Match Making System

Technical Design Document

*Version 1.0*

*Omar Ibrahim*

*Website: omar47i.github.io/*

*Mobile: +85259393797*

# Table of Contents

[Table of Contents](#_f3hbr6p0ak37)

[Design History](#_e5akww2oi216)

[Solution Overview](#_j2kld1ana13b)

[Solution Summary](#_i566fs1eqqtn)

[Target Platform](#_w0h9hbmhvcr)

[Development Overview](#_h7psffrwssbg)

[Development Team](#_xxk0bak9cyj9)

[Development Software](#_drxstdfky6pr)

[External Code](#_3wx81gnjfbw1)

[Game Mechanics](#_hm65yr3rs85w)

[Main Technical Requirements](#_c2kb94gs6u8i)

[Architecture](#_yxdrr8vse4bb)

[Game Flow](#_93tph0q2eust)

[Graphics](#_dal8viau5j83)

[Audio](#_oc7d279r9ymu)

[Game Objects and Logic](#_jm2i9mtxtpro)

[User Interface](#_rvrktgcok4ku)

[In Game HUD](#_9sbf9gvn1b4w)

# 

# 

# 

# Design History

Version 1.0: First produced version of the match making system.

# 

# Solution Overview

--------------------------------------------------------------------------------------------------------------------

# Solution Summary

The match making system is responsible for match making the players based on the game mode they pick, which is PvP or 1v1, a team of 2v2, and a team of 3v3. The first mode, which is 1v1, is

## Target Platform

As a casual game, it mainly target PC web browsers, with support to mobile web browsers too.

# 

# Development Overview

--------------------------------------------------------------------------------------------------------------------

# Development Team

[Omar Saeed](http://github.com/Omar47i/)

Game Design and implementation, UI Design, Animations, Sounds, AI, pretty everything.

# Development Software

Project is developed using Typescript and Pixi-js framework, Photoshop is used for UI image editing, Version control is handled by [Git](http://git-scm.com/) with the repository hosted on [GitHub](http://github.com/).

# External Code

[Howler.js](https://github.com/goldfire/howler.js) audio library for sound effects, I haven’t used any other plugin or ide, I used notepad++ for coding.

The game has been made from scratch.

# Game Mechanics

--------------------------------------------------------------------------------------------------------------------

# Main Technical Requirements

The main technical requirements for the minesweeper game is straight forward:

* Opening a cell.
* Open adjacent cells.
* Determine danger locator texts
* Place mines across the grid
* Winning and losing handling
* HUDs
* Smart mine placement (place mines after first click)
* Placing flags.
* Bot auto playing

# Architecture

There is only one main module in the game which is the main play area, let’s call it gameplay module.

Gameplay module hosts all the game elements such as: cells, mines, flags, and HUDs,

# Game Flow

The main flow of the game is as follows: When game starts you are presented with a 2-d grid containing unrevealed cells and the HUD.

When you click on a safe cell the cell being clicked is revealed and revealing all adjacent cells and revealing danger locator texts.

You can click on Bot button to make the bot play the game instead of you, you can also pause the bot and continue playing at any time which is cool right?

you lose by clicking on a mine cell and win by clicking on all empty cells.

You can always restart the game by clicking on ‘New Game’ button, you can also export the game as a text of 2d grid containing 0s for safe cells, m letter for mines, and numbers for danger locator cells.

# Graphics

Game graphics is pretty simple, most of the image resolutions are 128 px maximum.

# Audio

The game has some sound effects that are played upon:

* Clicking on an empty cell.
* Clicking on a mine.
* Marking a cell.
* Winning.
* Losing.

# Game Objects and Logic

**Cell**

The main game object in the game, Cell game object is responsible for storing the cell state and displaying the cell graphics.

**Grid**

The grid game object handles the process of instantiating the 2d grid of cells and handling the group operations made on all the grid cells

**HUD**

HUD.js is responsible for making the heads up display and updating their sprite states upon change.

**Bot**

Bot is responsible for playing the game automatically with the ability to pause the bot anytime.

**Main**

Main is the starting script in the game as it’s responsible for loading the game and instantiating their objects.

# 

# User Interface

--------------------------------------------------------------------------------------------------------------------

# In Game HUD

The game HUD contains:

* Mines count text.
* Face icon (changes on cell click)
* Cells count text.
* New Game button.
* Export button.
* Bot button.
* Win text.
* Lose text.