ABSTRACT
One of the most addictive game is 2048 which is available across all platforms even in wearable devices. We created an agent which capable of playing the game without human interference and with an increased possibility of winning this game when compared to an average player. For this, we used exceptiminmax often referred as expectimax to solve the game which calculates all the possible moves and selects the one with highest probability.

Author Keywords
2048, 2048 Solver, 2048 Expectimax.

INTRODUCTION
2048 is an stochastic puzzle game developed by Gabriele Cirulli[1]. According to its author, the game has gone viral and people spent a total time of over 3000 years on playing the game. Besides the online version the game is available across all the platforms, moreover the game is simple to learn and play on the first go, but as the game progresses it will get tougher and tougher to reach the actual 2048 the winning probability of that is 1% among the million.

2048 is a single player game which is played on a 4 x 4 board which has a total of 16 tiles. Each tile on the board can be empty or have a tile with a value of x which is a multiple of 2. The game starts with two tiles with tile values either 2 or 4, a random tile will be inserted for each and every move at random locations among the empty cells, the tile with value 2 which has the probability of 0.9 and value 4 with a probability of 0.1, So tile 2 occurs most of the time.

To play the game we have to slide or choose in the direction to merge tiles, where there are two adjacent tiles of same tile value till it reaches 2048. At each move we have to choose a direction among 4: UP, DOWN, RIGHT and LEFT. After each move, a new 2-tile or 4-tile is randomly generated according to the aforementioned probabilities. When me merge a tile with same values a new tile is formed with the sum of the merged tiles. When making a move, all the tiles move towards the desired direction if there is an empty cell, not only the one with the same tile value.

Moreover while making a move the player is rewarded with a reward of the value of the merged tile and a reward of zero if there is no tile merged. The player is considered won when he creates a tile 2048. However the player can continue to play the game after he reaches the 2048 tile, the game terminate if the player has no possible moves left i.e. all the square are filled and all tiles isolated with no tile nearby with the same value. The game score is a sum of all the rewards the player earned till he reaches the terminal state.

PROJECT DESCRIPTION
Set of states, S: (4 x 4) 18.
Set of actions, A: UP, RIGHT, DOWN or LEFT
Probabilities, P : 0.9 for tile 2 and 0.1 for tile 4
Rewards: for each merge, player gains a reward equal to the value of the new merged tile.
Transition function: nondeterministic due to the random tile generation procedure, which determines the probabilities.

Goal: merging the adjacent tiles in order to create a tile with the value of 2048.

Data Sets: We will use the tile number in current state, score obtained by the agent at goal, time taken by the agent to reach the goal and the number of moves as data set.

Proposed Approach: The proposed approach to solve this project is by using Q-Learning, the agent travels to all the possible nodes and stores the rewards for each and every move and uses Q-Learning to make the agent to decide what is the best action to take which helps the agent by providing all the required information about the path to take in and win the game.

With this approach we faced a setback when we are expanding the nodes since we have a very large state space the Q-Learning helps the agent to take the moves initially but as we go deep in the tree the exploration algorithm cannot handle the large state space and runs out of memory so we decided to change the approach.

Used Approach: Since we had a problem in the node exploration due to the enormously large state space so we have to use an approach that helps that agent not to go and stuck in the depth. So to solve the puzzle we use Expectimax which helps the agent to choose the best path and thereby win the game.

Expectimax: Expectimax is a special variation of minimax game tree used to play two-player zero-sum games such as backgammon by artificial intelligence systems, in which the moves depend upon the player's skill and random chances. Expectimax has chance nodes in addition to min and max, which takes the expected value of random event that is about to occur, While in a a normal minimax approach we have min and max nodes.

Expectimax's chance nodes are interleaved with max and min nodes, in contrast with minimax where the levels of the tree alternate from max to min until the depth limit of the tree has been reached, where interleaving depends on the game. Chance nodes take a weighted average where weight is the probability that the child is reached instead of taking the min or max of utility values. In each turn the game is evaluated as the max node and the opponent turn is calculated as the min node.

Pseudocode: The pseudocode for expectimax

```plaintext
function expectimax(node, depth, player)
    if node is a terminal node or depth = 0
        return the heuristic value of node
    if the adversary is to play at node
        let α := +∞
        foreach child of node
            let α := min(α, expectimax(child, depth-1))
        else if we are to play at node
            let α := -∞
            foreach child of node
                let α := max(α, expectimax(child, depth-1))
            else if random event at node
                let α := 0
                foreach child of node
                    let α := α + (Probability[child] * expectimax(child, depth-1))
                return α
```
easier so we wrote a heuristic that keeps the tiles on the left. The heuristic works as follows, the tiles are made to occupy the left side of the grid then we play with the make it merge on the left i.e. the keeping the left column cells filled all the time.

DISCUSSION
When deciding which approach to take when we figured our Q-Learning is not a good approach to solve the puzzle. We both agreed to use expectimax because its better than minimal in terms of performance and reliability.

The first approach consider was minimax game tree search. It uses alpha-beta pruning, a human-designed static evaluation function and several hand-tuned parameters. Its winning rate depends on the time the minimax is allowed to search for the best move. We checked that, given 100 ms for making a move, it wins 89% of games, which is a significantly worse result than that achieved by the best of our agents.

Expectimax searches with the depth limit and a heuristic evaluation function. It is capable of winning the game in 100% of cases, but this number is based on only 100 games, and thus it may be overestimated. However, despite an efficient implementation with some low level optimizations, a single game takes as long.

CONCLUSION
The puzzle game 2048 constitutes a new interesting challenge for computational intelligence methods. In this study we have shown that for this stochastic game the Expectimax equipped with a heuristic function is able to produce agents winning nearly 90% of games on average. This performance obtained is comparable with the performance of the best computationally-intensive search-based agents.

ACKNOWLEDGMENTS
The work described in this paper was developed as part of a fall 2014 Artificial Intelligence course, in the Computer Science department of the University of Massachusetts Lowell by Prof. Fred Martin. Vignesh did the GUI part and the grid and game manager part, Gayas did the AI the Expectimax.

REFERENCES
1. 2048 original game.
   http://gabrielecirulli.github.io/2048/
2. Expectimax pseudocode.