## horizontal line



SCORE Project: DRisk  
A Dynamically Configurable Risk Game  
- Design Document -

11.01.2015

**M**eng-Jung Lin  
**A**nant Srivastava  
**S**hujian Ke  
**C**haoyi Fu  
**S**atyam Jaiswal

**Table of Content**

[Executive Summary](#h.au51mny0sx6)

[Purpose](#h.nnilk4tmvifq)

[Project Overview](#h.36m4896to6b0)

[Sprints](#h.tfdkg1d9n6vs)

[Project Requirements](#h.vibbtmy0fiuj)

[System Overview](#h.3at9u9s4e0vp)

[Sequence Diagram](#h.pko9tevneflj)

[Deployment Diagram](#h.wb47ntcuzhrx)

[Flowchart](#h.xctk0i1xlptg)

[System Architecture](#h.4p7xi5bvhxdr)

[Frontend](#h.o5cd6vwww1cq)

[Database](#h.oywyiom4i2d)

[Server](#h.u677yw8n5rel)

[Interface and Functionalities](#h.81wnuycdlej8)

[enter.html](#h.wncyjeeafkb)

[configure.html](#h.vlo5i77pqub6)

[room.html](#h.s0fb8yzgcbiq)

[RiskMap.html](#h.pn8zytsqirzw)

[Stabilization Plan](#h.alupqf49wjud)

[Test Results](#h.msxzmzk860nb)

[Best Practices](#h.wib9o8jerm57)

[Team Project Website](#h.w0cgoxbtuhsk)

[Project Management](#h.p7nf0va67itf)

[Code Repository](#h.goz1kusa9kvw)

[Software Development Methodology](#h.ez5dj1w5nhp)

[Documented Coding Standards](#h.14l6j66uy1n4)

[Bug Tracking System](#h.m4vvepqpeta3)

[Contribution Summary](#h.yyrhu7ml5bea)

[Meng-Jung Lin](#h.buwz1tcz7y35)

[Anant Srivastava](#h.4sd7ifz8md67)

[Shujian Ke](#h.fic19u74iadq)

[Chaoyi Fu](#h.txxehu395anb)

[Satyam Jaiswal](#h.z7fkk2u88vd3)

# Executive Summary

*SCORE Project: DRisk - A Dynamically Configurable Risk Game -* (DRisk) is a project built by Team MASCS, a group of students from Arizona State University. DRisk is one of the project topics of the 2016 SCORE Contest, which is a student contest on software engineering held every year. The DRisk project is sponsored by Nazareno Aguirre.

Risk is a popular strategy board game designed and created in the late 1950s. Risk is a turn-based board game that allows two to six players to fight over conquering forty two territories. Because of its popularity, and with the technology growing at a rapid pace in the recents years, there have been several PC version of Risk game release to the public, including web versions.

Risk is a strategy game. There are many rules set in stone in the manual. With the current official set up, the playing time of the Risk game can be very lengthy, and can be challenging for some first-time players. There are a couple of simple versions of the risk game, but the rules and game play are still immutable.

The ultimate goal of this project is to create a dynamic Risk game (D stands for Dynamic in DRisk). Team MASCS has created a web-based dynamic Risk game that allows the users to configure some variables within the game. With this setting, we are giving the user more control on how the games is setup and on the playing time of the game, we also think that this can help the first-time players learn the game in a simpler version before going on to a complex full version of the game.

# Purpose

The purpose of the design document is to define the detailed design for all components of DRisk, to give the develops a better idea on how to construct the project. The document explains the project in details, both from the user’s standpoint and the construction behind the scene. The lower level components will be designed, implemented, and tested accordingly.

# Project Overview

The duration of the project is September 21, 2015 through November 23, 2015, which is a total of 9 weeks. Throughout the whole project, there are a total of 4 sprints (2 weeks each) and a mini sprint (1 week). The team will conduct weekly meetings to discuss current status of the project and future plans. Sprint reviews are also conducted during these meetings (every 2 weeks). The team has clear goals and expectation for each sprint, and we strive to meet those goals through constantly reviewing the progress. The team uses many tools to help each member stays on the same page even when we are not meeting in person. Some of the tools the team uses can be found under the [Best Practices](#h.wib9o8jerm57) section of this document.

## Sprints

There are a total of 4 regular sprints and one mini sprint.

Sprint I is Sep. 21 - Oct. 5. This is the very beginning of the project. The team members focus on getting familiar with the project. The team learns the classic board game Risk and discuss our approach of project. Sprint I is the learning portion of the project, where the team is getting used to the tools that we are using and getting our feet wet before going into intense coding. The team starts drafting out the code in the second half of Sprint I.

Sprint II is Oct. 6 - Oct. 19. This is the time when the team starts coding intensively. During Sprint II, the goal is to create a offline version of the Risk game. At the end of Sprint II, the team will have build a web-based Risk game that allows multiple players to play on a single machine.

Sprint III is Oct. 20 - Nov. 2. The focus of Sprint III is to complete every component of the project without a online server. The team will create other webpages that are necessary for accessing the game, and setting up a virtual server on a local machine. At the end of Spring III, the team will have a project that is mostly completed, including allowing user configurations and joining existing games.

Sprint IV is Nov. 3 - Nov. 16. This is the last regular sprint for the team. The major goal of Sprint IV is to push the project online. The team will have a online server to host the website, allowing the players to play the game from different machines. Testing will be conducted to make sure the product is completed at the higher level. The team will be implementing additional features if there is additional time.

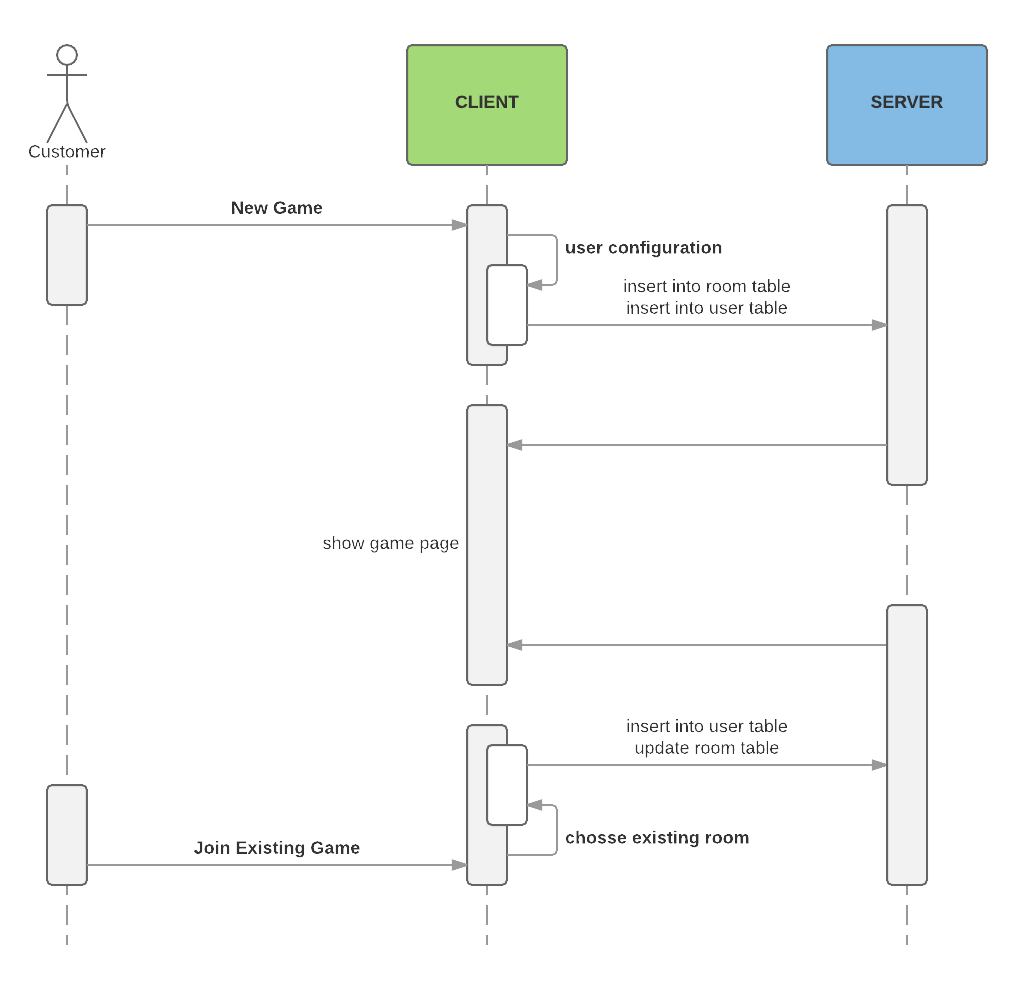
The final sprint is the mini sprint, which is Nov. 17 - Nov. 23. The team will focus on stabilization of the project and work on the final deliverable and presentation. The team is not expecting a huge amount of changes or implementation during this week of mini sprint.

# Project Requirements

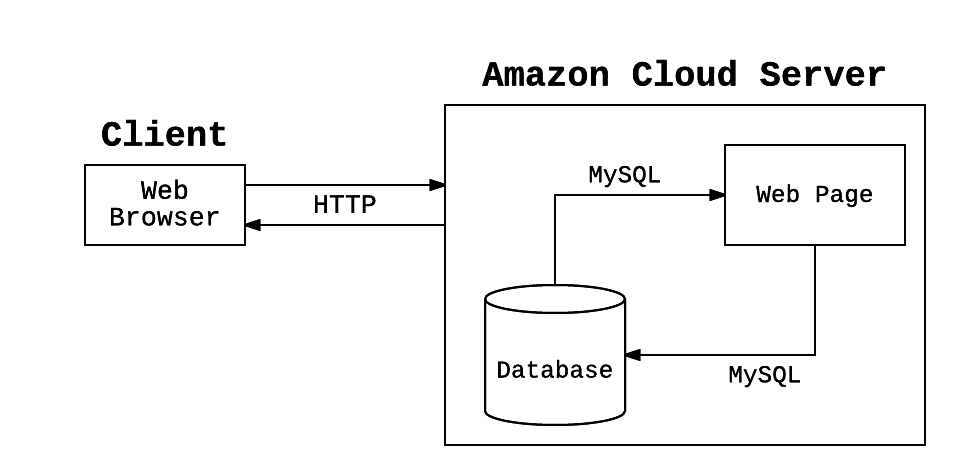
Please refer to the [Requirements Document](https://docs.google.com/a/asu.edu/document/d/1lk0DBeI8PPqbjHiKnBhQMSMSMCiLiXYtqc-34Q6ewpk/edit?usp=sharing) for project requirements.

# System Overview

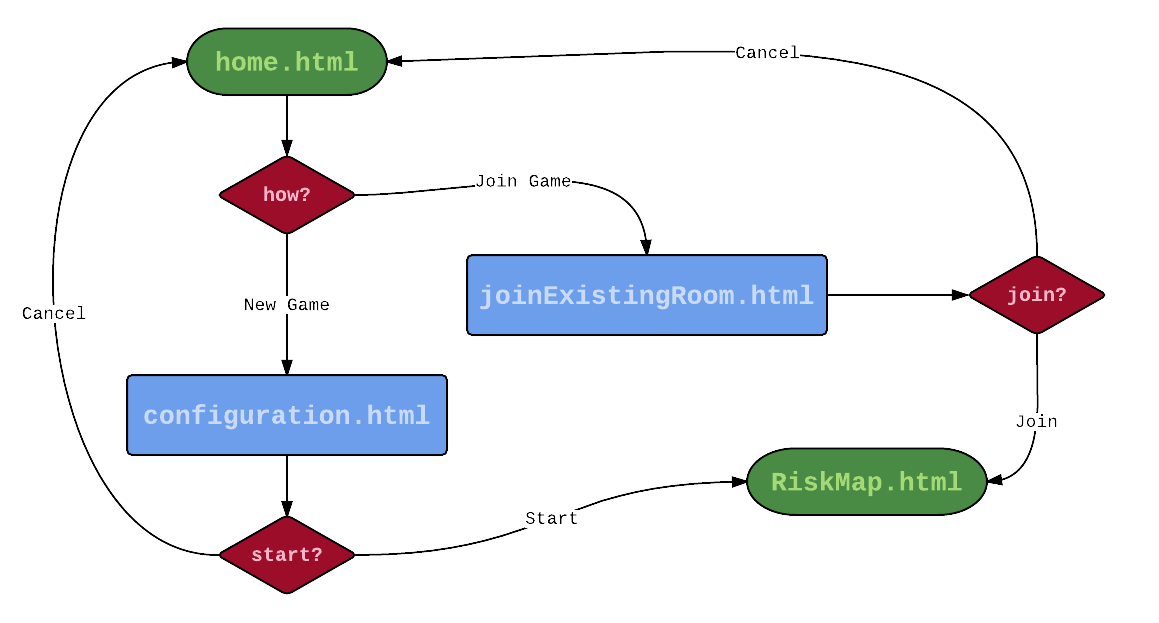
## Sequence Diagram



## Deployment Diagram



## Flowchart



# System Architecture

## Frontend

Because DRisk is a web-based project, the layout of the web pages are written in HTML. There are a total of 4 html pages, each serves its own purpose. There is a home page which is the entry point of the project. There is a configuration page which allows players to set configurations of the game. There is a join page for players to join an existing game by selecting a room or entering a specific room number. There is the RiskMap page that contains the actual Risk game, where the game is created based on the user configurations. There are css files to support the display of the page.

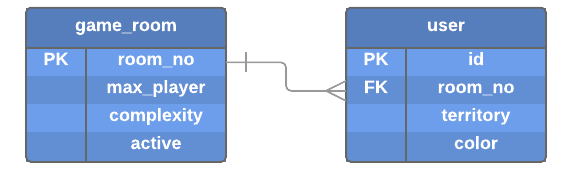
One of the main part of the project is the online Risk game. The team includes javascript code for the game page to handle many of the functionalities. The javascript code is used primarily to do calculations during game, such as calculation in the attack function. It handles the game play on a more complex level in computation. It is also used to enable some of the animation during the game play. The majority of the code for frontend development are html, javascript, and css.

The project also includes a database to store some of the data. The program the team will be using to make connections to the database will be written in PHP.

The frontend of the project is primarily what the user interact with, the team strive to make the website and the game as user friendly as possible.

## Database

The team adapts MySQL as the database. The primary use of the database is to keep track of the information of each game room and status of each user, therefore, we need 2 tables to handle the data.



[redo shcema]

|  |  |  |  |
| --- | --- | --- | --- |
| **Table: game\_room** | | | |
| **Key** | **Field** | **Type** | **Description** |
| P | room\_no | VARCHAR(8) | a unique string that represent each game room |
|  | max\_player | SMALLINT(1) | indicates the maximum number of players allowed in the specific game room |
|  | complexity | SMALLINT(1) | stores the complexity of the map in the specific game room (0=easy, 1=medium, 2=hard) |
|  | active | BIT(1) | indicates whether the game room is currently allowing players to join |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table: users** | | | |
| **Key** | **Field** | **Type** | **Description** |
|  | Name | VARCHAR(20) | Name for the player |
| P | ID | INT(10) | a unique string that represent each user |
| F | room | INT(11) | a string to represent the room the user is currently in |
|  | Territories | INT(10) | keep track of the number of territory the user has |
|  | color | VARCHAR(10) | the territory color for the user in the game |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table: territories** | | | |
| **Key** | **Field** | **Type** | **Description** |
| P,F | userID | INT(11) | a integer value that represent each user |
|  | 1 | TINYINT(1) | indicates whether territory 1 belongs to the user |
|  | 2 | TINYINT(1) | indicates whether territory 2 belongs to the user |
|  | ...  42 | TINYINT(1) | indicates whether territory … 42 belongs to the user |

## 

## Server

Amazon AWS???

# Interface and Functionalities

Each web page of the product will be listed in this section. The goal is to provide the developer a good understanding on the process and action made on each step and for different controls. A layout design will be provided to the developer as a protocol, the developer will design the web page according to the protocol. Controls on the page will be labeled, and will be explained in detail in a table.

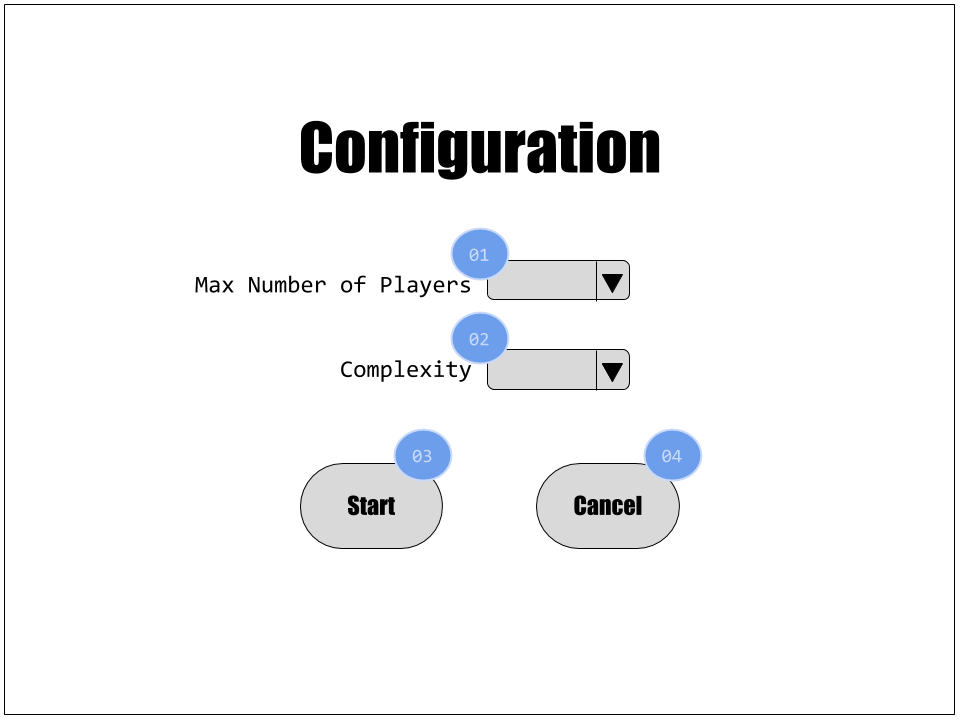
## enter.html



This page is the entry point of the product. The user will enter this page and decide how he want to start playing the game.

|  |  |  |
| --- | --- | --- |
| **ID** | **Control Type** | **Description** |
| 01 | button | By clicking on the “New Game” button, the user will be redirected to the configure.html page. |
| 02 | button | By clicking on the “Join Room” button, the user will be redirected to the room.html page. |

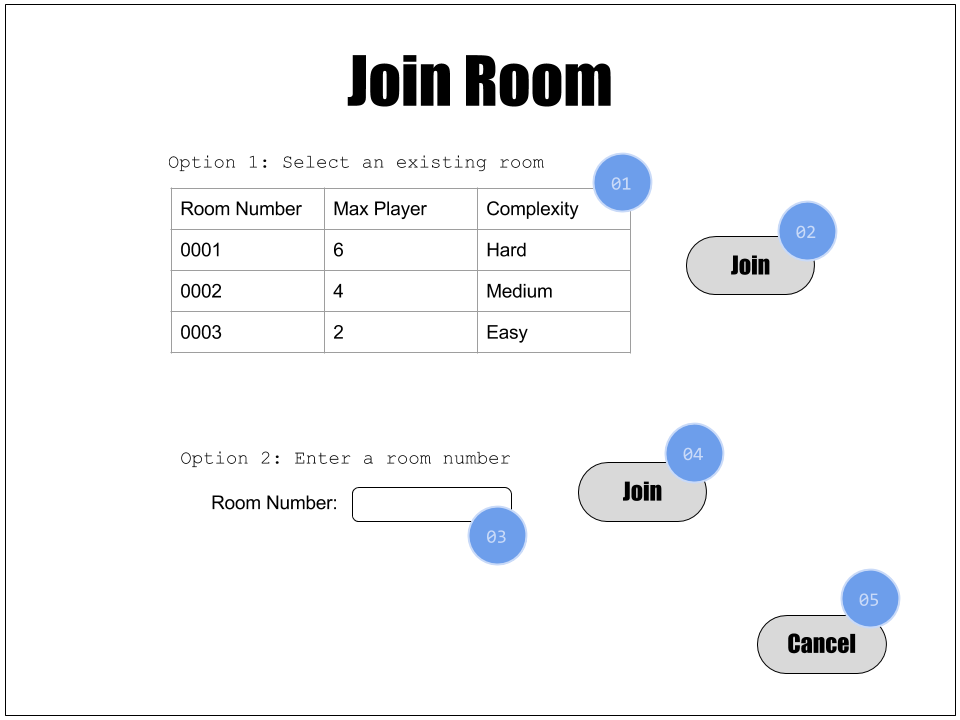
## configure.html



The user will be able to set configuration for his game on this page. The user can initial a new game or cancel to go back to the entry point.

|  |  |  |
| --- | --- | --- |
| **ID** | **Control Type** | **Description** |
| 01 | dropdownlist | The dropdownlist allow the user to select the max number of player for the game. [2, 3, 4, 5, 6] |
| 02 | dropdownlist | The dropdownlist allow the user the select the complexity of the map for the game. [Easy, Medium, Hard] |
| 03 | button | By clicking on the “Start” button, the user initializes a new game.  1. A new entry is inserted into the game\_room table with info about the room.  2. A new entry is inserted into the user table with info about the user (host if the game).  3. User is redirected to RiskMap.html to start a new game, using the configuration set at 01 and 02. |
| 04 | button | By clicking the “Cancel” button, the user will be redirected to the entry point of the product, which is enter.html. |

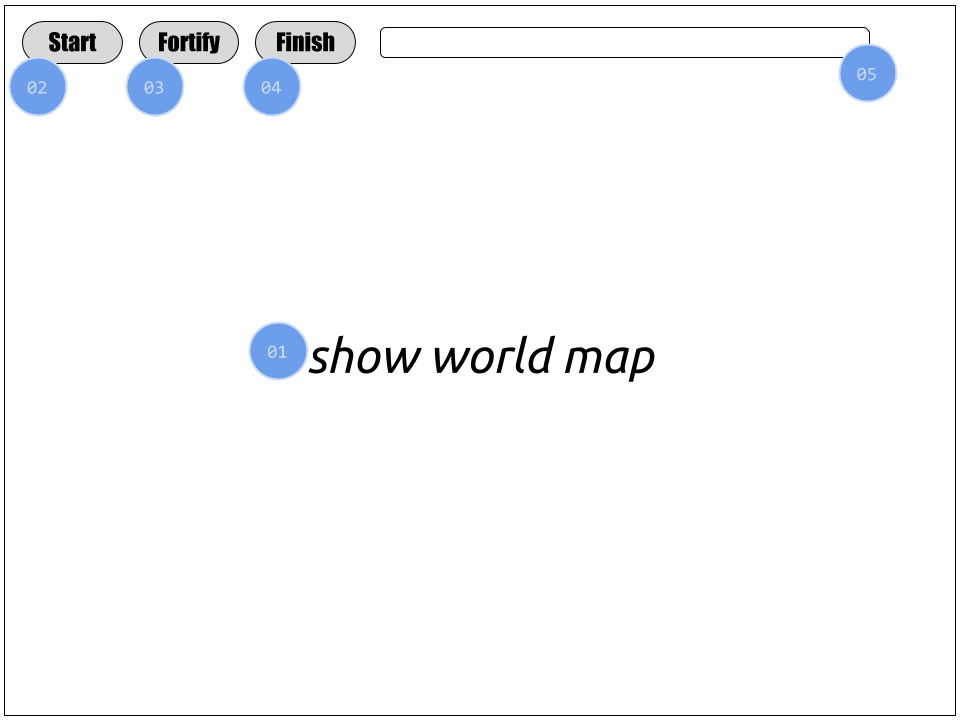
## room.html



The user will be choosing the existing game that he wants to join. There are two options for the user to choose the game room. The user can decide to join a game or cancel and return to the entry point.

|  |  |  |
| --- | --- | --- |
| **ID** | **Control Type** | **Description** |
| 01 | table | The table will show a list of game rooms that are currently active and available for joining.  SELECT room\_no, max\_player, complexity FROM game\_room WHERE active = ‘1’ |
| 02 | button | By clicking on the “Join” button, the user will join the game that was selected.  1. A new entry is inserted into the user table with info about the user.  2. User is redirected to RiskMap.html to join the game that was selected in 01. |
| 03 | textbox | The textbox allows the user to enter a specific game room number instead of searching through the list. |
| 04 | button | By clicking on the “Join” button, the user will join the game with the specific room number.  1. A new entry is inserted into the user table with info about the user.  2. User is redirected to RiskMap.html to join the game that was selected in 03. |
| 05 | button | By clicking the “Cancel” button, the user will be redirected to the entry point of the product, which is enter.html. |

## RiskMap.html



This is the major game page of the product. The user will be able to play the game on this page, whether it is a game he initialized or joined. Parameters will be passed to this page during new game or join game to be able to display the page accordingly. The page consist an interactive map and several other controls related to the game play.

|  |  |  |
| --- | --- | --- |
| **ID** | **Control Type** | **Description** |
| 01 | image | The center area of the page is an interactive map. The user will be able to click on the territories on the map to play the game. |
| 02 | button | By clicking on the “Start” button, the user will officially start the game.  The status of the game room will be updated in the database.  UPDATE game\_room SET active = ‘0’ WHERE room\_no = ‘0001’ |
| 03 | button | By clicking on the “Fortify” button, the user will enter the fortify steps. The user will be able to move troops from one territory to another in this stage. |
| 04 | button | By clicking on the “Finish” button, the user will and his turn. The next user will begin his turn. |
| 05 | label | The label will be giving hint to the users while the game is in play. |

# Stabilization Plan

The team has a stabilization plan for the project. Towards the end, the team will be conducting testing of the whole product at a higher level. The functionalities will be tested as the project develop, the high level testing is to make sure that each part of the project work together properly.

Please refer to the [Test Cases](https://docs.google.com/a/asu.edu/spreadsheets/d/1KbbvjpBVZFqcI3AKDt3CbE_XLmSuUAtnO6DS_5KZB6k/edit?usp=sharing) document for specific test cases and details.

## Test Results

The below table briefly shows the results of the high level test cases.

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Pass/Fail** | **Comments** |
| TC01 |  |  |
| TC02 |  |  |
| TC03 |  |  |

# Best Practices

## Team Project Website

<https://sites.google.com/a/asu.edu/teammascs/>

The team created a Google site to use as the team project website. The site served as a wiki page for the project, containing project information and updates.

## Project Management

<https://tree.taiga.io/project/ser515asu-drisk-team-mascs/>

<https://drive.google.com/drive/folders/0B7Bgtpfy_SIzQ1F5NHlKa0JQNVU>

The team used Taiga and Google Drive for project management.

Taiga served as the SCRUM board for the team. User stories and sprint tasks will be posted here. Status of the sprint tasks will be updated accordingly. Taiga provides a convenient way to monitor the project.

Google Drive is primarily where the team share the documents. Google Drive gives each member of the team to access any documents at any time. This will eliminate emailing the documents and have multiple documents around without knowing which the most current version is.

Meeting minutes are posted every week after our weekly meeting. Sprint notes are updates during each sprint review (every 2 weeks). Documents to be submitted to Dr. Gary are also shared on Google Drive.

## Code Repository

<https://github.com/ser515asu/DRisk-Team-MASCS>

The team used GitHub as the code repository and code management. Team members used the GitHub repository to keep track of the status of the project and as version control system.

## Software Development Methodology

The team adapts SCRUM methodologies as the development process.

## Documented Coding Standards

* The project is written in PHP on the frontend
* Team members follow the coding standards so the code is easy to follow and understand
* Coding Standard for PHP:
  + Files use only *<?php* and *<?=* tags
  + Files use only UTF-8 without BOM
  + Files either declare symbols or cause side-effects, but not both
  + Class names are declared in *StudlyCaps*.
  + Method names are declared in *camelCase*.
  + Code use 4 spaces for indentation, no tabs
  + One blank line after the *namespace* declaration
  + One blank line after the block of *use* declaration
  + Opening braces for classes go on the next line, closing braces go on the next line after the body.
  + Visibility is declared for all properties and methods: *abstract* and *final* is declared before the visibility, *static* is declared after the visibility
  + Control structure keywords have one space after, method and function calls does not.
  + Opening braces for control structures go on the same line, closing braces go on the next line after the body.
  + Opening parentheses for control structures does not have space after, closing parentheses for control structures does not.
* Coding Standard for JavaScript:
  + Variable and function names written as *camelCase*
  + Global variable written in *UPPERCASE*
  + Constants written in *UPPERCASE*
  + All names starts with a letter.
  + Always put spaces around operators and after commas.
  + Code use 4 spaces for indentation, no tabs
  + Always end a simple statement with a semicolon
  + Opening braces at the end of the first line, closing braces go on the next line after the body without leading spaces
  + One space before opening braces
* Coding Standard for HTML:
  + Declare document type as the first line in document
  + Lowercase element names
  + Close all html elements
  + Close empty html elements
  + Use lowercase for attribute names
  + Quote attribute Values
  + No spaces around equal signs
  + No blank lines when unnecessary
  + Code use 2 spaces for indentation, no tabs
  + Use *<html>* and *<body>* tags
* Coding Standard for CSS:
  + Code use tabs for indentation, no spaces
  + Two blank lines between sections
  + One blank lines between blocks in sections
  + Each selector on its own line, ending with either a comma or an open brace
  + One space before opening braces

## Bug Tracking System

The team will be using a Google spreadsheet to keep track of bugs found in the system.

# Contribution Summary

## Meng-Jung Lin

* Initial setup of backend structure
* Implement *Attack* logic
* Contribute to the Design Document
* Contribute to Requirements Document
* Contribute to User Manual
* Contribute to Test Cases Document
* Act as team Scrum Master, managing Taiga scrum board
* Mid-term Video: voice & script
* Housekeeping items:
  + Managing team website
  + Managing team Google drive (minutes, sprint notes, documentations)
  + Organize meetings: study room reservation, meeting reminder, meeting agenda, take minutes

## Anant Srivastava

* Build the interactive map
* Figure out the basic function for backend
* Implement UI for map page
* Implement the game to allow 2 users
* Implement database
* Create server script

## Shujian Ke

* Implement the interactive map
* Implement logic for troop deployment
* Implement *Deploy* function
* Contribute to front end design
* Research frameworks to be used for the project (KineticJS tool)
* Mid-term video: script & animation

## Chaoyi Fu

* Build the interactive map
* Implement *Deploy* and *Fortify* functions
* Mid-term video: animation & recording

## Satyam Jaiswal

* Research frameworks to be used for the project (AngularJS, MVC, AJAX)
* Understanding the Risk game thoroughly
* Implement *Attack* and *Fortify* function
* UI enhancement
* Mid-term video: recording