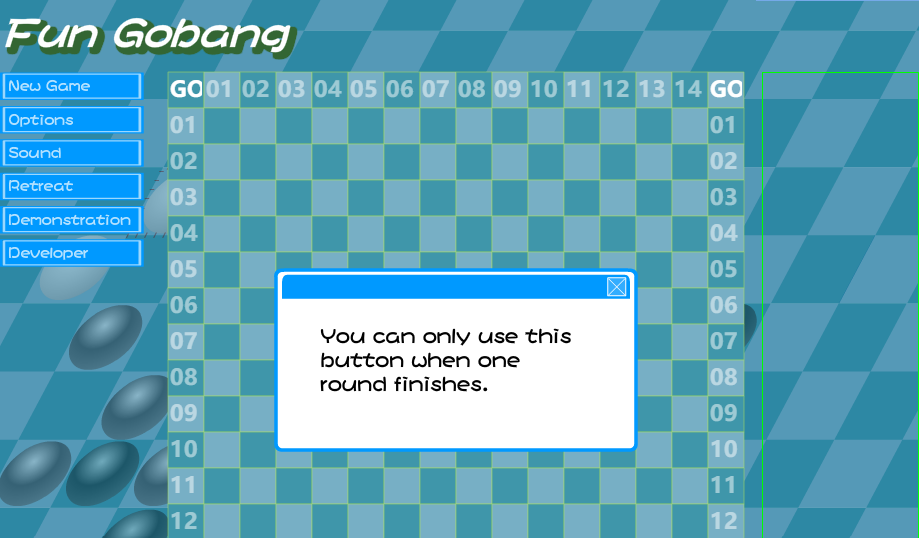
Post-Requisite: NA

Action: demonstration

Inputs: Click the button of demonstration

Expected Output: failure

Actual Output:



Test Result: pass

Gaming:

Test Scenario: Demonstration -2

Test Description: demonstration in game

Test Priority: high

Pre-Requisite: NA

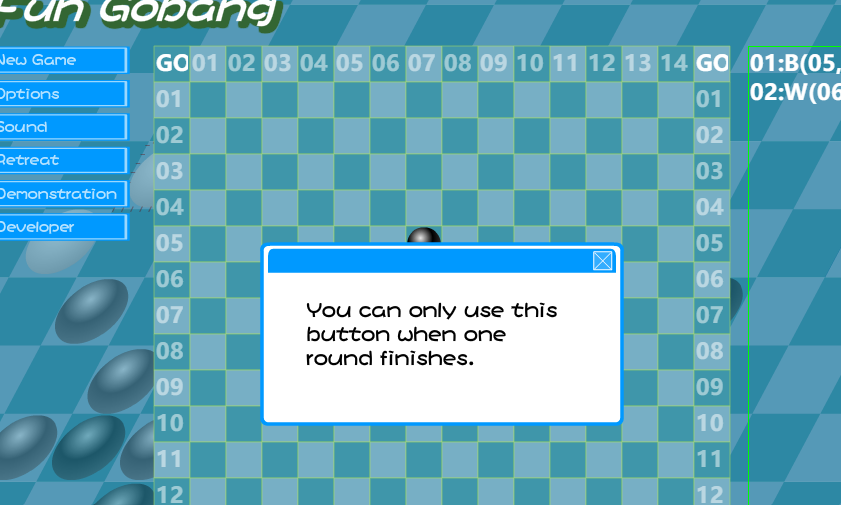
Post-Requisite: NA

Action: demonstration

Inputs: Click the button of demonstration

Expected Output: failure

Actual Output:



Test Result: pass

Over Game:

Test Scenario: Demonstration -3

Test Description: demonstration after game

Test Priority: high

Pre-Requisite: NA

Post-Requisite: NA

Action: demonstration

Inputs: Click the button of demonstration

Expected Output: success

Actual Output:

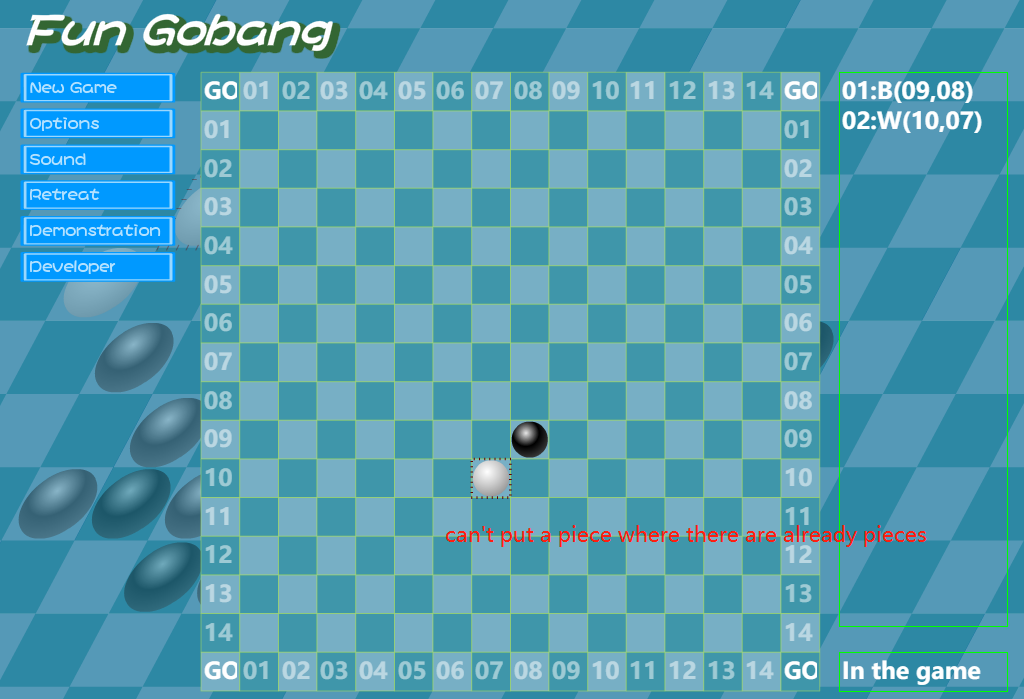
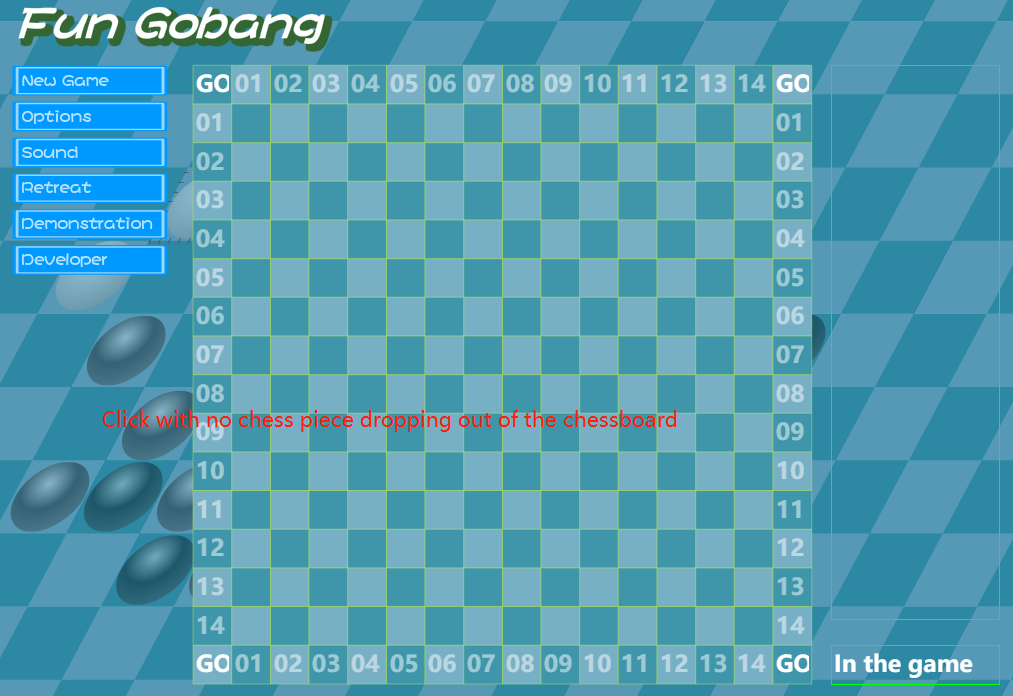
 

Test Result: pass

### Rule and Victory Judge Test Case

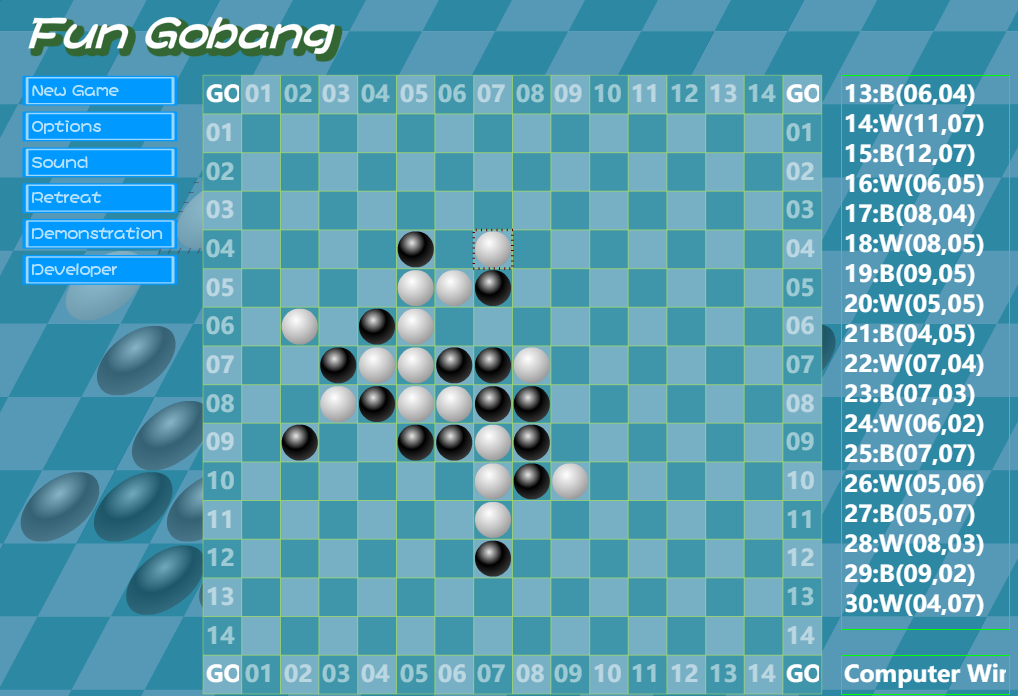
#### Rule Judge-1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Scenario ID | | Rule-1 | | | Test Case ID | | Rule-1B | |  |
| Test Case  Description | | Rule-Negative test  case | | | **Test Priority** | | High | |
| Pre-Requisite | | NA | | | **Post-Requisite** | | NA | |
| Test Execution Steps: | | | | | | | | | |
| S.No | **Action** | | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | | **Test Result** | **Test Comments** |
| 1 | Start a new game and choose the chess color | | Select balck | Black first;  White second; | Player go First | Google Chrome | | Pass | Choose color successful |
| 2 | Drop a piece out of the chessboard | | Click out of the chessboard | Nothing happened | Nothing happened | Google Chrome | | Pass | Player can’t drop the piece out of the chessboard |
| 3 | Drop the chess pieces continuously | | Click the chessboard before AI drop the chess piece | Can’t drop the chess piece | There are no more pieces on the chessboard | Google Chrome | | Pass | The game must be played by turns |
| 4 | put a piece where there are already pieces | | Click the chessboard where there are already pieces | Can’t drop the chess piece | There are no more pieces on the chessboard | Google Chrome | | Pass | Only place the chess pieces where there is space |

****

Rule Judge-2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Scenario ID | | Rule-2 | | | Test Case ID | | Rule-2B | |  |
| Test Case  Description | | Rule-Positive test  case | | | **Test Priority** | | High | |
| Pre-Requisite | | NA | | | **Post-Requisite** | | NA | |
| Test Execution Steps: | | | | | | | | | |
| S.No | **Action** | | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | | **Test Result** | **Test Comments** |
| 1 | Start a new game and choose the chess color | | Select balck | Black first;  White second; | Player go First | Google Chrome | | Pass | Choose color successful |
| 2 | Drop a piece in the chessboard | | Click the chessboard | A piece is placed on the chessboard | A piece is placed on the chessboard | Google Chrome | | Pass | Valid drop pieces attempt |
| 3 | Drop the chess pieces by turns | | Click the chessboard after AI drop the chess piece | drop the chess piece successfully | drop the chess piece successfully | Google Chrome | | Pass | Valid drop pieces attempt |
| 4 | put a piece where there are no pieces before | | Click the chessboard where there are no pieces | A new chess piece is dropped there | A new chess piece is dropped there | Google Chrome | | Pass | Valid drop pieces attempt |



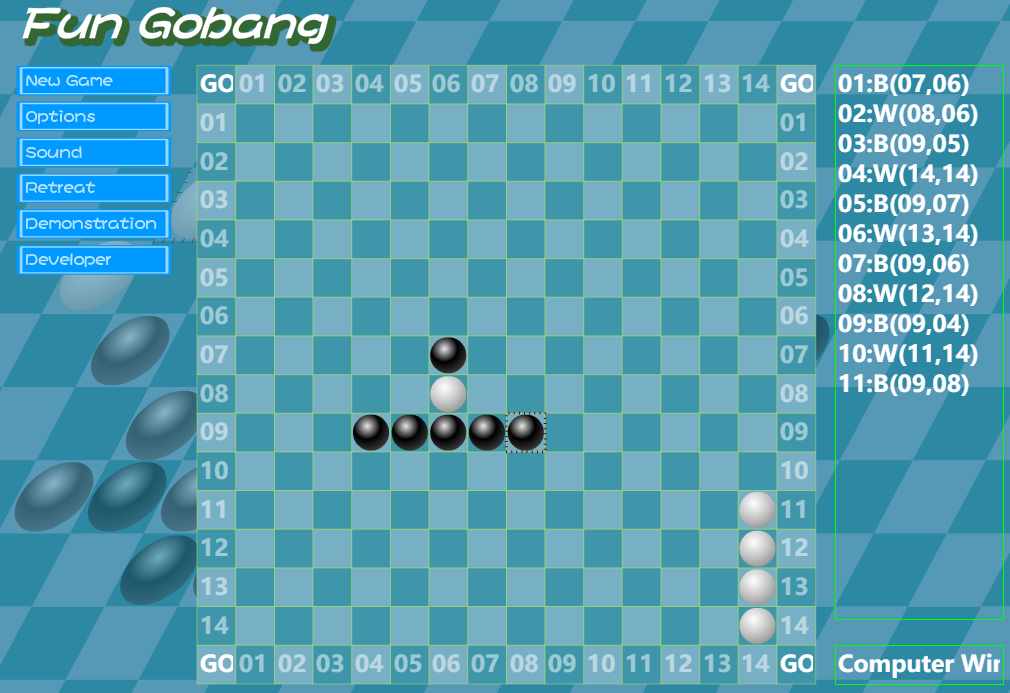
#### Victory Judge-1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Scenario ID | | victory-1 | | | Test Case ID | | victory-1B | |  |
| Test Case  Description | | Judge whether the system can confirm the victory in the vertical case | | | **Test Priority** | | High | |
| Pre-Requisite | | Start a new game | | | **Post-Requisite** | | NA | |
| Test Execution Steps: | | | | | | | | | |
| S.No | **Action** | | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | | **Test Result** | **Test Comments** |
| 1 | Start a new game | | Select white | Black first;  White second; | AI go First | Google Chrome | | Pass | Choose color successful |
| 2 | Put down the pieces to make then connected vertically | | Click the chessboard to place pieces | Piece can be put on the chessboard | Piece can be put on the chessboard | Google Chrome | | Pass | The chess pieces are dropped successfully |
| 3 | Put down the last piece to make there are five pieces connected vertically | | Click the chessboard to place pieces | Game over | Game over | Google Chrome | | Pass | Vertical victory judgement successfully |



#### Victory Judge-2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Scenario ID | | victory-2 | | | Test Case ID | | victory-2B | |  |
| Test Case  Description | | Judge whether the system can confirm the victory in the horizontal case | | | **Test Priority** | | High | |
| Pre-Requisite | | Start a new game | | | **Post-Requisite** | | NA | |
| Test Execution Steps: | | | | | | | | | |
| S.No | **Action** | | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | | **Test Result** | **Test Comments** |
| 1 | Start a new game | | Select white | Black first;  White second; | AI go First | Google Chrome | | Pass | Choose color successful |
| 2 | Put down the pieces to make then connected horizontally | | Click the chessboard to place pieces | Piece can be put on the chessboard | Piece can be put on the chessboard | Google Chrome | | Pass | The chess pieces are dropped successfully |
| 3 | Put down the last piece to make there are five pieces connected horizontally | | Click the chessboard to place pieces | Game over | Game over | Google Chrome | | Pass | horizontal victory judgement successfully |



#### Victory Judge-3

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Scenario ID | | victory-3 | | | Test Case ID | | victory-3B | |  |
| Test Case  Description | | Judge whether the system can confirm the victory with pieces placed from bottom-left to top-right | | | **Test Priority** | | High | |
| Pre-Requisite | | Start a new game | | | **Post-Requisite** | | NA | |
| Test Execution Steps: | | | | | | | | | |
| S.No | **Action** | | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | | **Test Result** | **Test Comments** |
| 1 | Start a new game | | Select white | Black first;  White second; | AI go First | Google Chrome | | Pass | Choose color successful |
| 2 | Put down the pieces to make then connected from bottom-left to top-right | | Click the chessboard to place pieces | Piece can be put on the chessboard | Piece can be put on the chessboard | Google Chrome | | Pass | The chess pieces are dropped successfully |
| 3 | Put down the last piece to make there are five pieces connected from bottom-left to top-right | | Click the chessboard to place pieces | Game over | Game over | Google Chrome | | Pass | from bottom-left to top-right victory judgement successfully |



#### Victory Judge-4

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Scenario ID | | victory-4 | | | Test Case ID | | victory-4B | |  |
| Test Case  Description | | Judge whether the system can confirm the victory with pieces placed from bottom-right to top-left | | | **Test Priority** | | High | |
| Pre-Requisite | | Start a new game | | | **Post-Requisite** | | NA | |
| Test Execution Steps: | | | | | | | | | |
| S.No | **Action** | | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | | **Test Result** | **Test Comments** |
| 1 | Start a new game | | Select white | Black first;  White second; | AI go First | Google Chrome | | Pass | Choose color successful |
| 2 | Put down the pieces to make then connected from bottom-right to top-left | | Click the chessboard to place pieces | Piece can be put on the chessboard | Piece can be put on the chessboard | Google Chrome | | Pass | The chess pieces are dropped successfully |
| 3 | Put down the last piece to make there are five pieces connected from bottom-right to top-left | | Click the chessboard to place pieces | Game over | Game over | Google Chrome | | Pass | from bottom-right to top-left victory judgement successfully |

****

# Defect Reporting

## Defect Tracking

PingCode will be used for defect/Issue tracking.

## Defect Reporting and Reports

Defects will be reported until18:00 daily. Defect reports will be generated by PingCode to be reviewed and analyzed during the daily bug defects meeting. The reports will be saved on the network and can be found by accessing the below link:

## Defect Management Process

### Create questions

In the test management system, all users can create new problems, including requirements problems and software defects. When creating a problem, you need to describe it clearly and choose the correct option.

### Confirm the problem

Usually, after receiving a new problem, the development engineer needs to analyze and confirm whether the problem is a bug. If it is a bug, select "confirm status"; If it is not considered a bug, indicate the reason and assign it back to the creator.

### Problem solving

This is the main responsibility of the development engineer, including bug recurrence, modification and modification verification.

If the development engineer needs to analyze and solve the confirmed bug in time and verify it by himself, it will be regarded as the solution state. Select the corresponding option for the solution in the defect management system. After the solution is solved, the system will automatically assign it back to the creator. If the bug cannot be solved or the modification has a great impact, you can apply to enter the "delay solution" process.

### Closing questions

If the bug passes the verification, the verifier needs to indicate the verification result and close it, and the system will assign it to closed. If a closed bug occurs in a later version, activate the bug and the system will automatically assign it back to the resolver

## Defect Severity Definitions

|  |  |
| --- | --- |
| **Critical** | The defect causes a catastrophic or severe error that results in major problems and the functionality rendered is unavailable to the user. A manual procedure cannot be either implemented or a high effort is required to remedy the defect. Examples of a critical defect are as follows:   * System abends * Data cannot flow through a business function/lifecycle * Data is corrupted or cannot post to the database |
| **Medium** | The defect does not seriously impair system function can be categorized as a medium Defect. A manual procedure requiring medium effort can be implemented to remedy the defect. Examples of a medium defect are as follows:   * Form navigation is incorrect * Field labels are not consistent with global terminology |
| **Low** | The defect is cosmetic or has little to no impact on system functionality. A manual procedure requiring low effort can be implemented to remedy the defect. Examples of a low defect are as follows:   * Repositioning of fields on screens * Text font on reports is incorrect |

# Environment

The System Testing Environment will be used for System Testing.

## Hardware Environment

|  |  |  |
| --- | --- | --- |
| **Hardware Environment** | | |
| Test equipment | CPU | 8核 1.80GHZ |
| Memory | 16g |
| Hard Disk | 1T |

## Software Environment

|  |  |  |
| --- | --- | --- |
| **Software Environment** | | |
| Test equipment | OS | Windows 10，64bit |
| Database type | File |
| Test software | Visual studio 2019 |

Because the system is created in freshman year, we didn’t know how to use database and server. The project doesn’t have server and advanced database.

# Test Schedule

System testing is scheduled for a period of 2 weeks starting 06/06/2022 to 06/21/2022. The test team will complete the execution of all the tests during the first 1 weeks. The defects retesting and regression testing will occur in the last week of System Testing. The run dates for defect retesting period may be changed according to the need to retest and close the defects. The defects retesting will reduce the number of open defects that need to be carried to UAT.

|  |  |  |
| --- | --- | --- |
| Task | Date | Time |
| Make test plan | 06/06/2022-06/10/2022 | 5 days |
| Black box test | 06/11/2022-06/12/2022 | 2 days |
| White box test | 06/13/2022-06/15/2022 | 3 days |
| Unit test | 06/16/2022-06/18/2022 | 3 days |
| System test | 06/19/2022-06/21/2022 | 3 days |

# Assumptions

* Process of battle by two players is simulated by one tester.
* The test code is always available to tester, and the developer does not take back the code during the test process.
* No force majeure causes interruption of the test process.
* Test equipment is intact that there will be not something wrong with the computer.

# Risks and Contingencies

## Demand risk

Inaccurate understanding of software requirements leads to errors in test scope, missing part of requirements or executing wrong test methods; In addition, requirement changes lead to test case changes, and there are errors in synchronization.

## Test case risk

The test case design is incomplete, ignoring the boundary conditions and exception handling, and the use case does not fully cover the requirements; Test cases are not fully implemented, and some of them are intentionally or unintentionally omitted.

## Code quality risk

The poor quality of software code leads to many defects, and it is easy to miss the test.

## Test environment risk

In some cases, the test environment is not completely consistent with the production environment, resulting in errors in the test results.

# Appendices