Team Member:

Jiahui Wei jiahuiwe@usc.edu

Zhenyang Zhong zhenyanz@usc.edu

Zhongyang Gao zhongyag@usc.edu

Cheuk Yin Matthew Lam cheukyin@usc.edu

Yiqing Xu yiqingxu@usc.edu

Brandon Keiji Horton Bkhorton@usc.edu

**Technical Requirement**

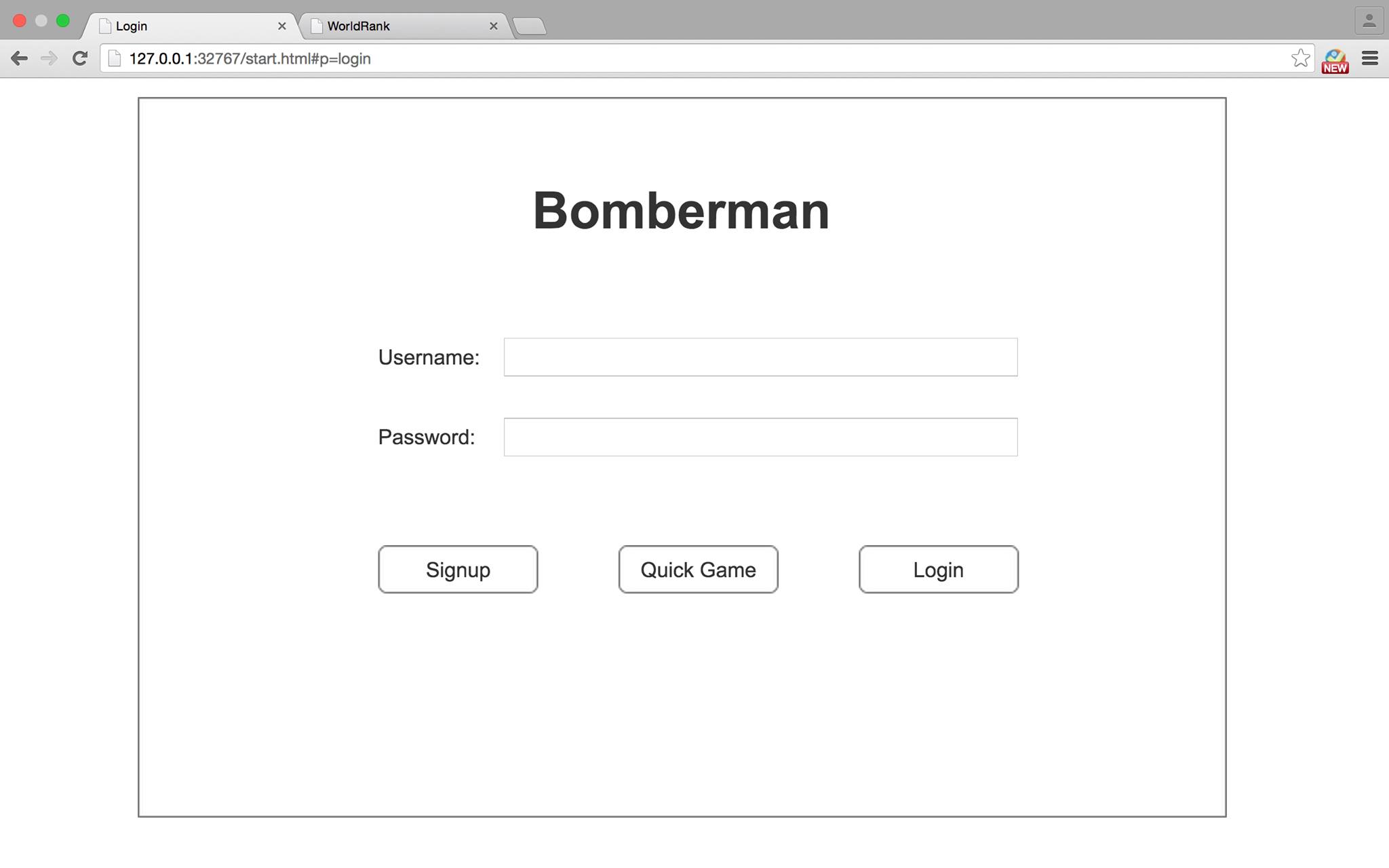
Login Page:

When the user starts the program they will be greeted with the login window. There will be 2 text lines, prompting the user for a username and password. If the player does not have an account, they can make one by pressing the “sign-up” button (this will open up a separate window).

When the user enters a username and password, then presses the “Login” button, the Login Window will close and the Main Page window will open. Their account information will be extracted/parsed from a database and loaded into the program.

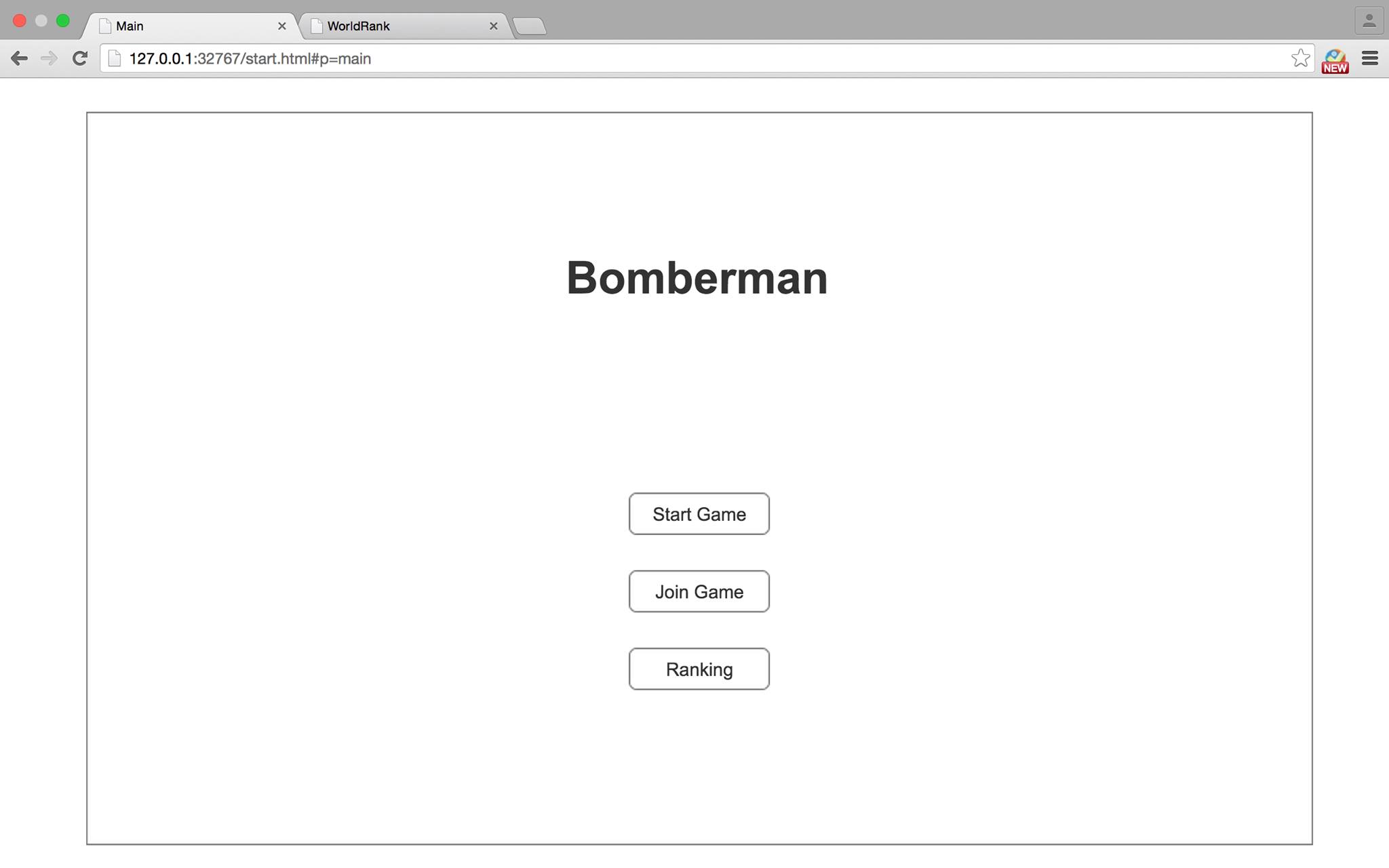
The user will also have the “Quick Game” option from this login window. If they click the quick game button, the program will start a game for the user against only NPCs. Each player gets 1 life and the game only lasts 1 minute. The player will not receive any account-related benefits during the game and his/her score will not be recorded at the end of the game. This is mostly meant for practice purposes (or for people who don’t have time for a full game, but still want to play because our game is super awesome).

There will be another text field below Password and above the buttons. That is the “secret code” textfield. Entering a secret code and sign up and let the user be a privileged member.



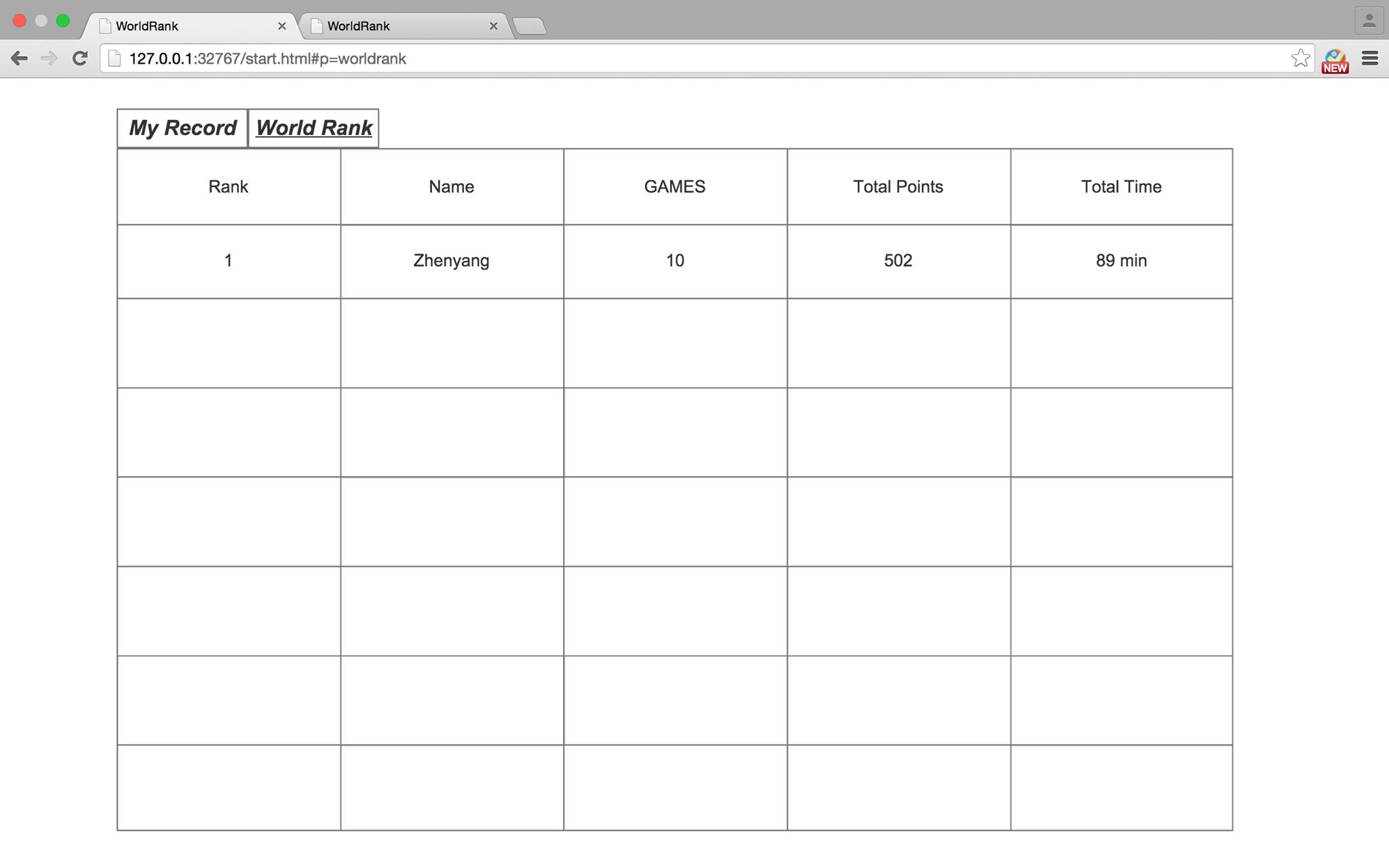
Main Page:

The player can choose “Start Game”, “Join Game”, or “Ranking” on this page. Each button will direct the player to a separate page. If the player chooses “Start Game”, the player will enter the game room as the host. The host has the privilege to start the game. If the player chooses “Join Game”, the player will join an existing room. In this case, the game will start only when the host of the room starts the game. If the player chooses “ranking”, the player can see the score ranking of all the existing players, or the player’s own score history.



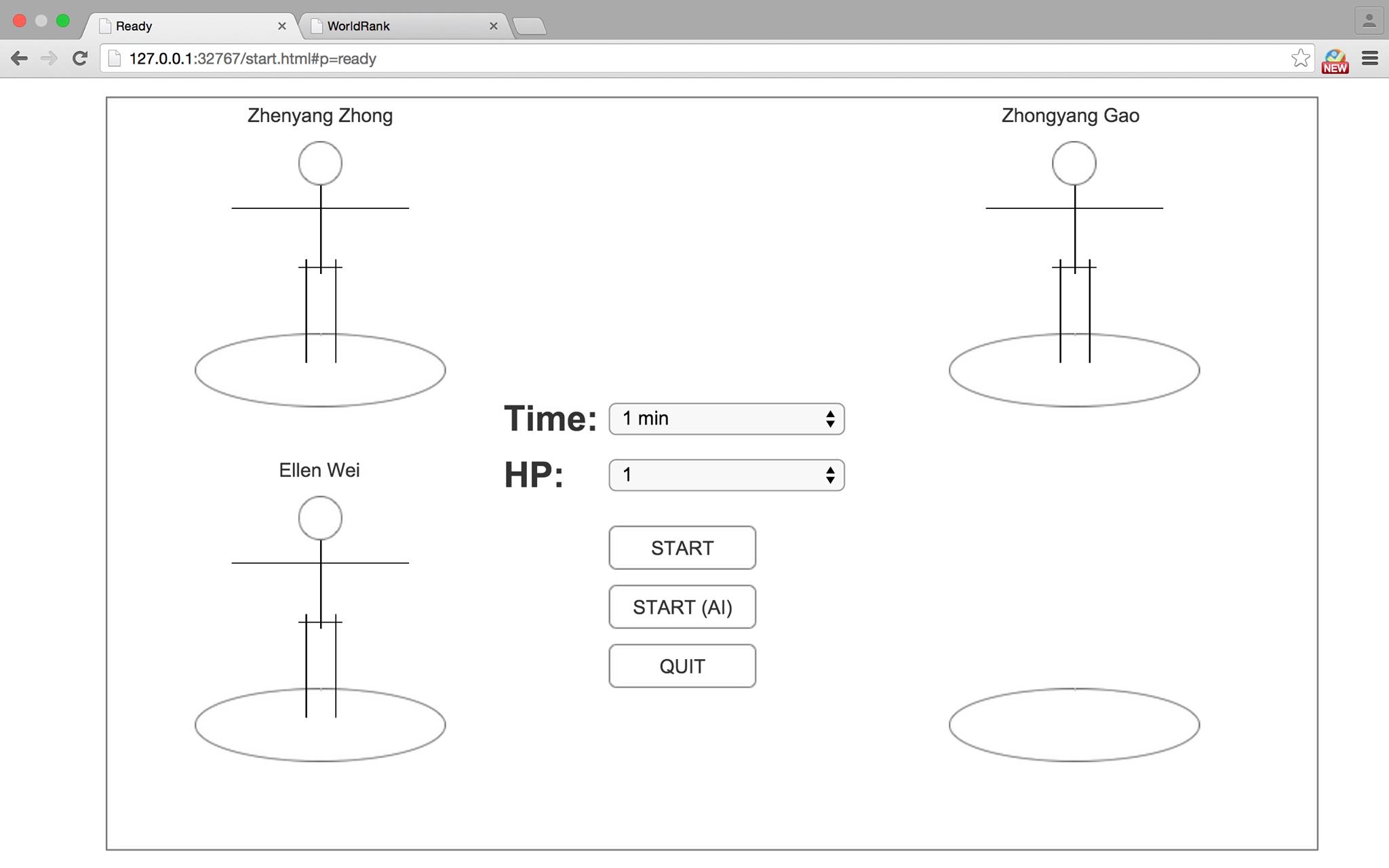
Ranking Page:

This page consists of two tabs (JTabbedPanes). “My Record” displays the player’s own score history in descending order. “World Rank” displays the overall ranking of all players participated in the game in descending order. For both pages, Rank, Name, Date, Points, and Time are all in record.



Game Room page:

The page shows all the players in the room. The host of the room is always at the top left position. The players join the room later will take the position at top right, bottom left, and bottom right in sequence. The “Start” button will only show up on host’s game room page. Only the host can start the game. If the player chooses “Quit”, it will direct the player back to Main page. If the host quits the game, all other players will be dismissed and asked to enter a different room. If the host starts the game when the room is not full, bots will take up all the empty spots.



Game Board Page:

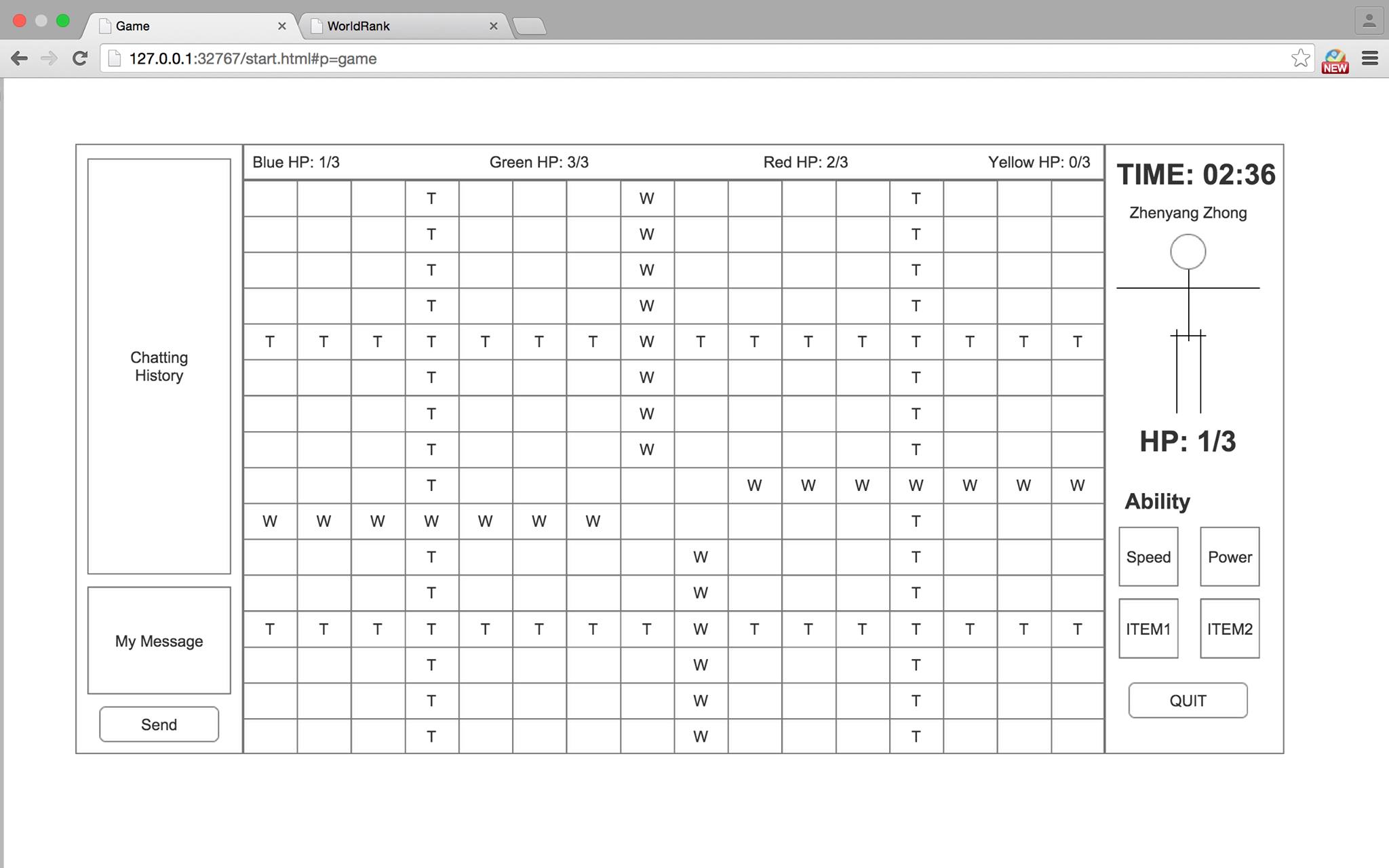
The top row displays the number of lives of all players. The main player together with his number of lives will be displayed on the right of the game board. The player can see the remaining time of the game. The “Speed” and “Power” labels will show the player’s current speed and power level. Some tools may boost up “speed” or “power”. The player can store two tools he picks up at maximum. The newest tool picked up will kick out the oldest tool in storage. Two newest tools will be stored at “Item1” and “Item2”. If the player chooses to “Quit” the game, bot will continue the game as the player. “W” in the game board stands for wall, which cannot be destroyed. “T” stands for tiles, which can be destroyed by bombs. The player can use the chatting box on the left to do live chat with other players in the game.

A privileged member will have 4 lives at the beginning.

Power-ups are: Higher move speed; Greater bomb explosion range; Faster cooling time (after dropping a bomb); Faster bomb detonating time

//Amendment 2

At start, 4 bombermen are located at the 4 corners. The user controls one of them. The user presses arrow keys (up, left, right, down) to move the bomberman to another grid (except where T or W is located at), and spacebar to drop a bomb. If the bomberman moves to a grid where an item is located at, the bomberman picks it up.



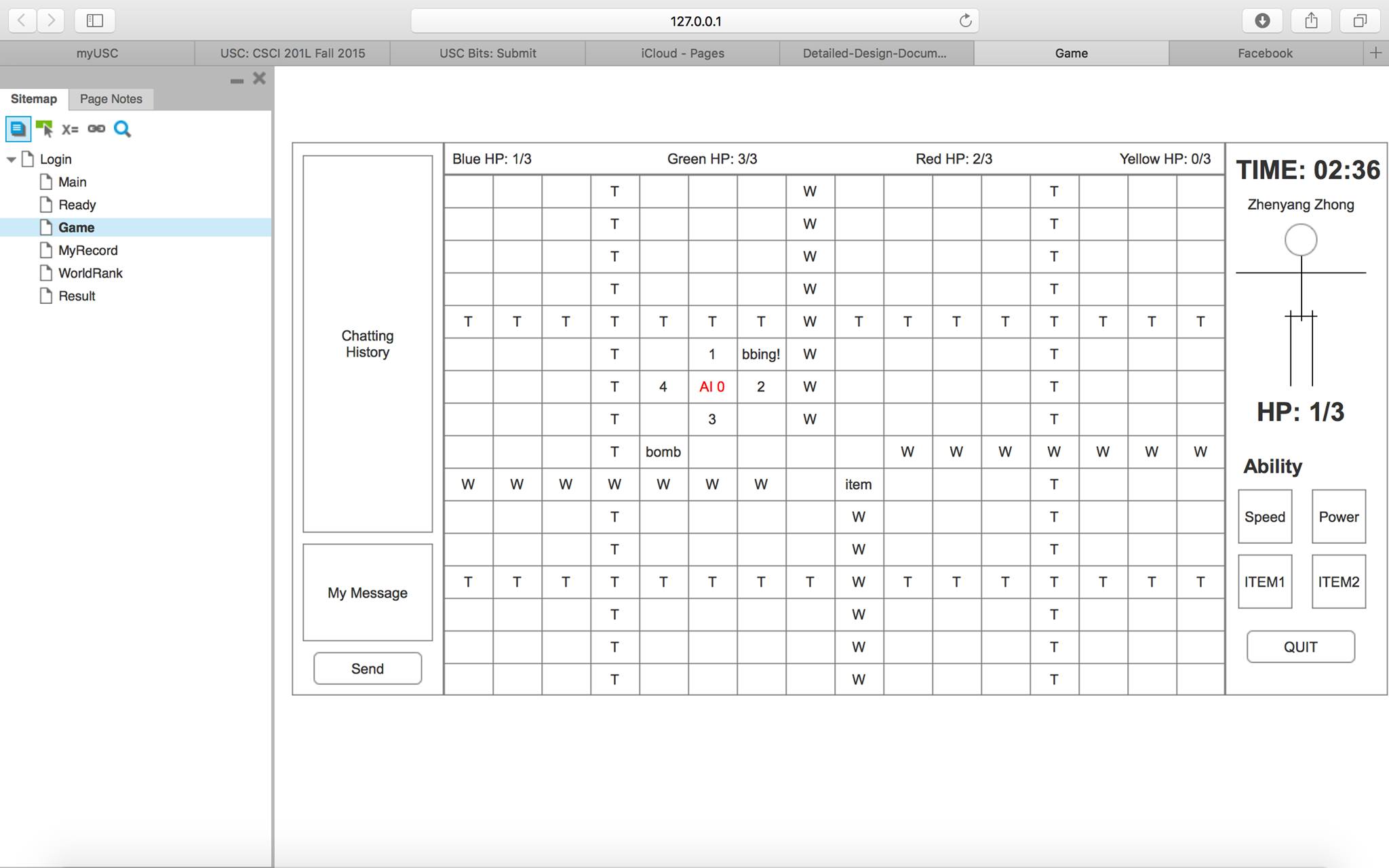
//Amendment 2

How does the bomb work?

In the diagram, there is “bbing” (bombing) near the red “AI 0.” That indicates a bomb is exploding. The blast range of the bomb is 4 (or 6, when a certain power up is picked up) spaces on above, below, left, right of and on the space where a bomb is exploding. A bomb can blast on 4\*4 + 1 spaces at most.

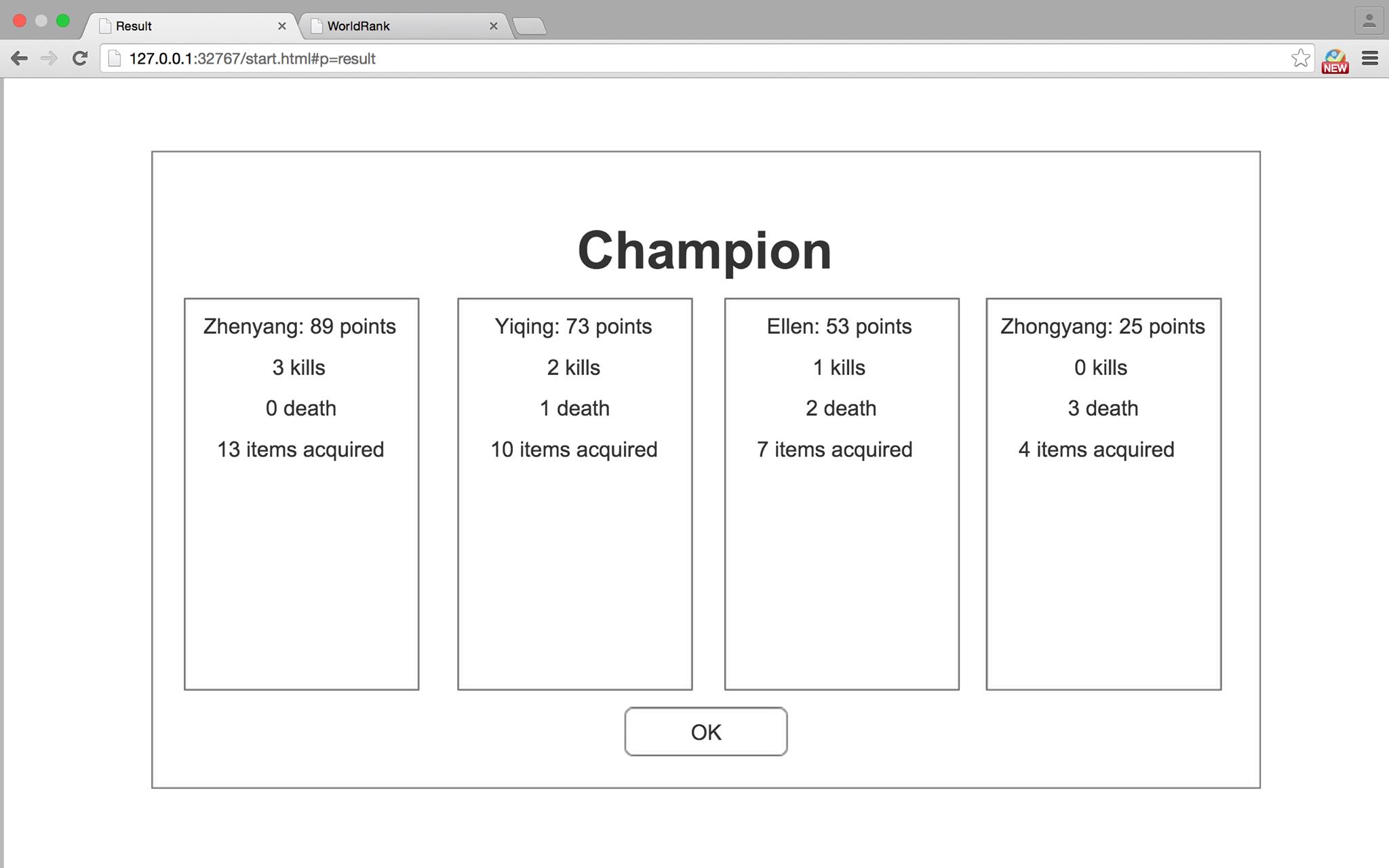
However, the bomb cannot blast through a wall. In the diagram, since there is a wall “W” right next to “bbing,” which is not destroyable, the bomb cannot blast through the wall. In other words, the player on the right of the wall will not be hit by the bomb, even the player is 2 spaces on the right of the bomb. There is another wall 3 spaces below the bomb. 2 spaces below will be blasted, and nothing happens on the wall and the space below the wall.

A bomb can destroy a tile, but if that happens, any further spaces will not be blasted. There is a tile “T” right above bombing. When the bomb explodes, the tile above is destroyed. However, the 3 spaces above the tile will not be blasted. There is another tile on 3 spaces left of the bomb. That tile and the 2 spaces on the left of the bomb will be blasted, but not the one on 4 spaces left of the bomb.



Result Page:

This page shows up when the game ends. It displays the total points, number of kills, time of deaths, and the number of items picked up of each player. If the player chooses “OK”, the player will be directed to the main page.



Hardware Requirements

Windows:

Windows 7

Windows 8

Windows 10

RAM: 128MB

Disk Space: 512MB  Processor: Minumum Pentium 2 266MHz processor

Mac OS X:

Intel-based Mac running Mac OS X 10.8.3+

Software Requirements

Java 8

Eclipse IDE for Java EE Developers(Luna/Mars)

Packages:

**Client:**

**BMGameFrame extends JFrame** － The main Frame of the game. Store all other panels the client requires.

Function:

+ static void main()

+ quit() : void

+ presentLogin() : void

+ presentMenu() : void

+ presentRoom() : void

+ presentBoard() : void

+ presentRank() : void

+ setUserId(int): void

- closeServer() : void

- refreshComponents() :void

Variable:

- JPanel: BMLoginPanel loginPanel

- JPanel: BMMenuPanel menuPanel

- JPanel: BMRoomPanel roomPanel

- JPanel: BMBoardPanel boardPanel

- JPanel: BMRankPanel rankPanel

- JPanel: BMServer server

- JPanel: BMManager mManager

- int: current user's id

- BMReceiver receiver

**BMLoginPanel extends JPanel** - Responsible for player login in. The player will type in username and password. The username and password will be sent to Simulation class. Enter the correct secret code (i.e. CSCI201) to sign up as a privileged player.

Function:

+BMLoginPanel(ActionListener loginAction, BMReceiver receiver)

-initializeVariables(): void

-createGUI() : void

-addAction(): void

-addUserId(int)

Variable:

-JLabel UsernameLabel

-JLabel PasswordLabel

-JLabel SecretCodeLabel

-JDialog ErrorDialog

-JTextField UsernameTextField

-JTextField PasswordTextField

-JTextField SecretCodeTextField

-JLabel TitleLabel

-JButton SignupButton

-JButton QuickGameButton

-JButton LoginButton

-BMReceiver receiver

**BMMenuPanel extends JPanel** - The player will choose to host or join a game, or see the ranking in this window. If the player chooses to join, they will be prompted with a JOptionPane that asks for the IP address.

Function:

+ BMMenuPanel(ActionListener MenuAction)

-initializeVariables(): void

-createGUI() : void

-addAction(): void

-hostServer() : void

-joinServer() : void

Variable:

-JLabel TitleLabel

-JButton HostGameButton

-JButton JoinGameButton

-JButton RankingButton

**BMRoomPanel extends JPane**l - The window will display all the players in the room. BMSimulation class will automatically start the game if the game room is full. If new message is received from the server, redraw() function will be called and the JPanel will repaint. The QuitButton allows the player to return the BMMenuPanel.

Function:

+BMRoomPanel(ActionListener startgameAction, ActionListener quitgameAction, BMReceiver receiver)

-initializeVariables(): void

-createGUI() : void

-addAction(): void

-redraw(): void

Variable:

-JButton StartButton

-JButton StartwithAiButton

-JButton QuitButton

-JPanel Player1Panel

-JPanel Player2Panel

-JPanel Player3Panel

-JPanel Player4Panel

-BMReceiver receiver

**BMBoardPanel extends JPanel** - The main game board of the game. It will initialize every grid of the board. The redraw() function will be be called every time the players make new moves.

Function:

+BMBoardPanel(ActionListener startAction, BMReceiver receiver)

-initializeVariables(): void

-createGUI() : void

-addAction(): void

-redraw(): void

+ endGame( new BMResultFrame() ) : void

Variable:

- JPanel ChatPanel

- JPanel GameBoardPanel

- JPanel PlayerHPPanel

- JPanel MainPlayerPanel

- Vector<BMPlayerPanel> players

- Vector<Vector<BMNodePanel>> nodes

- JPanel BMPlayerPanel(inner class)

- JPanel BMNodePanel(inner class)

- BMReceiver receiver

- BMChatClient chatreceiver

- Keylistener keylistener

**BMPlayerPanel extends JPanel** - The panel displays the player.

Function:

+ BMPlayerPanel()

+ void update(int x, int y)

Variable:

- ImageIcon

BMNodePanel extends JPanel - Display the grid of the game board.

Function:

+BMNodePanel(BMNode)

protected void update()

Variable:

- ImageIcon

**BMResultFrame extends JFrame** - Display new frame includes the final result of the game. If "OK" is clicked, the player will quit the game. The system will automatically save the results.

Function:

+BMResultFrame(ActionListener resultAction, BMReceiver receiver);

-initializeVariables(): void

-createGUI() : void

-addAction(): void

Variable:

- JLabel: TitleLabel

- BMResultPanel: NO1PlayerPanel

- BMResultPanel: NO2PlayerPanel

- BMResultPanel: NO3PlayerPanel

- BMResultPanel: NO4PlayerPanel

- JButton: OKButton

-BMReceiver receiver

**BMResultPanel extends JPanel** - Display the details of the player's score.

Function:

+BMResultPanel(BMManager mManager);

-initializeVariables(): void

-createGUI() : void

-addAction(): void

Variable:

-JLabel: PointLabel x4

-JLabel: KillLabel x4

-JLabel: DeathLabel x4

-JLabel: ItemLabel x4

**BMRankPanel extends JPanel** - Display the ranking page. World ranking will be displayed in descending order. My Record will display all the past record of the player according to date in descending order. The backButton lets the player return the MenuPanel.

Function:

+BMRankPanel(ActionListener rankAction, BMReceiver receiver)

-initializeVariables(): void

-createGUI() : void

-addAction(): void

Variable:

- JTabbedPane JTP

- JPanel WorldRankPanel

- JPanel MyRecordPanel

- JTable WorldRankTable

- JTable MyRecordTable

- DefaultTableModel: WorldRankTableModel

- DefaultTableModel: MyRecordTableModel

- JScrollPane tableScrollPane

- JButton backButton

-BMReceiver receiver

**BMChatClient extends JPanel implements Thread** - Monitor the chatting between players on the game board.

Function:

+ BMChatClient (String hostname, int port );

- void run();

Variable:

- BufferedReader br;

- PrintWriter pw;

- Socket s

**BMReceiver implements Thread** - Accept the information from the server and deliver the the direction information to the server.

In the Login Panel it sends log in information and received an acceptance/rejection from the server. In the Rank Panel it received ranking information from the server. In Room Panel it receives a signal to start a game. In Board Panel it sends the movement type of the player to the server (simply walking up, down, left, right, or dropping a bomb), and receives information in all nodes (whether there is a bomb or wall or power-up at node[x][y]). In Result Panel it receives the scores of all players.

Function:

+BMReceiver(Socket);

- initializeVariables();

+ run( ): void;

+ sendMoveType(int); void

+sendLoginInfo(String, String): void

Variable:

-Socket: mSocket

-ObjectInputStream: ois

-PrintWriter: pw

-BufferedReader: br

**Library** - Store and initialize all the constants of the game.

**BMLibrary : static**

Function:

+ BMLibrary(String)

+ getImages(String) : ImageIcon

+ getGameMap() : String[ ][ ]

Variables:

- HashMap<String, ImageIcon> : imageMap

+ static final String : SQLName

**BMNodeType : static**

Variables:

+ static final int road = 0

+ static final int wall = 1

+ static final int tile = 2

+ static final int bomb = 3

+ static final int bombing = 4

+ static final int niceShoes = 5

+ static final int badShoes = 6

+ static final int improvePower = 7

+ static final int reducePower = 8

+ static final int reduceCoolingTime = 9

+ static final int increaseCoolingTime = 10

+ static final int increaseDetonatedTime = 11

+ static final int reduceDenotatedTime = 12

+ static final int noCoolingTime = 13

**BMMove : static**

Variables:

+ static final int stop = 0

+ static final int up = 1

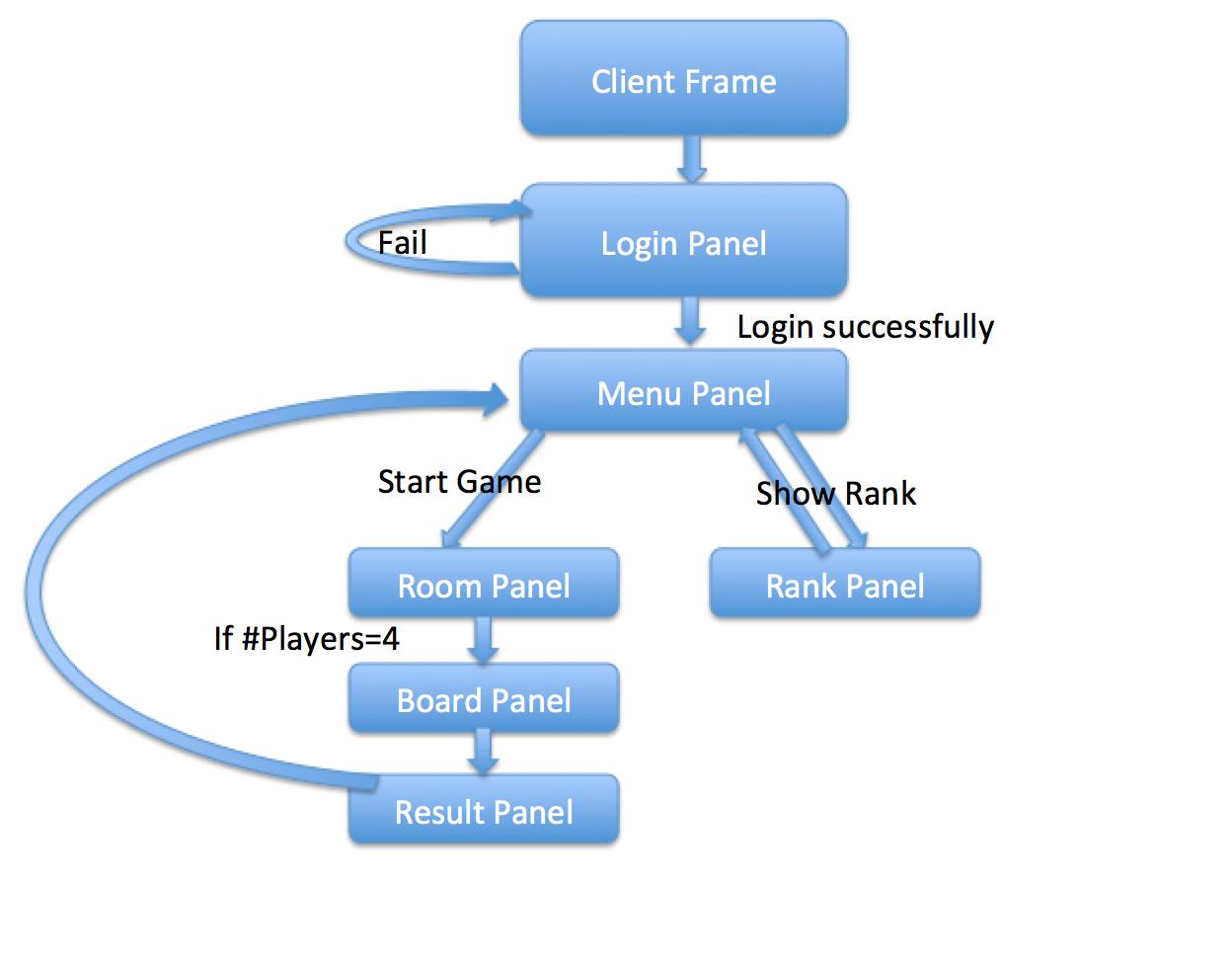
+ static final int down = 2

+ static final int left = 3

+ static final int right = 4

+ static final int bomb = 5

Case Diagram



**Server**

**BMServer** - Store all the game information.

Function:

+ BMServer()

- login(name,password) : Boolean

- signup(name,password) : Boolean

- addRoom() : void

Variable:

- BMSimulation simulation

- SocketServer ss

- Vector<BMPlayer> : clients

**BMSimulation implements Thread** - In charge of all the information of one round of the game.

Function:

+ BMSimulation(BMPlayer host, String ip)

+ getHPs() : String[ ]

+ getResults() : String[ ]

+ canMove(int x, int y) : Boolean

+ getNode(Point p) : BMNode

+ dropBomb(int x, int y) : void

+ setNode(int x, int y, BMNode node) : void

+ sendMessage(int id, string msg) : void

+ getGameData() : string // get the data of the game in json\_encode

- getHP(int id) : String

- getResult(int id) : String

- startGame() : void

Variable:

- BMNode[ ][ ] : nodes

- Vector<BMPlayer> players

- BMPlayer host

- Time : timeLeft

- BMChatServer : chatServer

- RatingPeriodResults result

- BMRankTable rankTable

- String ip

**abstract BMPlayer implements Thread** - Store all the information of a player. The functions support all possible actions of the player.

Functions:

+ BMPlayer(int ID, int initialLives)

+ setSimulation(BMSimulation simulation)

protected killed() : void //should check at every moves

protected getLocation() : Point

protected getItemsPossessed() : Vector<BMItem>

protected getNumOfItemsAcquired() : int

protected getLivesLeft() : int

protected getInitialLives() : int

protected getNumOfPlayerKilled() : int

protected startMove(int move) : void

protected getCurrentNode(Point p) : Node

protected getSpeed() : int

protected getPower() : int

protected getCoolingTime() : int

protected getDetonatedTime() : int

- canMove() : Boolean

Variables:

- Point location

- int speed

- int power

- int coolingTime

- int detonatedTime

- Vector<BMItem> items

- int deaths

- int initialLives

- int kills

- int ID

- bool lost

- BMSimulation simulation

**BMRealPlayer extends BMPlayer** - The human player of the game. Implement all the data variables and functions of BMPlayer class.

Function:

+ BMRealPlayer()

+ getMessage(string msg) : void

- run() : void

Variables:

- BurfferedReader : br

- Socket : s

- PrintWriter : pw

**BMAIPlayer extends BMPlayer** - The AI players of the game. Besides all the functions of BMPlayer, AIPlayer has its own algorithm to make the best move.

Function:

- run() : void

- aiMove() : void

- aiGetBestTry : int

- aiTry(int move) : int

**BMChatServer implements Thread** - In charge of the chatting between different players.

Function:

+ BMChatServer() : void

+ removeChatThread() : void

+ sendMsg(BMChatThread ct, String str)

Variable:

Vector<BMChatThread> : ctVector

**BMChatThread implements Thread** - Support chatting between different players.

Function:

+ ChatThread(Socket s, ChatServer cs)

+ sendMsg(String msg) : void

+ run() : void

Variable:

- BufferedReader br

- PrintWriter pw

- ChatServer cs

- Socket s

**abstract BMNode** - Display each grid in the game board. Vanish will support the explosion on the game board.

Function:

+ BMNode(int x, int y)

+ vanish() : Boolean // set board[x][y] to be new

BMBombing(x,y,BMNodeType.road), return true

Variable:

+ final int x

+ final int y

+ int type

+ final Boolean canPass

**BMRoad extends BMNode** - Display the road on the game board.

Function:

+ BMRoad(int x, int y)

Variable:

+ final int type = BMNodeType.road

+ final Boolean canPass = true

**BMWall extends BMNode** - Display the wall on the game board.

Function:

+ BMWall(int x, int y)

+ vanish() : Boolean // just return false

Variable:

+ final int type = BMNodeType.wall

+ final Boolean canPass = false

**BMTile extends BMNode** - The bricks can be destroyed by the player.

Function:

+ BMWall(int x, int y)

+ vanish() : Boolean // 30% get item, 70 percent get nothing, set board[x][y] to be new BMBombing(x,y,id), return false

Variable:

+ final int type = BMNodeType.tile

+ final Boolean canPass = false

**BMBomb extends BMNode implements Thread** - The bomb owned by each player to destroy the obstacles and other players.

Function:

+ BMBomb(int x, int y, int power)

+ vanish() : Boolean // bomb! return true

- run():void

Variable:

+ final int type = BMNodeType.bomb

+ final Boolean canPass = false

+ final int power

**BMBombing extends BMNode implements Thread** - The animation when the bomb explodes.

Function:

+ BMBombing(int x, int y, int afterBombing)

+ vanish() : Boolean // return true extend the bombing time to 0.3

+ end() : void // set board[x][y] to be the afterBombing type node

- run() : void //should last 0.3 second

Variable:

+ final int type = BMNodeType.bombing

+ final Boolean canPass = true

- int afterBombing

**BMItem extends BMNode** - The items the player can pick up during the way.

Function:

+ BMBombing(int x, int y, int type)

Variable:

+ final int type

+ final Boolean canPass = truec

**BMRankTable** - Stores player ranks, calculates new rank, connects to MySQL

Function:

+BMRankTable()

+getName(int id) : String

+getRank(int id) : int

-updateRank(RatingPeriodResults) : void

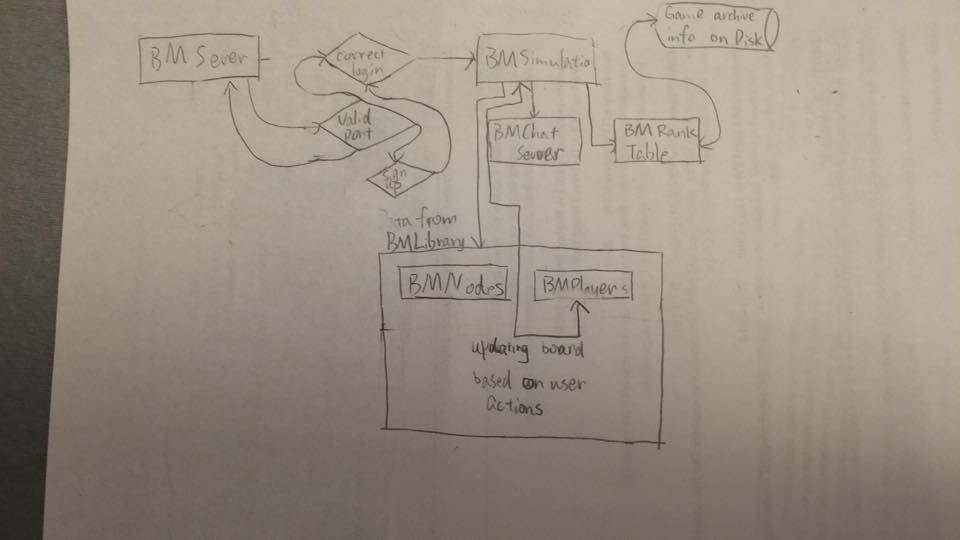
Variable:

- Connection conn

- Statement st

- RatingCalculator ratingcalculator

Case Diagram



**Utilities**

**RatingCalculator** - Calculates rating

Function:

+ RatingCalculator()

+ updateRatings(RatingPeriodResults results) : void

Variables:

- double default\_rating

- double default\_deviation

- double default\_volativity

**RatingPeriodResults** - Stores temporary results

Function:

+ RatingPeriodResults()

+ addResult(Rating winner, Rating loser)

+ addDraw(Rating player1, Rating player2)

Variables:

- List<Result> results

**Result** - Result between 2 players

Function:

+ Result(Rating winner, Rating loser)

+ Result(Rating player1, Rating player2, boolean isDraw)

Variable:

- boolean isDraw

- Rating winner

- Rating loser

**Rating** - Rating of 1 player

Function:

+ Rating(int id, RatingCalculator)

+ getRating() : double

+ setRating(double rating) : void

Variable:

- double rating

- double ratingDeviation

- double volatility

The above 4 classes are modified from Jeremy Gooch's work.

Copyright (C) 2013 Jeremy Gooch http://www.linkedin.com/in/jeremygooch/

<https://github.com/goochjs/glicko2>

Inheritance

BMNode

BMNode

BMNode

BMNode

BMNode

BMNode

BMNode

BMAIPlayer

BMRealPlayer

BMPlayer

ER Diagram

|  |
| --- |
| ***Game Record*** |
| Int ID |
| Int killCount |
| Int deathCount |
| Time time |

|  |
| --- |
| ***Rank*** |
| Int ID |
| Double rating |
| Double ratingdeviation |
| Double volativity |

n

1

An account has multiple game records.

1

1

An account has a rank

|  |
| --- |
| ***Account*** |
| Int ID |
| String username |
| String password |
| Boolean  isVIP |

//Amendment

AI Algorithm

The AI drops a bomb whenever it can. Since there is a period of cooling time after the player drops the bomb, AI can’t drop bomb endlessly without recess. The algorithm assumes that the AI will drop a bomb whenever it can, yet only consider 5 possible movements: up, down, left, right, or stay at the same space.

In the following diagram, the AI is labelled red. Option 0 is remain at the same space, and 1,2,3,4 representing up, right, down, left respectively. For each option we calculate a rating. The option with the highest rating will be chosen. There are 4 factors that change the ratings:

1. A bomb is blasting on that space: -100
2. A bomb has not exploded, but will blast that space later:

-10\*(range of bomb – distance from bomb)

1. Distance of the closest item (BFS shortest path):

+ (rows + columns – distance from item)^2

1. Distance of the closest player (BFS shortest path):

+ (rows + columns – distance from player)^2 \* 0.3

In the following example, the board has 5 rows and 6 columns. Let’s say the bomb can blast 4 spaces above, below, left, and right itself.

Option 0: There is no bomb blasting on it, nor any bombs that will blast it later. 5 spaces from the item and 3 from the player. The rating is: 0 + 0 + (5+6-5)^2 + (5+6-3)^2\*0.3 = 55.2

Option 1: -100 + 0 + (5+6-6)^2 + (5+6-4)^2\*0.3 = -60.3

Option 2: -100 + 0 + (5+6-4)^2 + (5+6-4)^2\*0.3 = -36.3

Option 3: 0 + 0 + (5+6-4)^2 + (5+6-2)^2\*0.3 = 73.3 (chosen move)

Option 4: 0 – 10\*(4-2) + (5+6-6)^2 + (5+6-4)^2\*0.3 = 19.7

0,0 0,5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | 1 | bbing |  |  |
|  | 4 | AI,0 | 2 | W |  |
|  |  | 3 |  | W |  |
|  | bomb |  |  |  |  |
|  |  | Player |  | Item |  |

4,0 4,5

//Amendment

The secret code to register as a VIP player will be disclosed to the player only when the player meets both of the following two criteria:

1. The player has killed 20+players accumulatively
2. The player is not a VIP player

The database stores every player’s kill count. It will be shown in the BMResultFrame after the player finishes a game and has accumulated 20 kills. When a player has accumulated 20 kills, after every game the secret code will be shown to the player. There is only one secret code which can be shared.

Test cases

Main Page/Login Test Cases:

|  |  |
| --- | --- |
| Test # | 1 |
| Test Description | Only players, who’s coupled username and passwords exist in the User Database, can log in |
| Steps to run test | 1. Start up the login menu 2. Type a valid/invalid username into the “Username:” text-line 3. Type a valid/invalid password into the “Password:” text-line 4. Click the Login button |
| Expected Result | If the username and password are valid, meaning they exist in the database and are a pair (password belongs to the username), the game should close the login window/panel and open up the Main Page window/panel.  If the username and password combo are invalid (the user doesn’t exist, or the password is incorrect), then a pop up message will notify the user of this and not allow the program to proceed. |

|  |  |
| --- | --- |
| Test # | 2 |
| Test Description | When logged in, the correct user information is loaded into the program |
| Steps to run test | 1. Log In from the login window 2. From the Main Page, click the Rankings Button |
| Expected Result | If the proper user was loaded into the system, their username should be visible in the top left corner of the Main Page. Also, in the Rankings panel, under “My Record,” the appropriate information of the user’s record should be visible |

|  |  |
| --- | --- |
| Test # | 3 |
| Test Description | New user can only sign up with a username that doesn’t exist in the database |
| Steps to run test | 1. Start up program 2. Type in a desired username in the username text-line 3. Type in a desired password in the password text-line 4. Click the SignUp button |
| Expected Result | If the username exists in the system, then a pop up dialog will notify the user that that name is taken. The typed in username should NOT be added to the database (check the database for duplicates). If the username doesn’t exist, then a pop up dialog will notify the user that their account has been created. The user database should then have this new entry in it. |

|  |  |
| --- | --- |
| Test # | 4 |
| Test Description | A user tries to login twice with the same username to the server |
| Steps to run test | 1. Log In from the login window 2. Start another program and log in with the same username and password |
| Expected Result | A pop up window showing “You have been logged in to the server already.” |

Ranking Test Cases:

|  |  |
| --- | --- |
| Test # | 5 |
| Test Description | Personal Rankings should be displayed properly |
| Steps to run test | 1. Log in as a valid user (who has played through several games) 2. From the main page, click the Rankings button 3. Make sure the My Record tab is chosen |
| Expected Result | A list of rankings for only the logged-in user should be displayed. They should be listed in descending order (highest to lowest) |

|  |  |
| --- | --- |
| Test # | 6 |
| Test Description | World Rankings should be displayed properly |
| Steps to run test | 1. Log in as any user 2. From the main page, click the Rankings button 3. Make sure the World Records tab is selected 4. Repeat as different user |
| Expected Result | List of top 10 ranked players should be listed in descending order (highest to lowest rank). This list should be the same, regardless of which user is logged in. |

Hosting and Joining:

|  |  |
| --- | --- |
| Test # | 7 |
| Test Description | The user can host a game for other players to join |
| Steps to run test | 1. Log in 2. Press the Start Game button 3. Enter port number in the pop up dialog |
| Expected Result | When the first user (the host) enters their port number, their program/game should create a server that hosts the game. The Game Room (basically a waiting room) panel should show, where they will wait for more users to join. |

|  |  |
| --- | --- |
| Test # | 8 |
| Test Description | User can join a game that has been setup and is in the waiting stage |
| Steps to run test | 1. Log in 2. Press the Join Game Button 3. Enter the ip and port # for the server you’d like to join |
| Expected Result | If the host starts up their server before the joiner tries to join, then the Joiner will be put into the waiting room with the host and any other players in it. The GameRoom panel should be displayed. |

|  |  |
| --- | --- |
| Test # | 9 |
| Test Description | Only the host can start the game and adjust time limit/HP |
| Steps to run test | 1. One user should login and host a game 2. A second user should login and join that game 3. Have the host adjust the Timer/HP values once or twice |
| Expected Result | The option (buttons) to start the game, with or without AI, should only appear on the Host’s Game room panel. The Time and HP option panes should be editable by the Host.  The person who joined the game should have their Start Game buttons disabled in the Game Room window. Instead of an option pane, only a non-editable label with the Time and HP should appear on the joiners Game Room Panel. When the host changes the Time/HP values, the respective labels should change from the Joiners’ point of view |

|  |  |
| --- | --- |
| Test # | 10 |
| Test Description | The players currently waiting in the Game Room should be visible to everyone in the waiting room |
| Steps to run test | 1. Let a player host the game 2. Several players should join the host’s game (up to 2) |
| Expected Result | The Game Room should be visible to all the players. The avatars for each player in the waiting room should be visible. At least one spot should be vacant (since only up to 2 other players joined for the purposes of this test). |

|  |  |
| --- | --- |
| Test # | 11 |
| Test Description | User cannot join a game that is currently running |
| Steps to run test | 1. Have a host setup their own game (with or without AI) 2. Start up the game 3. Have another user attempt to join the game |
| Expected Result | A pop up dialog should appear on the Joining user’s screen, notifying them that the game is currently running. Their program will not proceed past the Main Page. |

|  |  |
| --- | --- |
| Test # | 12 |
| Test Description | Users cannot join games that are full |
| Steps to run test | 1. Have a host setup their own game, but stay in the waiting room 2. Have 3 other players join the Host’s server 3. Have 1 last player attempt to join |
| Expected Result | The last player should get a pop up dialog, notifying them that the Game Room is full for the server they attempted to join |

|  |  |
| --- | --- |
| Test # | 13 |
| Test Description | Join a game at an IP where no game is being host |
| Steps to run test | 1. Login in and make sure there is no other clients logged in 2. Join Game and insert an IP |
| Expected Result | A pop up dialog should appear, notifying the user that no game is being hosted currently at that particular IP |

Starting the Game:

|  |  |
| --- | --- |
| Test # | 14 |
| Test Description | Starting the game regularly (without AI) will initiate a game with just the players currently in the Game Room |
| Steps to run test | 1. Have a host set up a game 2. Have up to 2 players join the host’s Game Room. 3. Host must press the “Start” button |
| Expected Result | A game with only the number of players in the waiting/Game room should initiate. Vacant spots are left vacant |

|  |  |
| --- | --- |
| Test # | 15 |
| Test Description | Starting a game with AI will fill any vacant spots in the waiting room with AI players |
| Steps to run test | 1. Have a host set up a game 2. Have 0-2 players join the host’s game 3. Host presses “Start with AI” button |
| Expected Result | A game should initiate, where the vacant spots are filled with AI players. There should always be 4 players total in the game |

|  |  |
| --- | --- |
| Test # | 16 |
| Test Description | Host can set up Time limit for game (default 1 minute) |
| Steps to run test | 1. Setup a Game Room 2. Have the Host change the Timer value 3. Start game (with or without AI) 4. Play game, but have no player kill anyone/attempt to win 5. Repeat, but keep the default time |
| Expected Result | Timer in the Game should read 1 minute by default. If the host changed it, the Timer should say the altered Time, and count down from there. The game should end when the Timer reaches 0. |

|  |  |
| --- | --- |
| Test # | 17 |
| Test Description | The Host can adjust the number of lives each player has in game (default to 1) |
| Steps to run test | 1. Setup a Game Room 2. Have host change the HP options 3. Start game 4. Have one player dies N times (N = HP value selected in Game Room) 5. Repeat, but don’t adjust the HP value |
| Expected Result | In game, the players should all have the same number of lives that was specified in the HP option pane. After a player dies N times, they will not be able to play, and must watch the game until it ends. |

Playing the Game:

|  |  |
| --- | --- |
| Test # | 18 |
| Test Description | A clients loads a BMLibrary map text file and image files |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. 3. Wait till the game starts |
| Expected Result | The Game Board Page shows up correctly with walls, tiles, players, and bombs. |

|  |  |
| --- | --- |
| Test # | 19 |
| Test Description | A player can only walk on valid paths (no going over walls or obstacles) |
| Steps to run test | 1. Start a game 2. Have a player press the arrow keys in the desired direction of movement 3. Do this in cases where the path of desired movement contains  * A wall * An obstacle * An open path |
| Expected Result | If the tile in the direction of movement contains an open path, the player’s avatar will move there.  If the tile contains a wall or obstacle, the avatar will remain on their current tile |

|  |  |
| --- | --- |
| Test # | 20 |
| Test Description | A player can destroy obstacle tiles with bombs, unless the obstacle is a Wall |
| Steps to run test | 1. Start up a game 2. Have a player without powerups drop a bomb by obstacles and walls 3. Wait till the bomb explodes |
| Expected Result | All obstacles within 4 tiles up, down, left, and right of the bomb should be destroyed. The previously occupied tiles should then become open paths.  The only things that remain unaffected are walls. |

|  |  |
| --- | --- |
| Test # | 21 |
| Test Description | Power ups will boost the player’s characteristics accordingly |
| Steps to run test | 1. Start up a game 2. Have a player walk over a power up icon 3. Have the player move or drop bombs |
| Expected Result | Once the player walks over the icon, it should disappear (since it’s been used up).  The results of each respective power up are:   * Higher Move Speed: Player traverses tiles at 2x speed * Greater Bomb Range: Increases range of bomb explosion to 6 tiles * Faster cooling Time: Halves the time required to place another bomb * Faster Bomb detonation: Halves the time it takes for a bomb to explode after it’s been placed |

|  |  |
| --- | --- |
| Test # | 22 |
| Test Description | If player is bombed, he will lose 1 HP |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game 3. Wait till the game starts 4. On the game board, either deliberately go close to other players waiting to be bombed, or ask the other test person to deliberately bomb the player.   (Simulate multi players on the same PC to test) |
| Expected Result | If the player gets 1 HP deduced when bombed, the HP on the right side of the game board will show 1 HP deduction automatically. Losing 1 HP when bombed means the player’s HP is getting less. The player will die when his HP is 0. |

|  |  |
| --- | --- |
| Test # | 23 |
| Test Description | A game ends when only one or no player has HP>0 or there is no time left. |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host choose the minimum time duration for the game 3. Wait till the game starts 4. On the game board, deliberately bomb one player every time till only one player has HP>0 / do nothing till the game ends   (simulate multiple players on the same PC to test) |
| Expected Result | The game should end automatically when only one player has HP>0 or there is no time left. The game will pop up the window saying “End of the game” and showing the winner of the game or there is a tie. |

|  |  |
| --- | --- |
| Test # | 24 |
| Test Description | If User quits during the game, AI will replace it. |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host choose the minimum time duration for the game 3. Wait till the game starts 4. On the game board, make one player quit the game   (simulate multiple players on the same PC to test) |
| Expected Result | If the player quits the game, a confirmation box will pop up to make sure the player actually wants to quit the game; the game will freeze at that stage. If the player actually quits, the game will automatically generate an AI to take the place of the player just quitted. |

|  |  |
| --- | --- |
| Test # | 25 |
| Test Description | The chat box will correctly display the message sent by players. |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host choose the minimum time duration for the game 3. Wait till the game starts 4. On the game board, send message from every player’s window 5. (simulate multiple players on the same PC to test) |
| Expected Result | All the chat messages displayed in the chat box are displayed in time order and the content is exactly what the players sent |

|  |  |
| --- | --- |
| Test # | 26 |
| Test Description | The messages in the chatting history will automatically be updated. |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host choose the minimum time duration for the game 3. Wait till the game starts 4. On the game board, send message from every player’s window   (simulate multiple players on the same PC to test) |
| Expected Result | All the chat messages will automatically be updated in the chat box, and the chat box will never crash or freeze during the game. The players who already be killed can also send messages. |

|  |  |
| --- | --- |
| Test # | 27 |
| Test Description | The items boxes will correctly display the latest two items picked up by the player. |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host choose the minimum time duration for the game 3. Wait till the game starts 4. On the game board, one player picks up all the item on the way. Make sure other players don’t bomb him   (simulate multiple players on the same PC to test) |
| Expected Result | No matter how many items the player picks up, only the latest two items picked up will be shown on the right side of the game board in the items box. The item box is able to auto-update. |

|  |  |
| --- | --- |
| Test # | 28 |
| Test Description | Time is counting down correctly during the game. |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host choose the minimum time duration for the game 3. Wait till the game starts 4. On the game board, make sure the player doesn’t make any move   (simulate multiple players on the same PC to test) |
| Expected Result | The clock on the right side of the game board should be counting down when time elapses. |

|  |  |
| --- | --- |
| Test # | 29 |
| Test Description | The game will end when time expires. |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host choose the minimum time duration for the game 3. Wait till the game starts 4. On the game board, make sure the player doesn’t make any move   (simulate multiple players on the same PC to test) |
| Expected Result | The game will end when time ends. |

|  |  |
| --- | --- |
| Test # | 30 |
| Test Description | If the player is killed, he will observing the game until the game ends. |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host choose the minimum time duration for the game 3. Wait till the game starts 4. On the game board, deliberately bomb one player till he is dead   (simulate multiple players on the same PC to test) |
| Expected Result | After the player is dead, he will be notified by the system that he is dead. The player is able to stay in the game as an observer. |

|  |  |
| --- | --- |
| Test # | 31 |
| Test Description | The player being killed can also send out message during the game. |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host choose the minimum time duration for the game 3. Wait till the game starts 4. On the game board, deliberately bomb one player till he is dead   (simulate multiple players on the same PC to test) |
| Expected Result | After the player is dead, he will be notified by the system that he is dead. The player is still able to send messages to chat. |

|  |  |
| --- | --- |
| Test # | 32 |
| Test Description | The host sends BMGameBoard configuration every fixed amount time to clients for update. |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host choose the minimum time duration for the game 3. Wait till the game starts 4. Play the game |
| Expected Result | Same animations should be seen at the host and its clients’ screen. A low amount delay is allowed. |

Game ends:

|  |  |
| --- | --- |
| Test # | 33 |
| Test Description | Result page will pop up automatically when a game with AI ends. |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host chooses the minimum time duration for the game, and the host should ensure there are <4 human players. 3. Wait till the game starts 4. Wait till the game ends |
| Expected Result | The Result Page pops up when the game ends. |

|  |  |
| --- | --- |
| Test # | 34 |
| Test Description | Result page will pop up automatically when a game without AI ends. |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host chooses the minimum time duration for the game, and the host should ensure there are 4 human players. 3. Wait till the game starts 4. Wait till the game ends |
| Expected Result | The Result Page pops up when the game ends. |

|  |  |
| --- | --- |
| Test # | 35 |
| Test Description | The Result page will show the correct result of this game and current player will be highlighted. |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host chooses the minimum time duration for the game, and the host should ensure there are 4 human players. 3. Wait till the game starts 4. Wait till the game ends 5. Check the result page |
| Expected Result | The Result page shows all data correctly, and the current player's data is highlighted. |

|  |  |
| --- | --- |
| Test # | 37 |
| Test Description | Rank is updated after a game |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host choose the minimum time duration for the game, and the host should ensure there is 4 human players. 3. Wait till the game starts, hence till it ends 4. Click “OK” on the result page 5. Click “Rankings” on the main page. 6. Check “My record” and “World rank” |
| Expected Result | My record and World Rank both update correctly even when other players are still in the "result page." |

|  |  |
| --- | --- |
| Test # | 36 |
| Test Description | “OK” button on Result page directs the user back to main page |
| Steps to run test | 1. Login from the Login window 2. Either choose join or host the game. The host choose the minimum time duration for the game, and the host should ensure there are 4 human players. 3. Wait till the game starts 4. Wait till the game ends 5. Click “OK” on the result page |
| Expected Result | Return from the result page to main page. |

On the server side:

|  |  |
| --- | --- |
| Test # | 38 |
| Test Description | Turn off MySQL when the server tries to connect to it |
| Steps to run test | 1. Turn off MySQL 2. Run a BMCentralServer 3. Input the name of the database 4. Press "connect" |
| Expected Result | Show error message "cannot connect to MySQL" |

|  |  |
| --- | --- |
| Test # | 39 |
| Test Description | A Server can connect to MySQL server and correctly retrieve player's login and ranking information. |
| Steps to run test | 1. Run MySQL 2. Run a BMCentralServer 3. Input the name of the database 4. Press "connect" 5. Print all tables in command line |
| Expected Result | Verify command line outputs with tables in the MySQL application |

|  |  |
| --- | --- |
| Test # | 40 |
| Test Description | Register a new user |
| Steps to run test | 1. Run MySQL, a BMCentralServer, and input the name of the database 2. Press "connect” 3. Run a BMClient, input a username and password and sign up |
| Expected Result | Show error message "username exists" or show sign up successfully. |

|  |  |
| --- | --- |
| Test # | 41 |
| Test Description | Two servers should not run on the same port |
| Steps to run test | 1. Run central server  2. Input the port number  3. Run another server  4. input the same port |
| Expected Result | Show error message "port in use” in the second server (run in step 3). |