## horizontal line



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

10.19.2015

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

**Table of Content**

[Executive Summary](#h.au51mny0sx6)

[Project Requirements](#h.rs1fxx4zy2zz)

[Functionality Requirements](#h.stkxeg8x1sbv)

[Usability Requirements](#h.gi64wgrytlet)

[Reliability Requirements](#h.azba5bz74g9q)

[Performance Requirements](#h.yl8qaa578f20)

[Supportability](#h.1dmpsv8wjasg)

[Constraints](#h.j5x2nqw0dht1)

[Project Schedule](#h.cj69ac1er7v1)

[Gantt Chart](#h.edovvi96pjpk)

[Tasks & Milestones](#h.1nikza7o7q2)

[System Overview](#h.3at9u9s4e0vp)

[System Architecture](#h.4p7xi5bvhxdr)

[Frontend](#h.o5cd6vwww1cq)

[Backend](#h.ch02qjfzae6f)

[Database](#h.w6ff0utv71hr)

[Server](#h.u677yw8n5rel)

[Test Results](#h.k3v48w4o3ocn)

[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 give the develops a better idea on how to construct the project.

# Project Requirements

Since DRisk is a contest project topic, there are some set requirements that the team has to follow when building the project.

Original project description and requirements can be found on the official SCORE project website (<http://score-contest.org/2016/projects/drisk.php>)

Note that ID is subject to change as the lists grow

## Functionality Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **CATEGORY** | **ID** | **PRIORITY** | **DESCRIPTION** |
| **FEATURE SET** | F01 | 1 | The project implements all core functionalities of the standard version of Risk game. |
| F02 | 1 | The project is a Risk game to be played through a web site. |
| F03 | 1 | The game allows 2-6 player to play. |
| F04 | 1 | The game has a preloaded, 42-territory map. |
| F05 | 1 | The game allows user to define and upload map configuration: image (aesthetic aspects) |
| F06 | 1 | The game allows user to define and upload map configuration: number of territories |
| F07 | 1 | The game allows user to define and upload map configuration: grouping into continents |
| F08 | 1 | The game allows user to define and upload map configuration: neighbor relations |
| F09 | 3 | The game allows user to define and upload map configuration: number of players |
| F10 | 1 | The game allows for maps to automatically adjust granularity by collapsing neighboring territories according to number of players |
| F11 | 1 | The game allows the user to select complexity of the game (easy, medium, hard) |
| **Security** | F12 | 1 | User will not be able to access the server |

## Usability Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **CATEGORY** | **ID** | **PRIORITY** | **DESCRIPTION** |
| **HUMAN FACTORS** | U01 | 1 | The game runs on Internet Explorer. |
| U02 | 1 | The game runs on Chrome. |
| U03 | 1 | The game runs on Firefox. |
| U04 | 5 | User can access the mobile version of the webpage through mobile devices. |
| **AESTHETIC** | U05 | 3 | The game contains an elegant interface to make it easy for user to navigate around. |
| **CONSIS- TENCY** | U06 | 3 | Information of each new game will be stored in the database in the same format |
| U07 | 3 | All code follows the same coding standards for easy understanding and maintenance. |

## Reliability Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **CATEGORY** | **ID** | **PRIORITY** | **DESCRIPTION** |
| **PREDICTA-BILITY** | R01 | 2 | Server will restart itself if it crashes. |
| R02 | 1 | Entering a URl will take the user to the correct page and load the correct game instance. |
| R03 | 2 | The web site work the same way in IE8, Chrome 46, and Firefox 41 |
| **ACCU- RACY** | R04 | 4 | HTTPS will be supported for users who wish to use it. |
| R05 | 1 | Information for each game is accurate. |
| **RECOVER-ABILITY** | R06 | 2 | When user enters a wrong URL or click on a broken link, a 404 page is displayed. |

## Performance Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **CATEGORY** | **ID** | **PRIORITY** | **DESCRIPTION** |
| **RESOURCE CONSUMPTION** | P01 | 2 | Database storage does not exceed 30 GB. |
| P02 | 2 | Website handles 1,000 games at the same time. |
| P03 | 4 | Server downtime does not exceed an hour per week. |
| **RESPONSE TIME** | P04 | 4 | Game page loaded within 5 seconds after starting new game or join existing game. |
| P05 | 1 | Web pages loaded under 10 seconds even under slow connection |

## Supportability

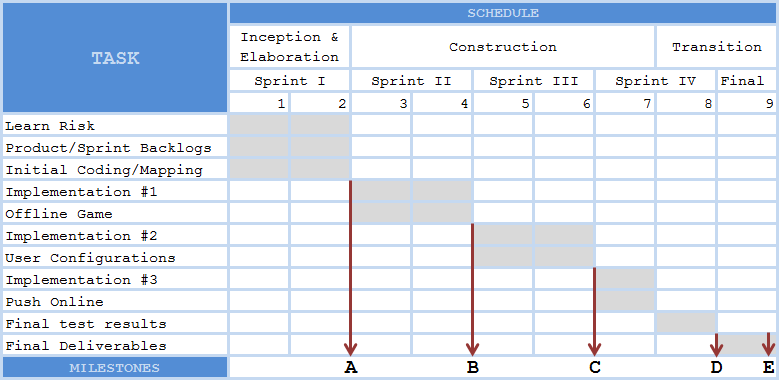
|  |  |  |  |
| --- | --- | --- | --- |
| **CATEGORY** | **ID** | **PRIORITY** | **DESCRIPTION** |
| **TESTA- BILITY** | S01 | 2 | Controllability: Possible to control the state of component under test as required for testing. |
| S02 | 2 | Observability: Testers are able to observe intermediate and final result. |
| S03 | 4 | Understandability: Component under test are documented or self-explaining. |
| **EXTENCI- BILITY** | S04 | 3 | New features are easily implemented without breaking or changing other parts of the program |
| **MAINTAIN-ABILITY** | S05 | 3 | Future maintenance is easy. |
| **EASE OF INSTAL** | S06 | 2 | User is able to access the website using any browser |

## Constraints

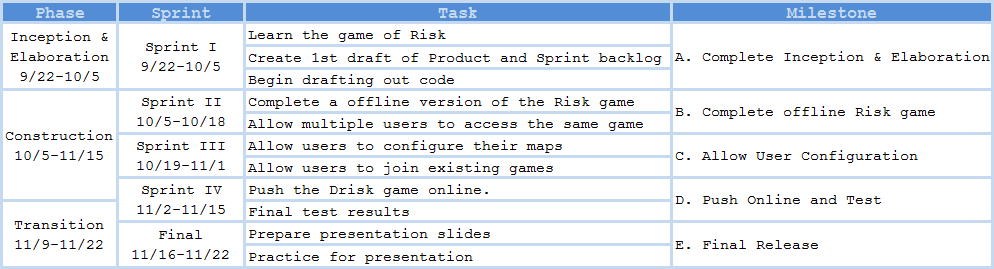
|  |  |  |  |
| --- | --- | --- | --- |
| **CATEGORY** | **ID** | **PRIORITY** | **DESCRIPTION** |
| **DESIGN** | C01 | 2 | Object-oriented design/language is used. |
| C02 | 3 | Information displayed on the website is handled by OO database model. |
| C03 | 4 | Development period is one semester |
| **IMPLEMENTATION** | C04 | 1 | Use PHP and Java as web programming languages |
| C05 | 1 | Use Amazon AWS as the server |
| C06 | 2 | The database is MySQL |
| **INTER- FACE** | C07 | 3 | The website is fully functional using IE, Chrome, and Firefox. |
| **PHYSICAL** | C08 | 2 | The website is accessible using PCs, smartphones, and tablets. |

# Project Schedule

## Gantt Chart



## Tasks & Milestones

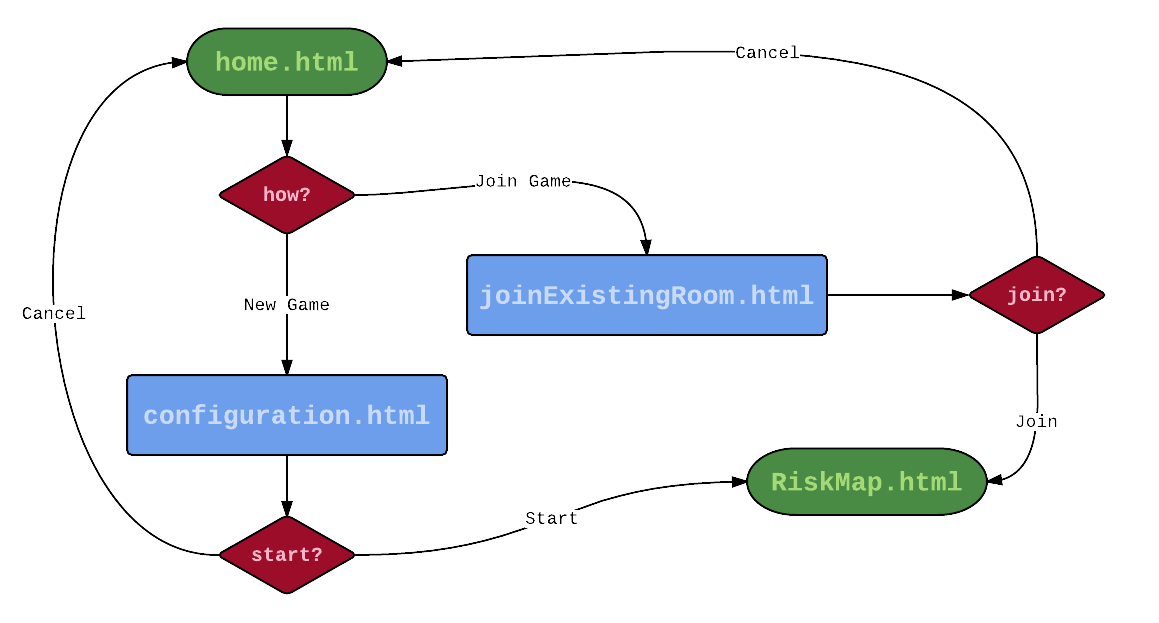


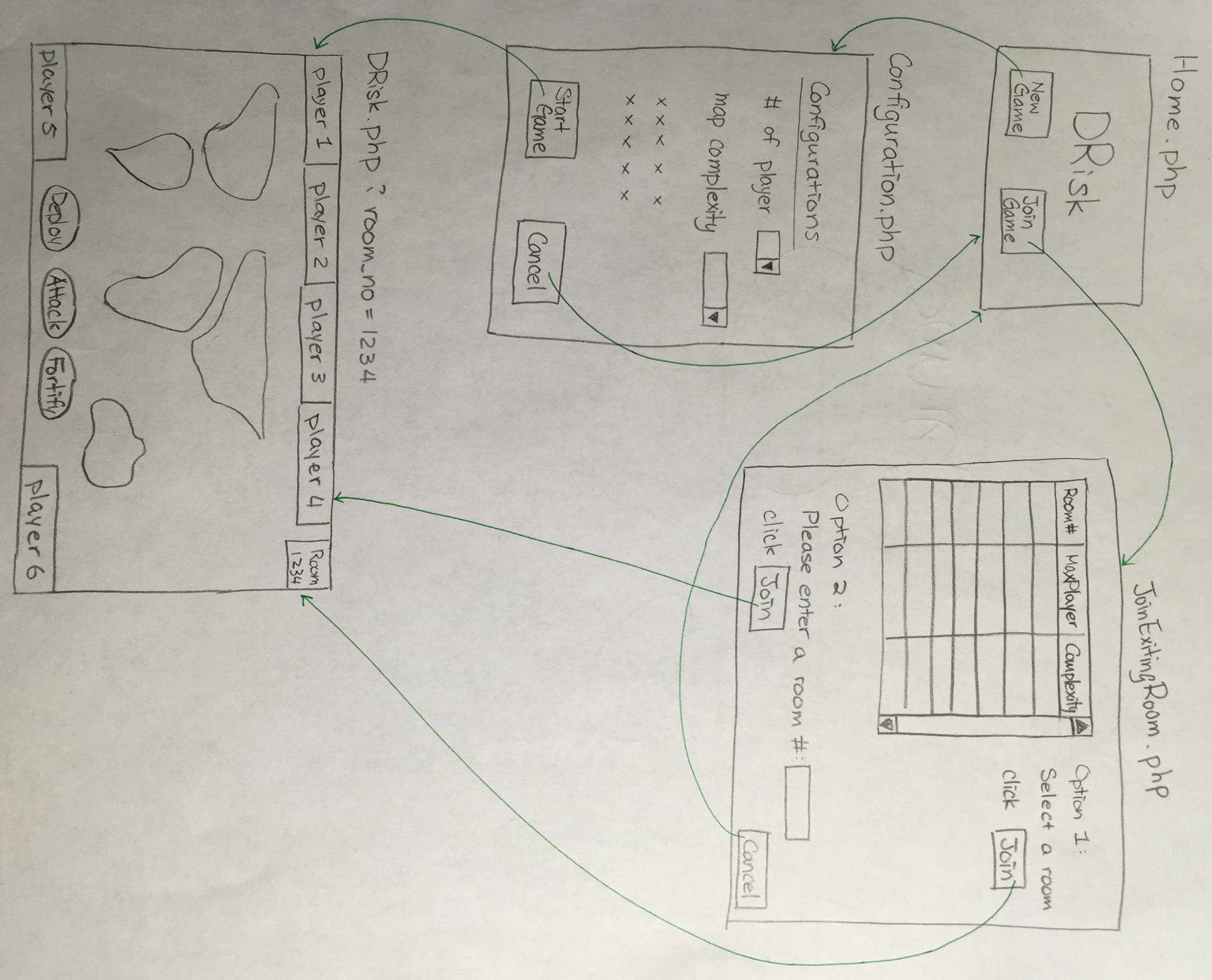
# System Overview

## Sequence Diagram

xxx

## Flowchart



this hand-drawn image will be removed later.

# 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.

[insert ERD]

|  |  |  |  |
| --- | --- | --- | --- |
| **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: user** | | | |
| **Key** | **Field** | **Type** | **Description** |
| P | id | VARCHAR(8) | a unique string that represent each user |
| F | room\_no | VARCHAR(8) | a string to represent the room the user is currently in |
|  | territory | VARCHAR(8) | keep track of the number of territory the user has |
|  | color | VARCHAR(8) | the territory color for the user in the game |

## Server

Amazon AWS???

# Interface and Functionalities

This section list out all the web pages and describe the functionalities

# Unit Testing

## Test Cases

|  |  |
| --- | --- |
| TC01 | xxxx |
| Precondition | xxxx |
| Sequence of Action | 1. xxx  2. xxx  3. xxx |

|  |  |
| --- | --- |
| TC02 | xxxx |
| Precondition | xxxx |
| Sequence of Action | 1. xxx  2. xxx  3. xxx |

|  |  |
| --- | --- |
| TC03 | xxxx |
| Precondition | xxxx |
| Sequence of Action | 1. xxx  2. xxx  3. xxx |

## Test Results

|  |  |  |
| --- | --- | --- |
| **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 Agile methodologies and 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
* Mid-term video: recording