EE4023 Distributed Systems

GROUP PROJECT REPORT

Submitted by

John Long	12132306
Naichuan Zhang	18111521

BSc. in Computer Systems

University of Limerick

November 2019

Table of Contents

Introduction

In this project report, we present a Tic-Tac-Toe online multiplayer game that uses the web services provided in Java. The project provides two interfaces - one written as a Java desktop application based on JFrame, and another one is a PHP driven website with SOAP.

The project is aimed at allowing the user playing Tic-Tac-Toe between them. The game will be played by many clients from different desktop. The users are required to register and login before playing the game. In addition, the project will also provide a score- board system and leader board system for users. The score system keeps track of the number of wins, losses and draws the logged in user has, and the leader board is used to illustrate the statistics for each registered user in the system.

Project Plan

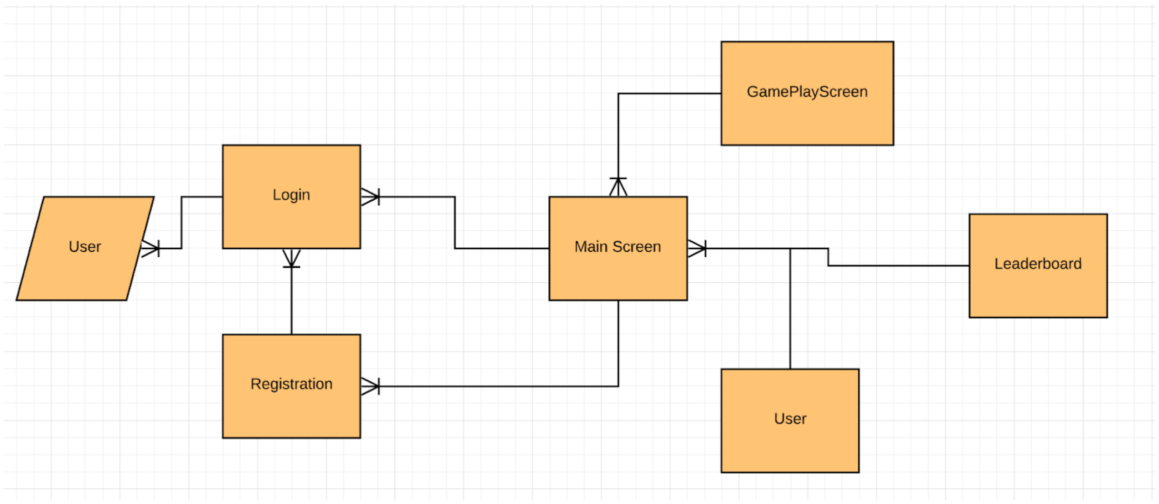
Deliverable	Responsibility
Java Desktop Application	John Long
PHP-Driven Website	Naichuan Zhang
Group Project Report	John Long / Naichuan Zhang

Design

Java Part

Swing Data Flow Diagram

Login and Registration



Upon running the program players are presented with the login screen where they have the option to login using their username and password if they have already registered or else they can click the register button where they are brought to the registration screen and can register their details there. If the player enters a valid login or registration they are then brought to the main screen.

Contained in the main screen users can see a list of open games and buttons for accessing a players own stats for wins, draws and losses, a leaderboard tracking all players wins draws and losses and a button that starts a new TicTacToe button.

From the main screen players can click on one of the games in the open games list and join it but only if they are not the creator of the game.

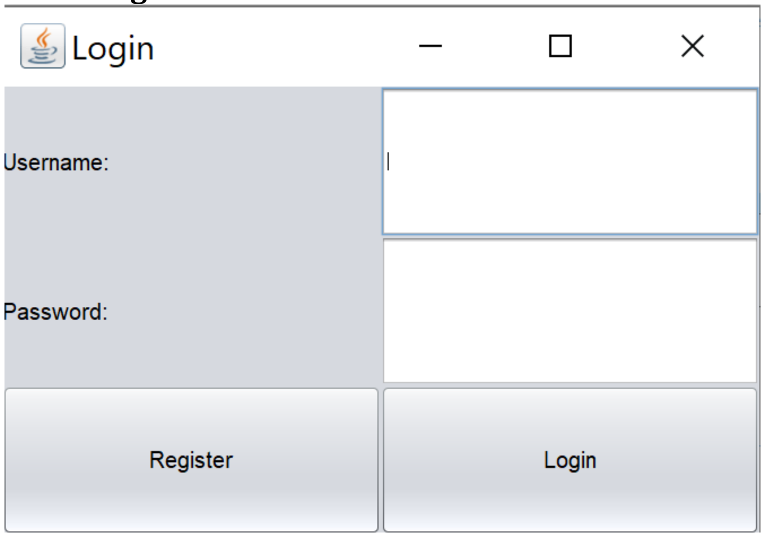
If a player clicks on the player stats button they are presented with a new screen where they can see their total wins, draws and losses.

If a player clicks on the leaderboard button they are shown a table that has all the players and their wins, draws and losses for each player.

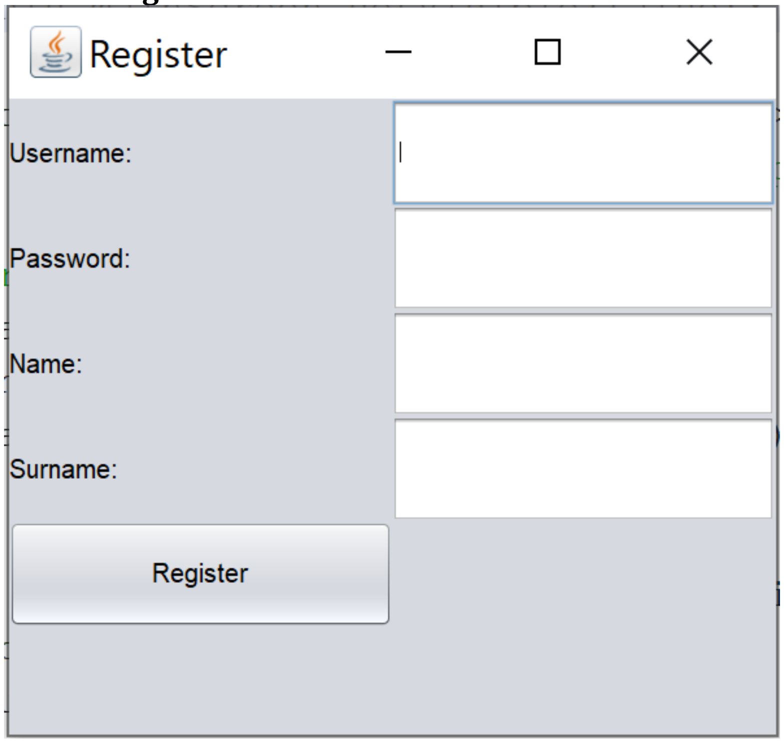
When a player wants to start a new game they can click the new game button which opens a new window and has 9 buttons organised in a grid. Once they have a second player to play with they can start the game and place their moves on the board. Once a game winning state is reached the game ends.

<u>Screenshots of Swing GUI</u>

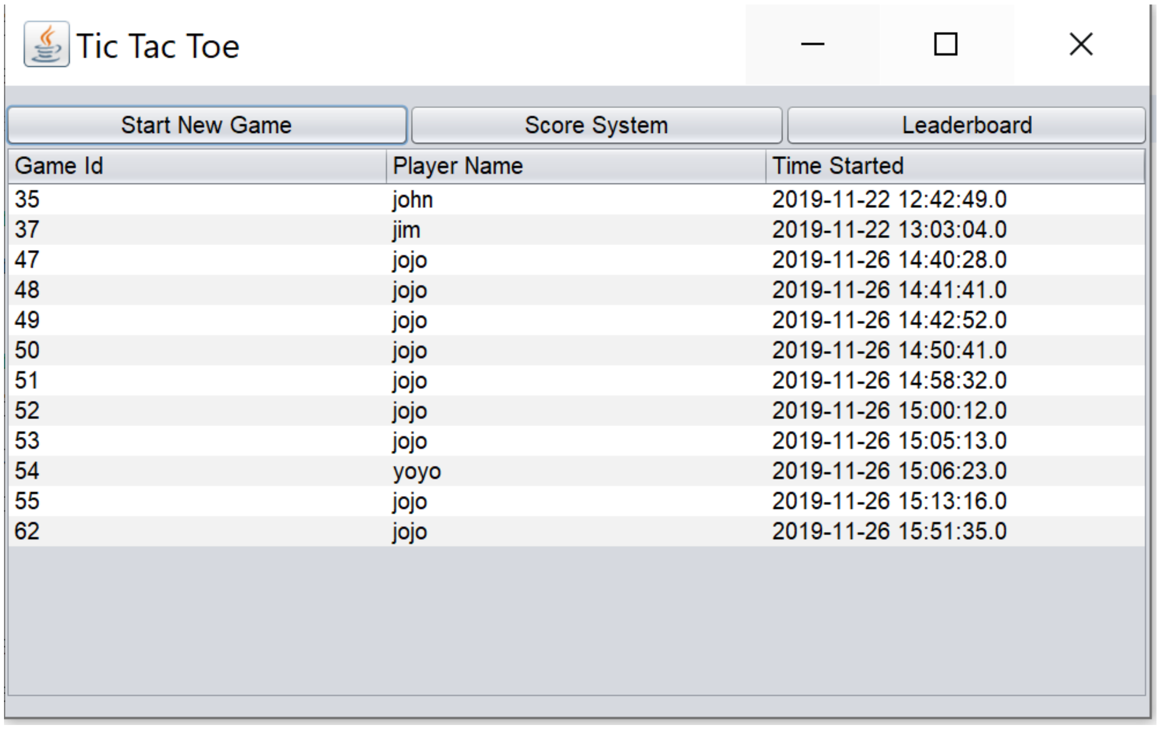
Login



Registration

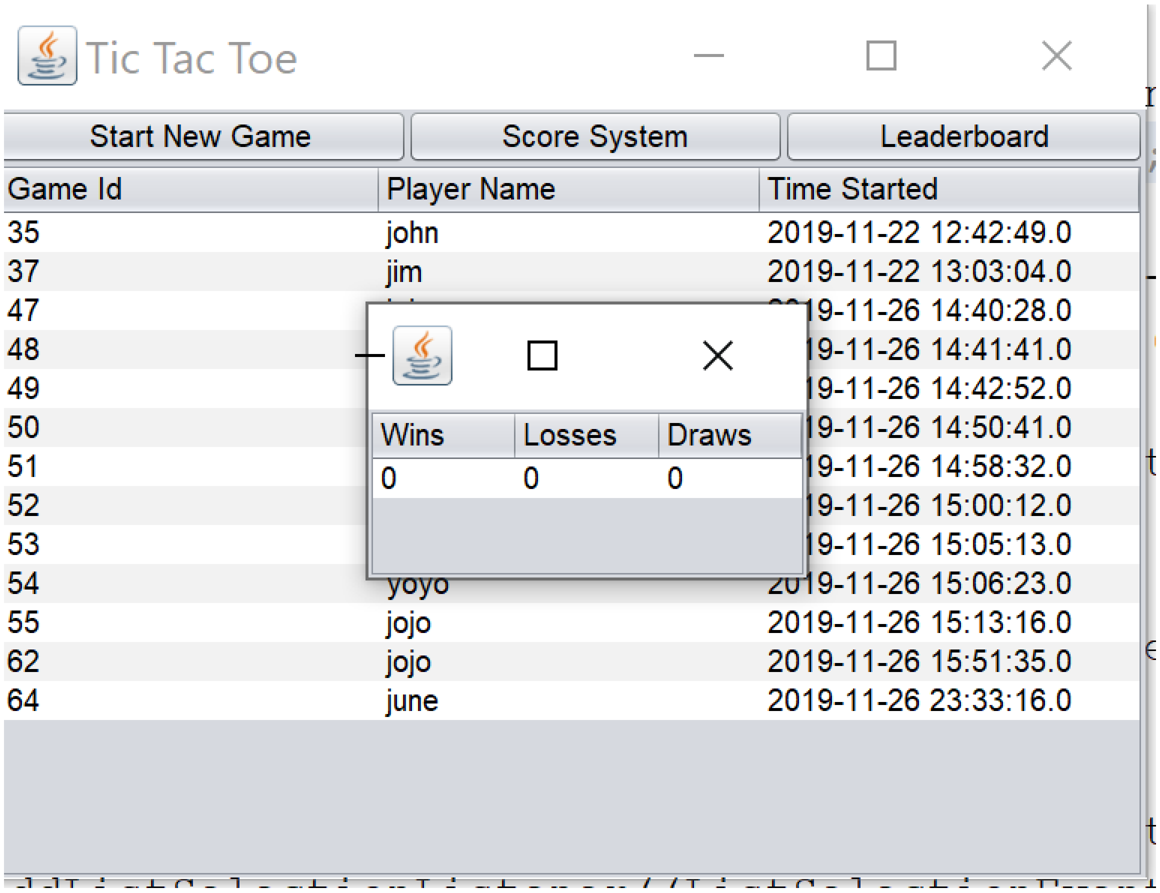


Main Screen



Game Id	Player Name	Time Started
35	john	2019-11-22 12:42:49.0
37	jim	2019-11-22 13:03:04.0
47	jojo	2019-11-26 14:40:28.0
48	jojo	2019-11-26 14:41:41.0
49	jojo	2019-11-26 14:42:52.0
50	jojo	2019-11-26 14:50:41.0
51	jojo	2019-11-26 14:58:32.0
52	jojo	2019-11-26 15:00:12.0
53	jojo	2019-11-26 15:05:13.0
54	yoyo	2019-11-26 15:06:23.0
55	jojo	2019-11-26 15:13:16.0
62	jojo	2019-11-26 15:51:35.0

User Stats Screen



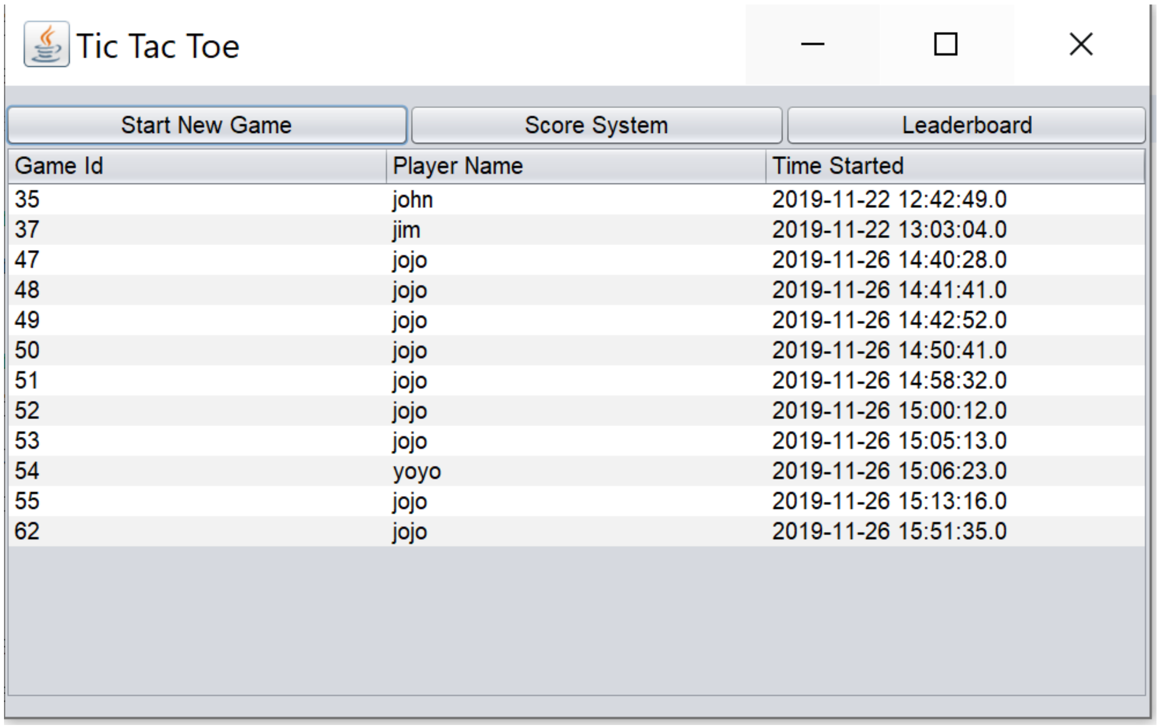
Tic Tac Toe

Start New Game | Score System | Leaderboard

Game Id	Player Name	Time Started
35	john	2019-11-22 12:42:49.0
37	jim	2019-11-22 13:03:04.0
47		19-11-26 14:40:28.0
48		19-11-26 14:41:41.0
49		19-11-26 14:42:52.0
50		19-11-26 14:50:41.0
51		19-11-26 14:58:32.0
52		19-11-26 15:00:12.0
53		19-11-26 15:05:13.0
54	yoyo	2019-11-26 15:06:23.0
55	jojo	2019-11-26 15:13:16.0
62	jojo	2019-11-26 15:51:35.0
64	june	2019-11-26 23:33:16.0

Wins	Losses	Draws
0	0	0

User Leaderboard Screen

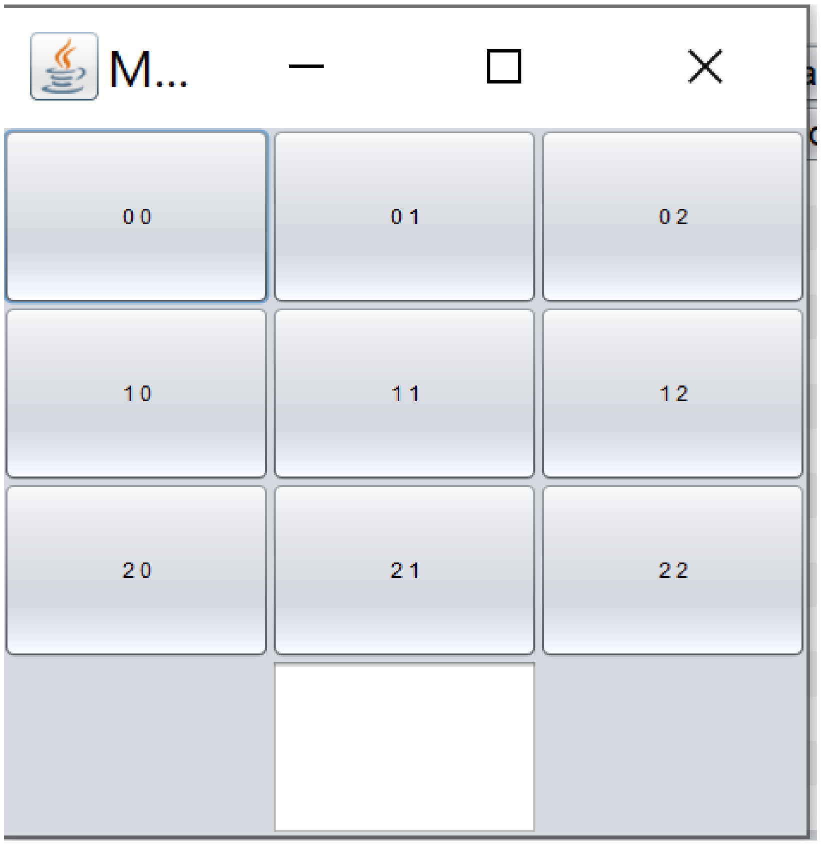


Tic Tac Toe

Start New Game | Score System | Leaderboard

Game Id	Player Name	Time Started
35	john	2019-11-22 12:42:49.0
37	jim	2019-11-22 13:03:04.0
47	jojo	2019-11-26 14:40:28.0
48	jojo	2019-11-26 14:41:41.0
49	jojo	2019-11-26 14:42:52.0
50	jojo	2019-11-26 14:50:41.0
51	jojo	2019-11-26 14:58:32.0
52	jojo	2019-11-26 15:00:12.0
53	jojo	2019-11-26 15:05:13.0
54	yoyo	2019-11-26 15:06:23.0
55	jojo	2019-11-26 15:13:16.0
62	jojo	2019-11-26 15:51:35.0

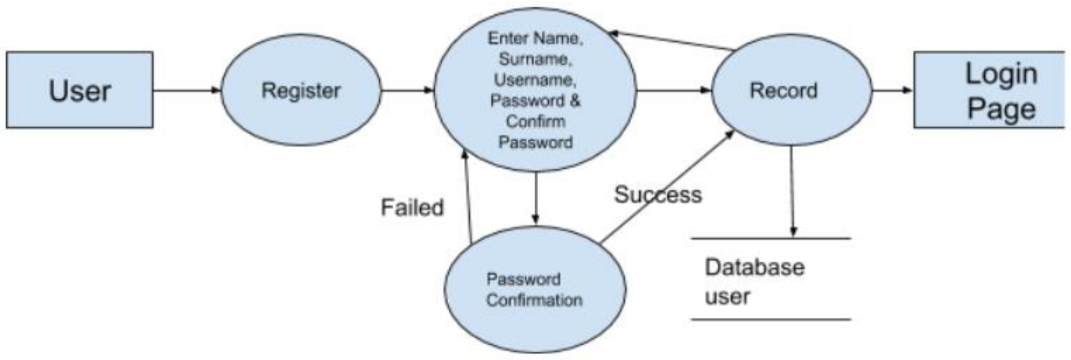
New Game Screen



PHP Part

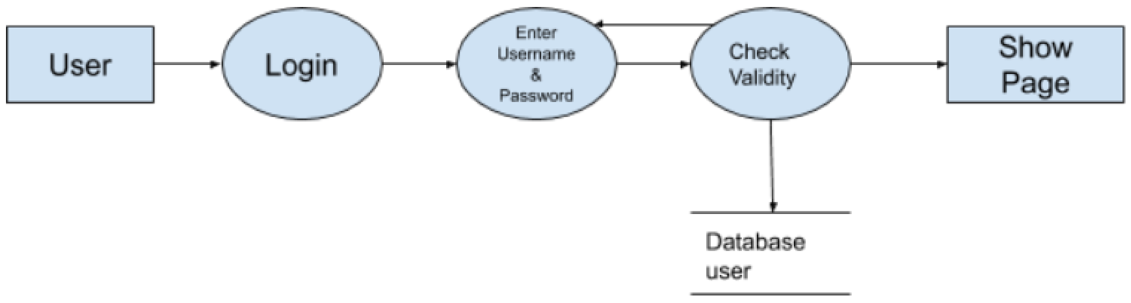
Data Flow Diagrams

Registration



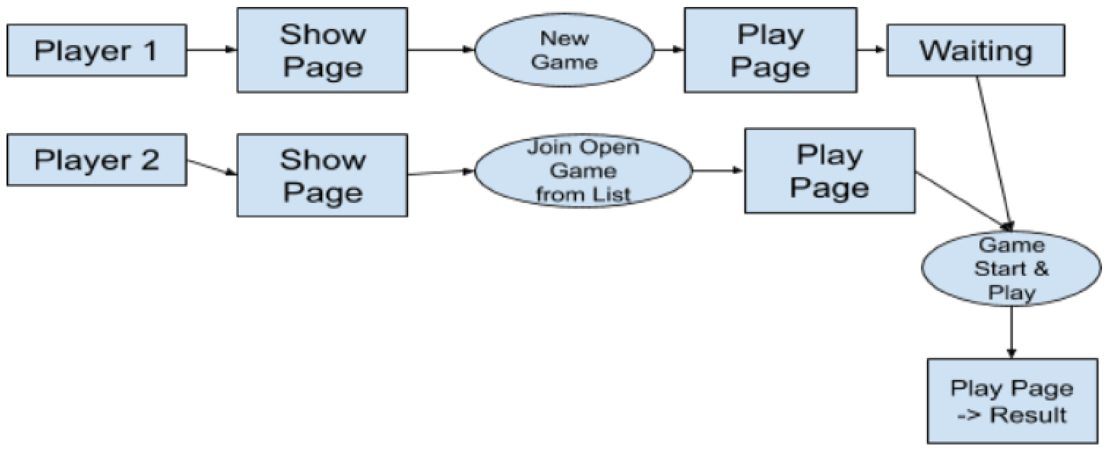
For a new user who hasn't registered yet, after entering to the Home page of the website, he/she can press Register button to register. The website will show him/her a Registration page to allow him/her to enter name, surname, username, password and confirm password. If the password confirmation succeeds, the website will redirect to the Login page; otherwise, the system will ask user to register again.

Login



After entering to the Home page of the website, if a user already has a username and password, then he/she can press Login button and a Login page will be shown to the user. A user is required to enter username and password, and if he/she is a valid user then the system will redirect to the Show page.

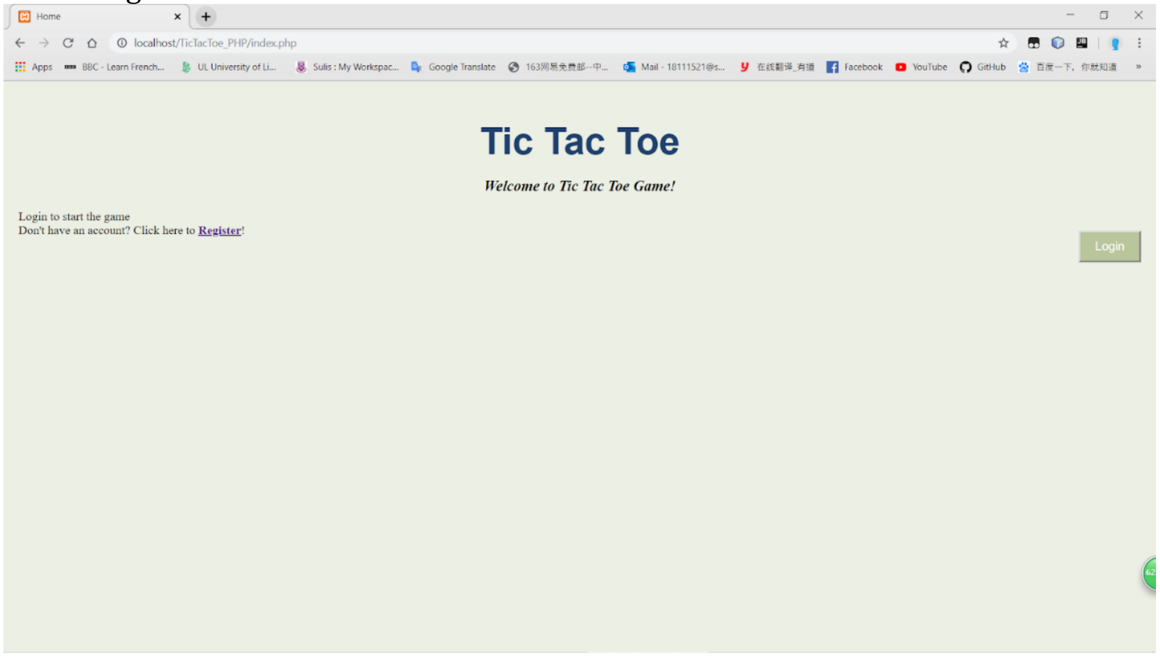
Play Game



After the user logs in, a Show page will be shown to the user. The show page contains a button to start a new game, a button to access the score system, a button to access the leader board system and a list of open games that the user can join. When a user create a new game by pressing the New Game button, he/she will become the game creator, and the system will bring him/her to the Play page. The Play page has a game board and a message to ask the user to wait for another player to join in. At the same time, another user can join the game from the open game list shown in the Show page. When the second user join the game with the same game id. The system will show another message to tell the game creator to take the first move, and the second user waits. If a user hasn't taken a move, the game board of another user will always be locked. When the game is over, another message will be shown on the Play page to display final results to both users.

Screenshots

Home Page

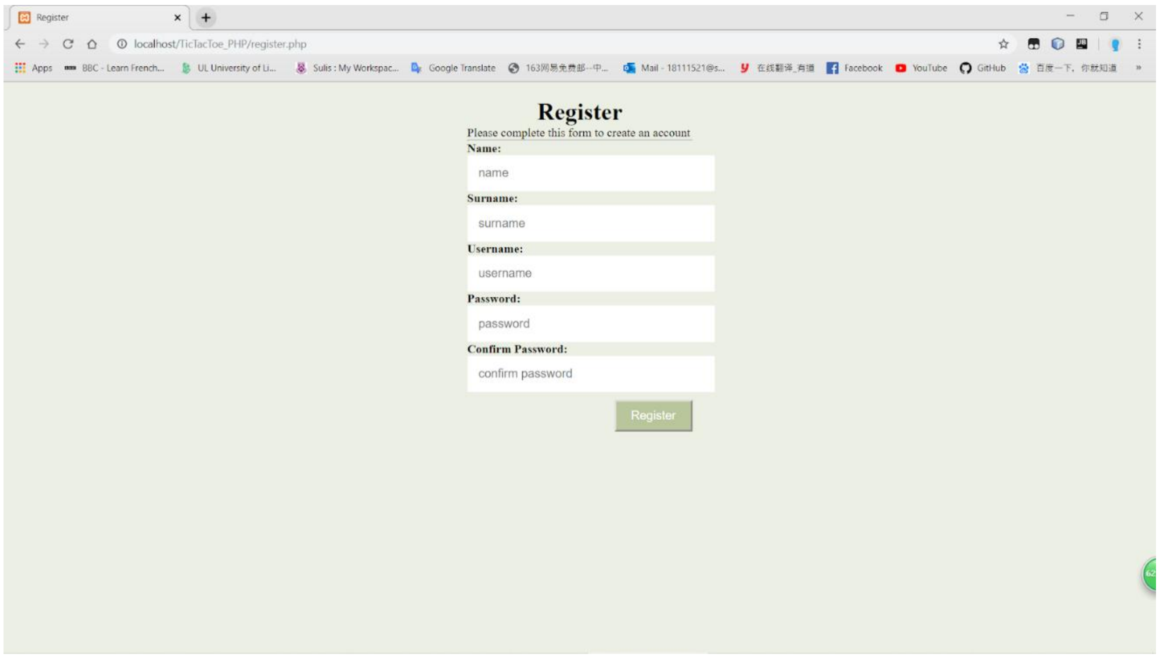


Tic Tac Toe

Welcome to Tic Tac Toe Game!

Login to start the game
Don't have an account? Click here to Register!

Registration Page



Register

Please complete this form to create an account

Name:

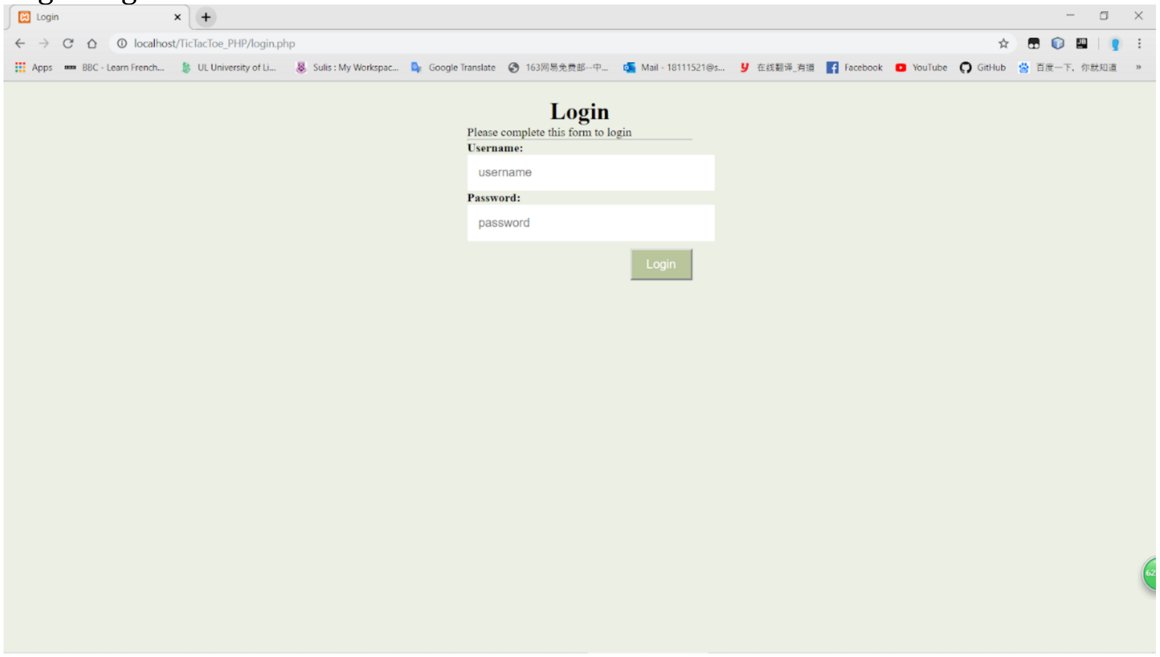
Surname:

Username:

Password:

Confirm Password:

Login Page



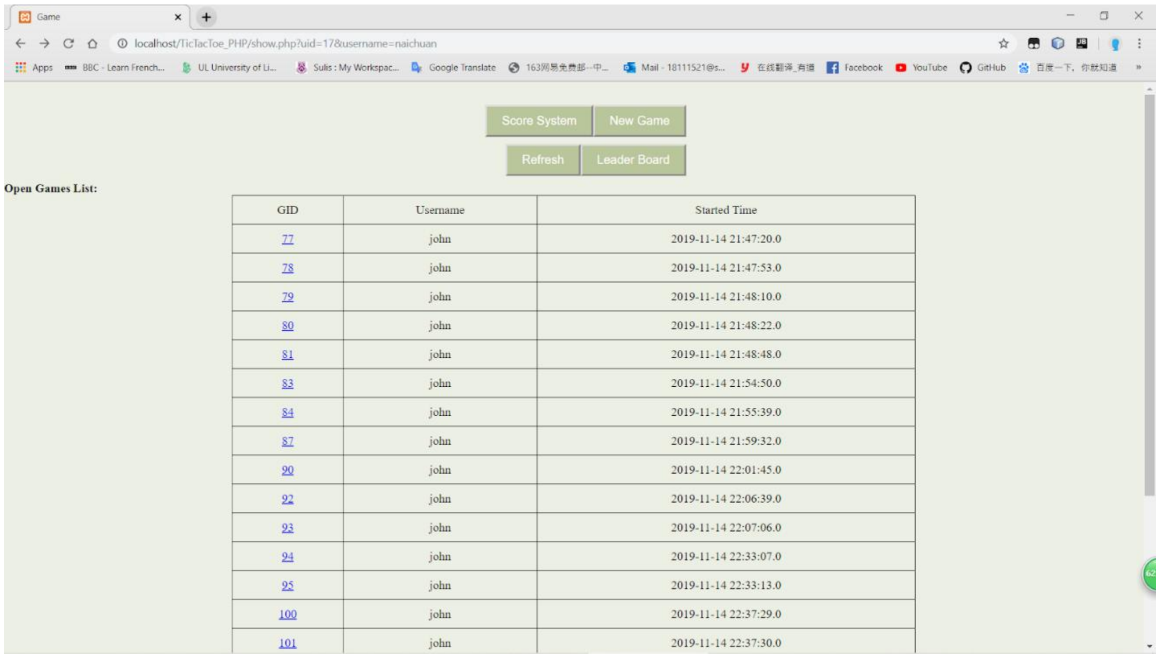
Login

Please complete this form to login

Username:

Password:

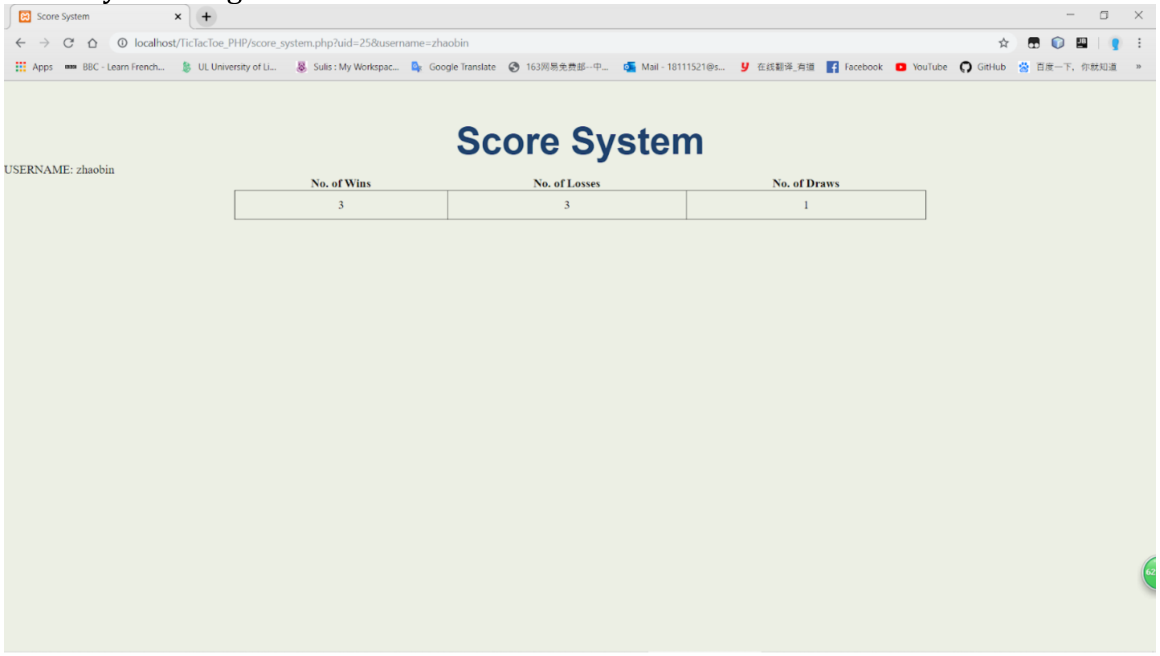
Show Page



Open Games List:

GID	Username	Started Time
77	john	2019-11-14 21:47:20.0
78	john	2019-11-14 21:47:53.0
79	john	2019-11-14 21:48:10.0
80	john	2019-11-14 21:48:22.0
81	john	2019-11-14 21:48:48.0
83	john	2019-11-14 21:54:50.0
84	john	2019-11-14 21:55:39.0
87	john	2019-11-14 21:59:32.0
90	john	2019-11-14 22:01:45.0
92	john	2019-11-14 22:06:39.0
93	john	2019-11-14 22:07:06.0
94	john	2019-11-14 22:33:07.0
95	john	2019-11-14 22:33:13.0
100	john	2019-11-14 22:37:29.0
101	john	2019-11-14 22:37:30.0

Score System Page

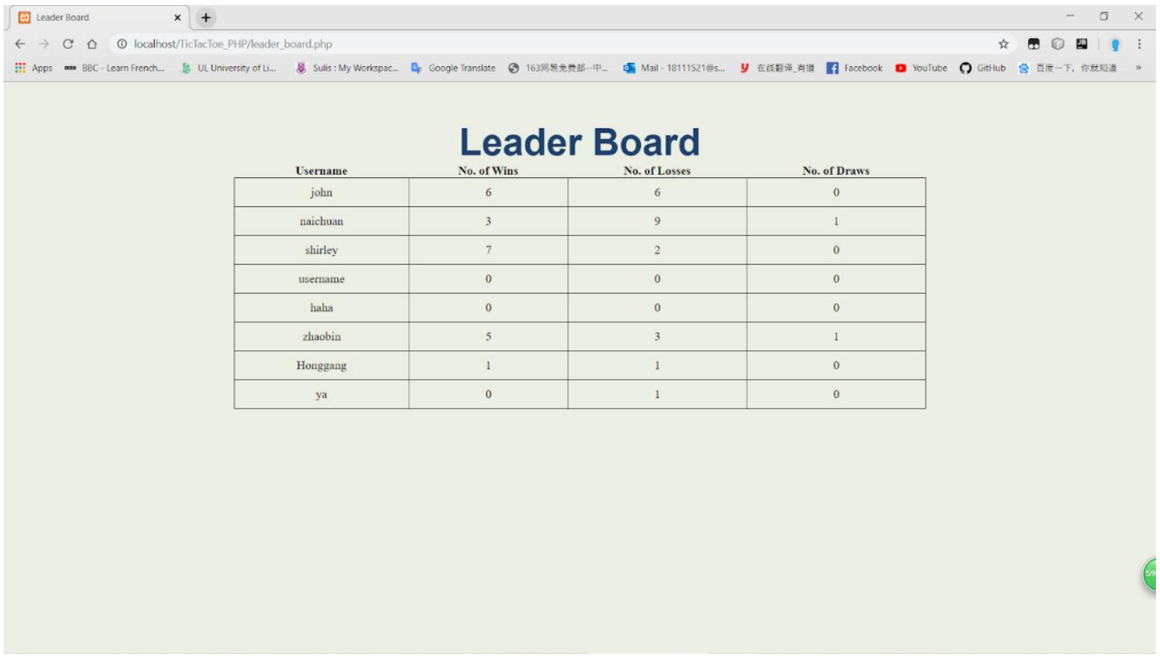

Score System
localhost/TicTacToe_PHP/score_system.php?uid=25&username=zhaobin

Score System

USERNAME: zhaobin

No. of Wins	No. of Losses	No. of Draws
3	3	1

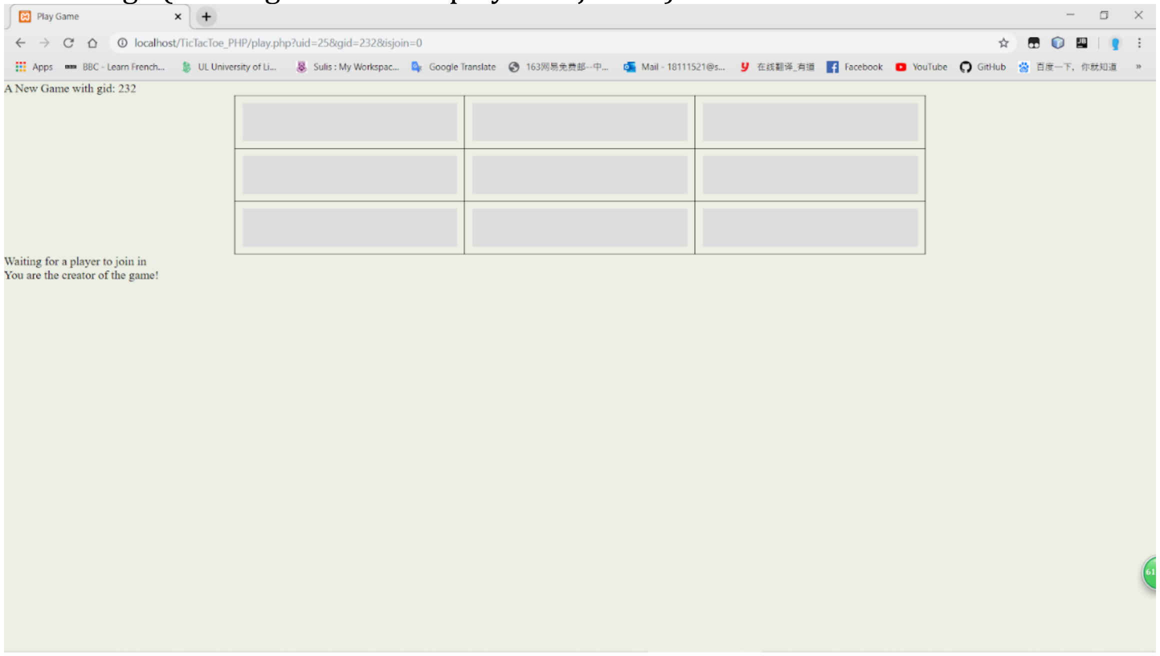
Leader Board Page



Leader Board

Username	No. of Wins	No. of Losses	No. of Draws
john	6	6	0
naichuan	3	9	1
shirley	7	2	0
username	0	0	0
haha	0	0	0
zhaobin	5	3	1
Honggang	1	1	0
ya	0	1	0

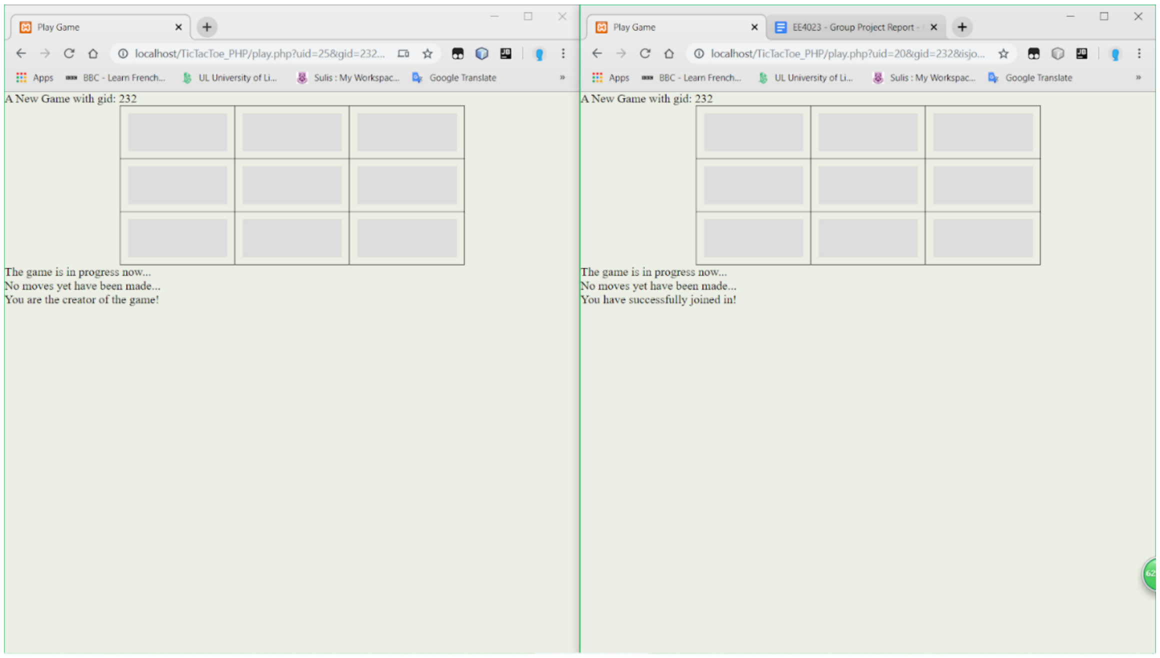
Game Page (Waiting for another player to join in)



A New Game with gid: 232

Waiting for a player to join in
You are the creator of the game!

Game Page (Playing)


Play Game
localhost/TicTacToe_PHP/play.php?uid=25&gid=232...
Play Game
EE4023 - Group Project Report
localhost/TicTacToe_PHP/play.php?uid=20&gid=232&isjo...

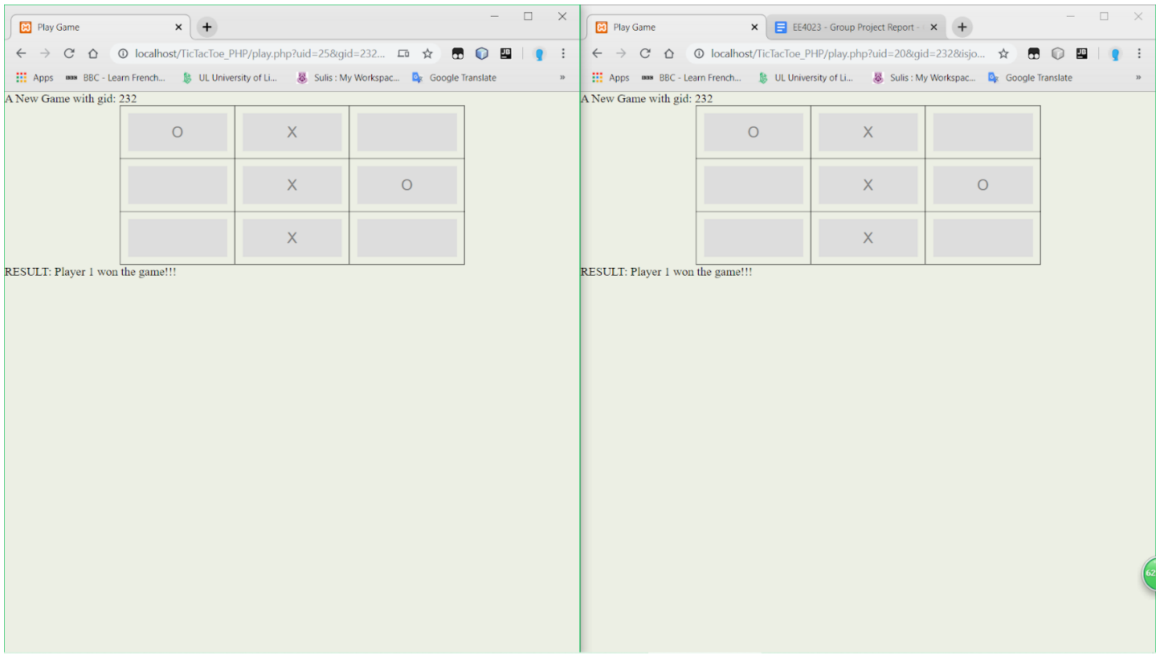
A New Game with gid: 232

The game is in progress now...
No moves yet have been made...
You are the creator of the game!

A New Game with gid: 232

The game is in progress now...
No moves yet have been made...
You have successfully joined in!

Game Page (Finish)


Play Game
localhost/TicTacToe_PHP/play.php?uid=25&gid=232...
Play Game
EE4023 - Group Project Report
localhost/TicTacToe_PHP/play.php?uid=20&gid=232&isjo...

A New Game with gid: 232

O	X	
	X	O
	X	

RESULT: Player 1 won the game!!!

A New Game with gid: 232

O	X	
	X	O
	X	

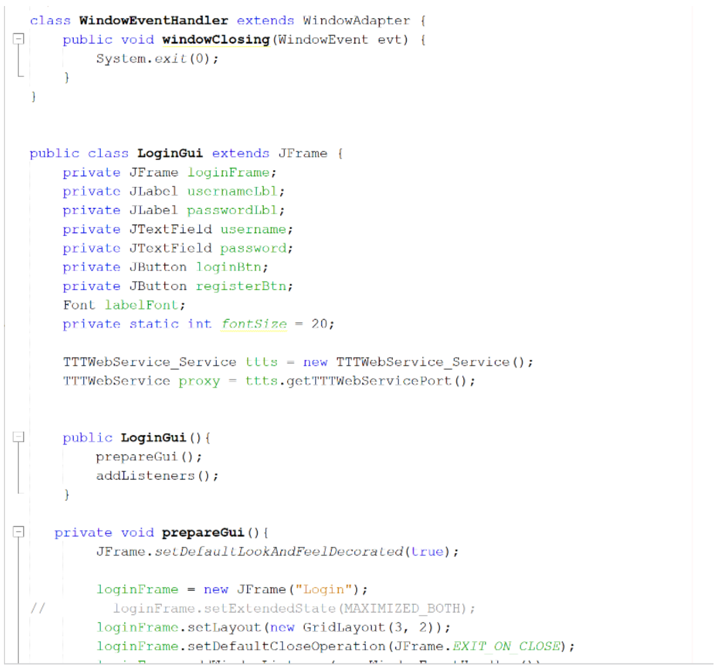
RESULT: Player 1 won the game!!!

Implementation

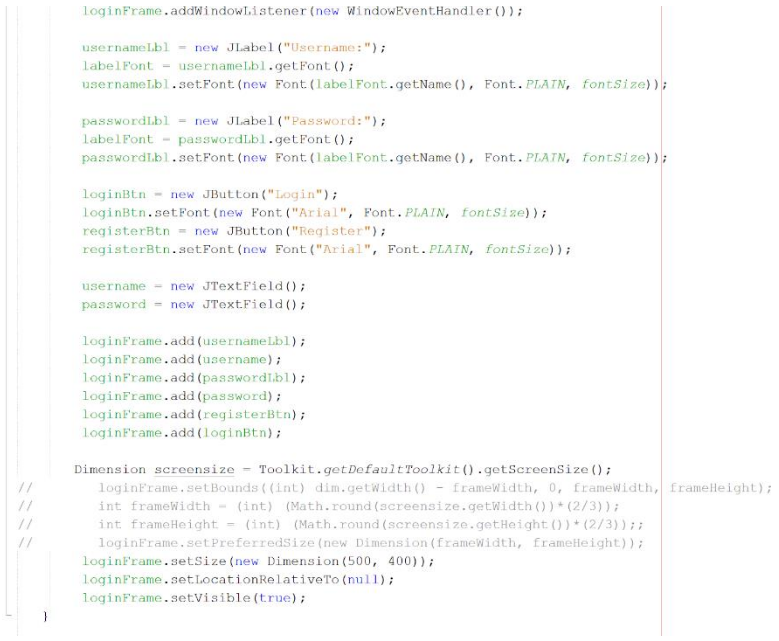
Java Part

The Java implementation uses the Swing library as a GUI and accesses the provided Web Service methods to interact with the Database. The Swing GUI can be run in any IDE that supports Java development.

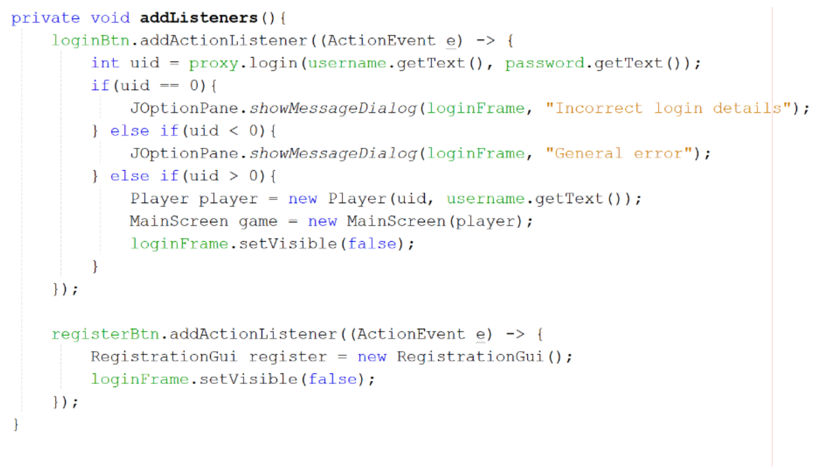
Login Implementation



```java
class WindowEventHandler extends WindowAdapter {
    public void windowClosing(WindowEvent evt) {
        System.exit(0);
    }
}


public class LoginGui extends JFrame {
    private JFrame loginFrame;
    private JLabel usernameLbl;
    private JLabel passwordLbl;
    private JTextField username;
    private JTextField password;
    private JButton loginBtn;
    private JButton registerBtn;
    Font labelFont;
    private static int fontSize = 20;

    TTTWebService_Service ttts = new TTTWebService_Service();
    TTTWebService proxy = ttts.getTTTWebServicePort();


    public LoginGui(){
        prepareGui();
        addListeners();
    }

   private void prepareGui(){
        JFrame.setDefaultLookAndFeelDecorated(true);

        loginFrame = new JFrame("Login");
//        loginFrame.setExtendedState(MAXIMIZED_BOTH);
        loginFrame.setLayout(new GridLayout(3, 2));
        loginFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```



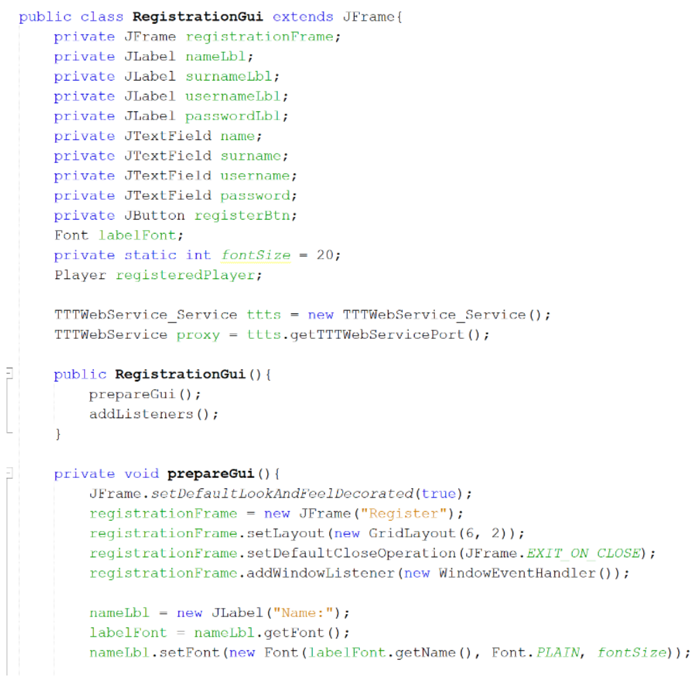
```java
        loginFrame.addWindowListener(new WindowEventHandler());

        usernameLbl = new JLabel("Username:");
        labelFont = usernameLbl.getFont();
        usernameLbl.setFont(new Font(labelFont.getName(), Font.PLAIN, fontSize));

        passwordLbl = new JLabel("Password:");
        labelFont = passwordLbl.getFont();
        passwordLbl.setFont(new Font(labelFont.getName(), Font.PLAIN, fontSize));

        loginBtn = new JButton("Login");
        loginBtn.setFont(new Font("Arial", Font.PLAIN, fontSize));
        registerBtn = new JButton("Register");
        registerBtn.setFont(new Font("Arial", Font.PLAIN, fontSize));

        username = new JTextField();
        password = new JTextField();

        loginFrame.add(usernameLbl);
        loginFrame.add(username);
        loginFrame.add(passwordLbl);
        loginFrame.add(password);
        loginFrame.add(registerBtn);
        loginFrame.add(loginBtn);

       Dimension screensize = Toolkit.getDefaultToolkit().getScreenSize();
//        loginFrame.setBounds((int) dim.getWidth() - frameWidth, 0, frameWidth, frameHeight);
//        int frameWidth = (int) (Math.round(screensize.getWidth())*(2/3));
//        int frameHeight = (int) (Math.round(screensize.getHeight())*(2/3));;
//        loginFrame.setPreferredSize(new Dimension(frameWidth, frameHeight));
        loginFrame.setSize(new Dimension(500, 400));
        loginFrame.setLocationRelativeTo(null);
        loginFrame.setVisible(true);
    }
```



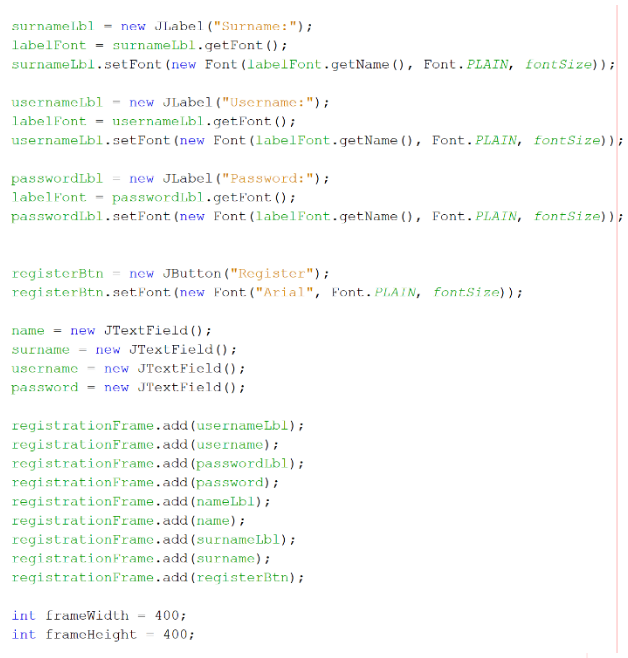
```
private void addListeners(){
    loginBtn.addActionListener((ActionEvent e) -> {
        int uid = proxy.login(username.getText(), password.getText());
        if(uid == 0){
            JOptionPane.showMessageDialog(loginFrame, "Incorrect login details");
        } else if(uid < 0){
            JOptionPane.showMessageDialog(loginFrame, "General error");
        } else if(uid > 0){
            Player player = new Player(uid, username.getText());
            MainScreen game = new MainScreen(player);
            loginFrame.setVisible(false);
        }
    });

    registerBtn.addActionListener((ActionEvent e) -> {
        RegistrationGui register = new RegistrationGui();
        loginFrame.setVisible(false);
    });
}
```

The login screen has two text fields where users can enter their login details or if they would like to register as a new user they can click the registration button that takes them to the registration page. If a user enters the wrong details or another error occurs when clicking the login button then a dialog box appears and shows an appropriate message. If login is successful then they are taken to the main screen. Listeners were added to the buttons to detect and event and decide on an action.

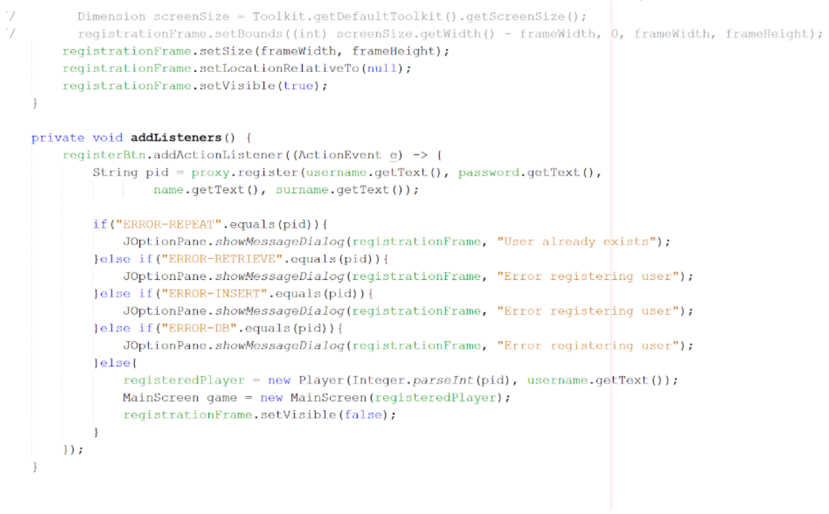
Registration Implementation



```
public class RegistrationGui extends JFrame{
    private JFrame registrationFrame;
    private JLabel nameLbl;
    private JLabel surnameLbl;
    private JLabel usernameLbl;
    private JLabel passwordLbl;
    private JTextField name;
    private JTextField surname;
    private JTextField username;
    private JTextField password;
    private JButton registerBtn;
    Font labelFont;
    private static int fontSize = 20;
    Player registeredPlayer;

    TTTWebService_Service ttts = new TTTWebService_Service();
    TTTWebService proxy = ttts.getTTTWebServicePort();

    public RegistrationGui(){
        prepareGui();
        addListeners();
    }

    private void prepareGui(){
        JFrame.setDefaultLookAndFeelDecorated(true);
        registrationFrame = new JFrame("Register");
        registrationFrame.setLayout(new GridLayout(6, 2));
        registrationFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        registrationFrame.addWindowListener(new WindowEventHandler());

        nameLbl = new JLabel("Name:");
        labelFont = nameLbl.getFont();
        nameLbl.setFont(new Font(labelFont.getName(), Font.PLAIN, fontSize));
```



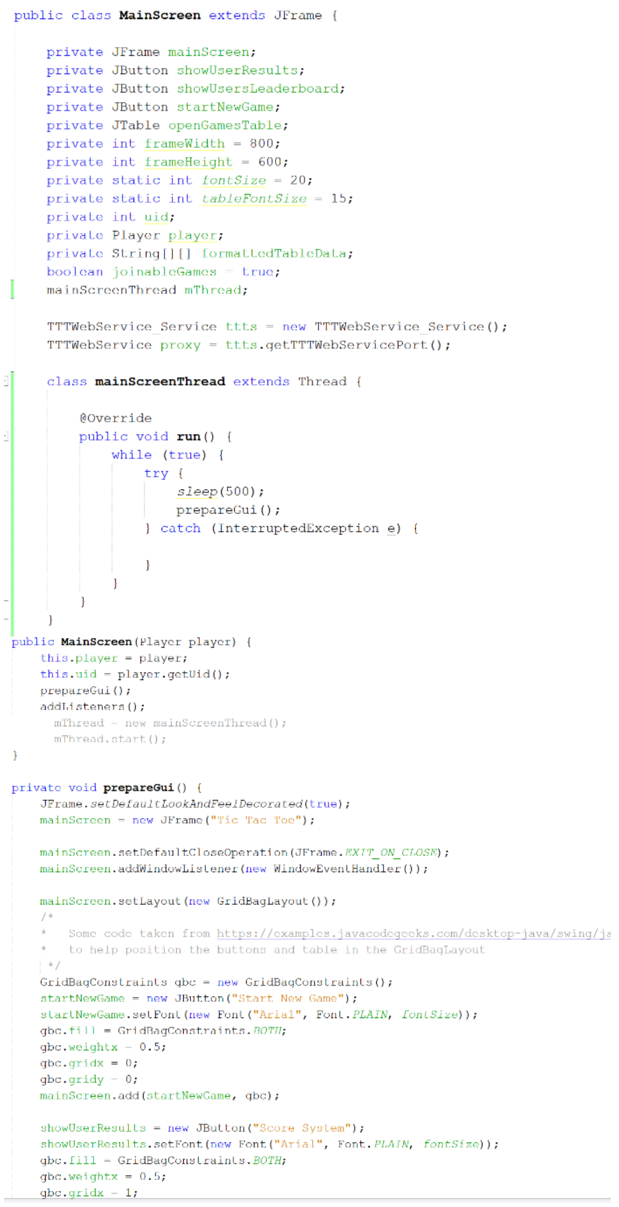
```
surnameLbl = new JLabel("Surname:");
labelFont = surnameLbl.getFont();
surnameLbl.setFont(new Font(labelFont.getName(), Font.PLAIN, fontSize));

usernameLbl = new JLabel("Username:");
labelFont = usernameLbl.getFont();
usernameLbl.setFont(new Font(labelFont.getName(), Font.PLAIN, fontSize));

passwordLbl = new JLabel("Password:");
labelFont = passwordLbl.getFont();
passwordLbl.setFont(new Font(labelFont.getName(), Font.PLAIN, fontSize));


registerBtn = new JButton("Register");
registerBtn.setFont(new Font("Arial", Font.PLAIN, fontSize));

name = new JTextField();
surname = new JTextField();
username = new JTextField();
password = new JTextField();

registrationFrame.add(usernameLbl);
registrationFrame.add(username);
registrationFrame.add(passwordLbl);
registrationFrame.add(password);
registrationFrame.add(nameLbl);
registrationFrame.add(name);
registrationFrame.add(surnameLbl);
registrationFrame.add(surname);
registrationFrame.add(registerBtn);

int frameWidth = 400;
int frameHeight = 400;
```



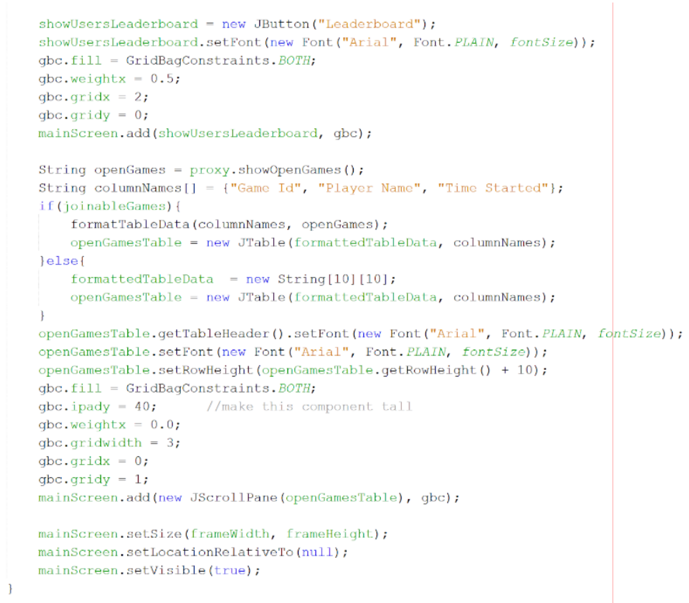
```
/       Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
/       registrationFrame.setBounds((int) screenSize.getWidth() - frameWidth, 0, frameWidth, frameHeight);
        registrationFrame.setSize(frameWidth, frameHeight);
        registrationFrame.setLocationRelativeTo(null);
        registrationFrame.setVisible(true);
    }

    private void addListeners() {
        registerBtn.addActionListener((ActionEvent e) -> {
            String pid = proxy.register(username.getText(), password.getText(),
                    name.getText(), surname.getText());

            if("ERROR-REPEAT".equals(pid)){
                JOptionPane.showMessageDialog(registrationFrame, "User already exists");
            }else if("ERROR-RETRIEVE".equals(pid)){
                JOptionPane.showMessageDialog(registrationFrame, "Error registering user");
            }else if("ERROR-INSERT".equals(pid)){
                JOptionPane.showMessageDialog(registrationFrame, "Error registering user");
            }else if("ERROR-DB".equals(pid)){
                JOptionPane.showMessageDialog(registrationFrame, "Error registering user");
            }else{
                registeredPlayer = new Player(Integer.parseInt(pid), username.getText());
                MainScreen game = new MainScreen(registeredPlayer);
                registrationFrame.setVisible(false);
            }
        });
    }
```

The registration screen has text fields to enter a new users username, password, first name and last name into the database. When a user has entered all the required fields they can click the register button which triggers an event and checks using the value returned from the register() web service method to either display an error or register a new user in the database and bring the user to the main screen.

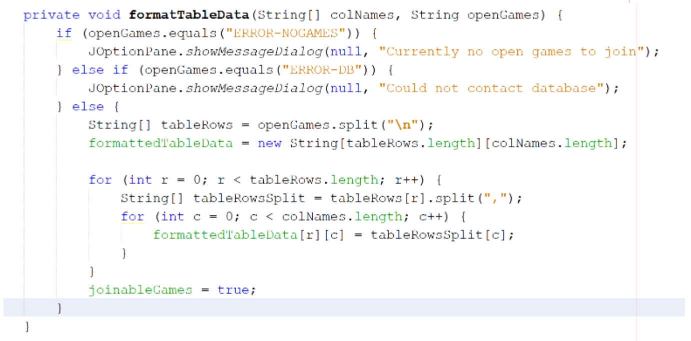
Main Screen Implementation



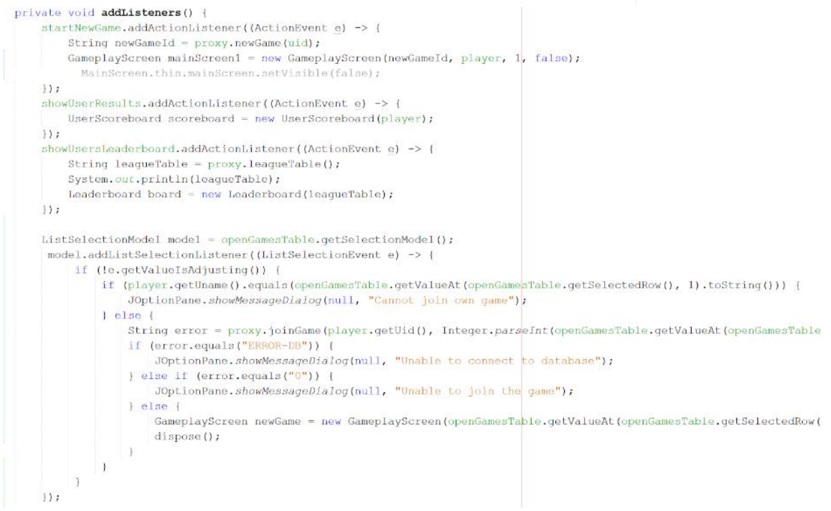
```java
public class MainScreen extends JFrame {

    private JFrame mainScreen;
    private JButton showUserResults;
    private JButton showUsersLeaderboard;
    private JButton startNewGame;
    private JTable openGamesTable;
    private int frameWidth = 800;
    private int frameHeight = 600;
    private static int fontSize = 20;
    private static int tableFontSize = 15;
    private int uid;
    private Player player;
    private String[][] formattedTableData;
    boolean joinableGames = true;
    mainScreenThread mThread;

    TTTWebService_Service ttts = new TTTWebService_Service();
    TTTWebService proxy = ttts.getTTTWebServicePort();

    class mainScreenThread extends Thread {

        @Override
        public void run() {
            while (true) {
                try {
                    sleep(500);
                    prepareGui();
                } catch (InterruptedException e) {

                }
            }
        }
    }
public MainScreen(Player player) {
    this.player = player;
    this.uid = player.getUid();
    prepareGui();
    addListeners();
      mThread = new mainScreenThread();
      mThread.start();
}

private void prepareGui() {
    JFrame.setDefaultLookAndFeelDecorated(true);
    mainScreen = new JFrame("Tic Tac Toe");

    mainScreen.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    mainScreen.addWindowListener(new WindowEventHandler());

    mainScreen.setLayout(new GridBagLayout());
    /*
    *   Some code taken from https://examples.javacodegeeks.com/desktop-java/swing/ja
    *   to help position the buttons and table in the GridBagLayout
     */
    GridBagConstraints gbc = new GridBagConstraints();
    startNewGame = new JButton("Start New Game");
    startNewGame.setFont(new Font("Arial", Font.PLAIN, fontSize));
    gbc.fill = GridBagConstraints.BOTH;
    gbc.weightx = 0.5;
    gbc.gridx = 0;
    gbc.gridy = 0;
    mainScreen.add(startNewGame, gbc);

    showUserResults = new JButton("Score System");
    showUserResults.setFont(new Font("Arial", Font.PLAIN, fontSize));
    gbc.fill = GridBagConstraints.BOTH;
    gbc.weightx = 0.5;
    gbc.gridx = 1;
```



```
        showUsersLeaderboard = new JButton("Leaderboard");
        showUsersLeaderboard.setFont(new Font("Arial", Font.PLAIN, fontSize));
        gbc.fill = GridBagConstraints.BOTH;
        gbc.weightx = 0.5;
        gbc.gridx = 2;
        gbc.gridy = 0;
        mainScreen.add(showUsersLeaderboard, gbc);

        String openGames = proxy.showOpenGames();
        String columnNames[] = {"Game Id", "Player Name", "Time Started"};
        if(joinableGames){
            formatTableData(columnNames, openGames);
            openGamesTable = new JTable(formattedTableData, columnNames);
        }else{
            formattedTableData  = new String[10][10];
            openGamesTable = new JTable(formattedTableData, columnNames);
        }
        openGamesTable.getTableHeader().setFont(new Font("Arial", Font.PLAIN, fontSize));
        openGamesTable.setFont(new Font("Arial", Font.PLAIN, fontSize));
        openGamesTable.setRowHeight(openGamesTable.getRowHeight() + 10);
        gbc.fill = GridBagConstraints.BOTH;
        gbc.ipady = 40;      //make this component tall
        gbc.weightx = 0.0;
        gbc.gridwidth = 3;
        gbc.gridx = 0;
        gbc.gridy = 1;
        mainScreen.add(new JScrollPane(openGamesTable), gbc);

        mainScreen.setSize(frameWidth, frameHeight);
        mainScreen.setLocationRelativeTo(null);
        mainScreen.setVisible(true);
    }
```



```
    private void formatTableData(String[] colNames, String openGames) {
        if (openGames.equals("ERROR-NOGAMES")) {
            JOptionPane.showMessageDialog(null, "Currently no open games to join");
        } else if (openGames.equals("ERROR-DB")) {
            JOptionPane.showMessageDialog(null, "Could not contact database");
        } else {
            String[] tableRows = openGames.split("\n");
            formattedTableData = new String[tableRows.length][colNames.length];

            for (int r = 0; r < tableRows.length; r++) {
                String[] tableRowsSplit = tableRows[r].split(",");
                for (int c = 0; c < colNames.length; c++) {
                    formattedTableData[r][c] = tableRowsSplit[c];
                }
            }
            joinableGames = true;
        }
    }
```

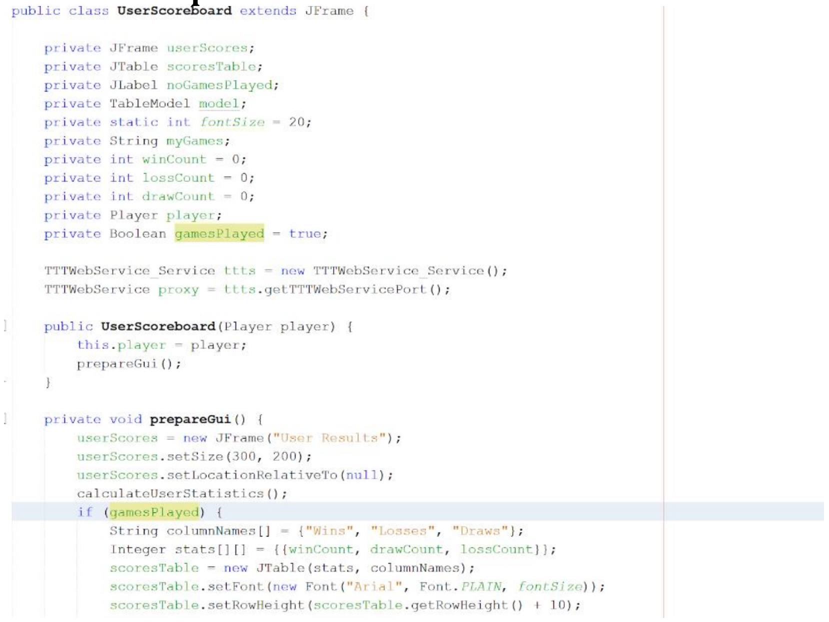


```
    private void addListeners() {
        startNewGame.addActionListener((ActionEvent e) -> {
            String newGameId = proxy.newGame(uid);
            GameplayScreen mainScreen1 = new GameplayScreen(newGameId, player, 1, false);
             MainScreen.this.mainScreen.setVisible(false);
        });
        showUserResults.addActionListener((ActionEvent e) -> {
            UserScoreboard scoreboard = new UserScoreboard(player);
        });
        showUsersLeaderboard.addActionListener((ActionEvent e) -> {
            String leagueTable = proxy.leagueTable();
            System.out.println(leagueTable);
            Leaderboard board = new Leaderboard(leagueTable);
        });

        ListSelectionModel model = openGamesTable.getSelectionModel();
         model.addListSelectionListener((ListSelectionEvent e) -> {
            if (!e.getValueIsAdjusting()) {
                if (player.getUname().equals(openGamesTable.getValueAt(openGamesTable.getSelectedRow(), 1).toString())) {
                    JOptionPane.showMessageDialog(null, "Cannot join own game");
                } else {
                    String error = proxy.joinGame(player.getUid(), Integer.parseInt(openGamesTable.getValueAt(openGamesTable
                    if (error.equals("ERROR-DB")) {
                        JOptionPane.showMessageDialog(null, "Unable to connect to database");
                    } else if (error.equals("0")) {
                        JOptionPane.showMessageDialog(null, "Unable to join the game");
                    } else {
                        GameplayScreen newGame = new GameplayScreen(openGamesTable.getValueAt(openGamesTable.getSelectedRow(
                        dispose();
                    }
                }
            }
        });
```

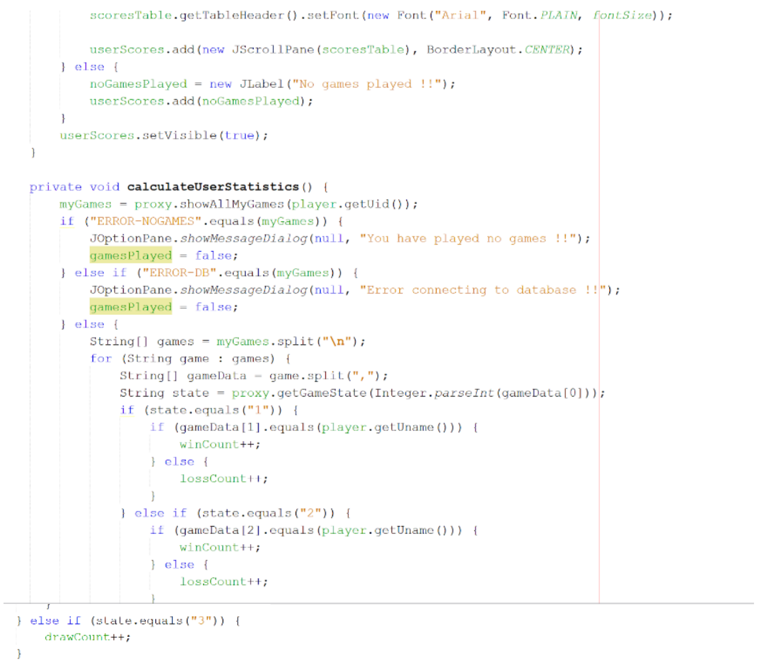
A large chunk of code in the main screen is dedicated to setting up the UI and setting font sizes on the buttons and on the table for open games.
After adding the buttons to the frame the table displaying a list of open games needed to be added and populated with data from the database.
To do this the method formatTableData() was used to populate the table. After calling the web service method to get all the open games, to get individual games

the result was split on a new line. This array then gave access to each open game that would make up a row in the table. Then using two for loops the individual games array was split by commas and each entry was added to the correct row and column that Jtable requires.
In order for a user to join an open game and play against another user a ListSelectionListener was added to the table. If the user was the creator of the game they would not be allowed to join but if they weren't the creator then they join the new game that is found by accessing the game id in the table and join a game.

UserStats Implementation



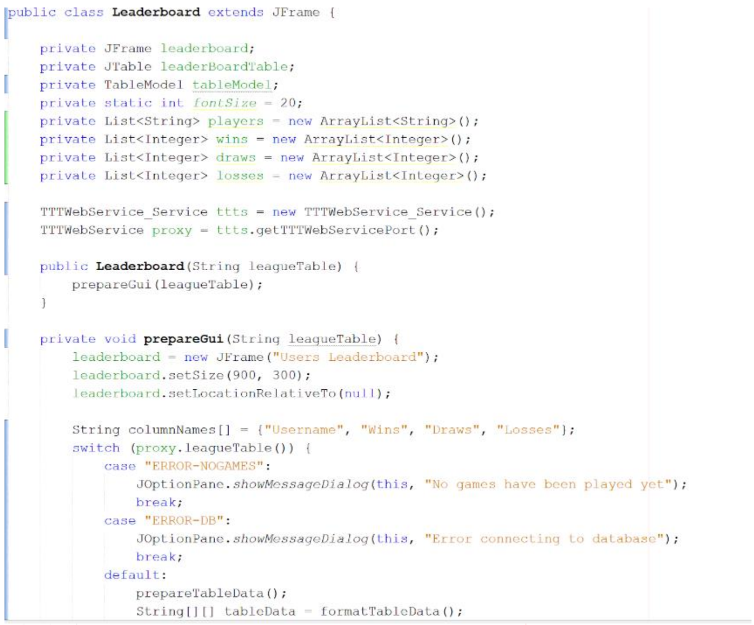
```java
public class UserScoreboard extends JFrame {

    private JFrame userScores;
    private JTable scoresTable;
    private JLabel noGamesPlayed;
    private TableModel model;
    private static int fontSize = 20;
    private String myGames;
    private int winCount = 0;
    private int lossCount = 0;
    private int drawCount = 0;
    private Player player;
    private Boolean gamesPlayed = true;

    TTTWebService_Service ttts = new TTTWebService_Service();
    TTTWebService proxy = ttts.getTTTWebServicePort();

    public UserScoreboard(Player player) {
        this.player = player;
        prepareGui();
    }

    private void prepareGui() {
        userScores = new JFrame("User Results");
        userScores.setSize(300, 200);
        userScores.setLocationRelativeTo(null);
        calculateUserStatistics();
        if (gamesPlayed) {
            String columnNames[] = {"Wins", "Losses", "Draws"};
            Integer stats[][] = {{winCount, drawCount, lossCount}};
            scoresTable = new JTable(stats, columnNames);
            scoresTable.setFont(new Font("Arial", Font.PLAIN, fontSize));
            scoresTable.setRowHeight(scoresTable.getRowHeight() + 10);
```



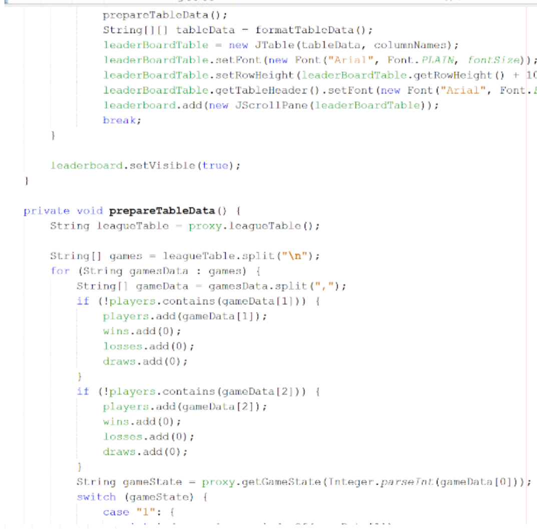
```
            scoresTable.getTableHeader().setFont(new Font("Arial", Font.PLAIN, fontSize));

            userScores.add(new JScrollPane(scoresTable), BorderLayout.CENTER);
        } else {
            noGamesPlayed = new JLabel("No games played !!");
            userScores.add(noGamesPlayed);
        }
        userScores.setVisible(true);
    }

    private void calculateUserStatistics() {
        myGames = proxy.showAllMyGames(player.getUid());
        if ("ERROR-NOGAMES".equals(myGames)) {
            JOptionPane.showMessageDialog(null, "You have played no games !!");
            gamesPlayed = false;
        } else if ("ERROR-DB".equals(myGames)) {
            JOptionPane.showMessageDialog(null, "Error connecting to database !!");
            gamesPlayed = false;
        } else {
            String[] games = myGames.split("\n");
            for (String game : games) {
                String[] gameData = game.split(",");
                String state = proxy.getGameState(Integer.parseInt(gameData[0]));
                if (state.equals("1")) {
                    if (gameData[1].equals(player.getUname())) {
                        winCount++;
                    } else {
                        lossCount++;
                    }
                } else if (state.equals("2")) {
                    if (gameData[2].equals(player.getUname())) {
                        winCount++;
                    } else {
                        lossCount++;
                    }
} else if (state.equals("3")) {
   drawCount++;
}
```

For calculating a logged in users draws, wins and losses the calculate user statistics method was used. Calling the web service method showAllMyGames() gave access to all the games the user has taken part in. This result was then split on a new line to access an individual game which could be used to add to the win, draw or loss count. Using a for loop to iterate over the individual games, the getGameState() method was used and the game id of that particular game was passed in. Checking the result of the game state and the status of the user the win, loss and draw count was added to and displayed in a table.

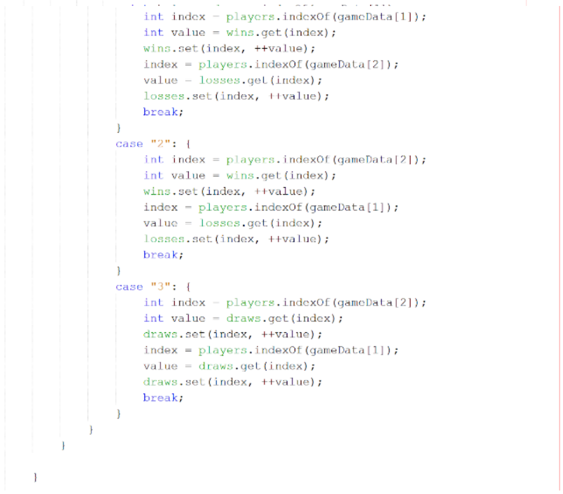
Leaderboard Implementation



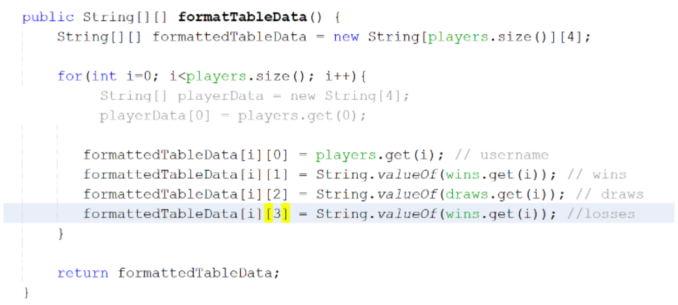
```java
public class Leaderboard extends JFrame {

    private JFrame leaderboard;
    private JTable leaderBoardTable;
    private TableModel tableModel;
    private static int fontSize = 20;
    private List<String> players = new ArrayList<String>();
    private List<Integer> wins = new ArrayList<Integer>();
    private List<Integer> draws = new ArrayList<Integer>();
    private List<Integer> losses = new ArrayList<Integer>();

    TTTWebService_Service ttts = new TTTWebService_Service();
    TTTWebService proxy = ttts.getTTTWebServicePort();

    public Leaderboard(String leagueTable) {
        prepareGui(leagueTable);
    }

    private void prepareGui(String leagueTable) {
        leaderboard = new JFrame("Users Leaderboard");
        leaderboard.setSize(900, 300);
        leaderboard.setLocationRelativeTo(null);

        String columnNames[] = {"Username", "Wins", "Draws", "Losses"};
        switch (proxy.leagueTable()) {
            case "ERROR-NOGAMES":
                JOptionPane.showMessageDialog(this, "No games have been played yet");
                break;
            case "ERROR-DB":
                JOptionPane.showMessageDialog(this, "Error connecting to database");
                break;
            default:
                prepareTableData();
                String[][] tableData = formatTableData();
```



```java
                prepareTableData();
                String[][] tableData = formatTableData();
                leaderBoardTable = new JTable(tableData, columnNames);
                leaderBoardTable.setFont(new Font("Arial", Font.PLAIN, fontSize));
                leaderBoardTable.setRowHeight(leaderBoardTable.getRowHeight() + 10);
                leaderBoardTable.getTableHeader().setFont(new Font("Arial", Font.PLAIN, fontSize));
                leaderboard.add(new JScrollPane(leaderBoardTable));
                break;
        }

        leaderboard.setVisible(true);
    }

    private void prepareTableData() {
        String leagueTable = proxy.leagueTable();

        String[] games = leagueTable.split("\n");
        for (String gamesData : games) {
            String[] gameData = gamesData.split(",");
            if (!players.contains(gameData[1])) {
                players.add(gameData[1]);
                wins.add(0);
                losses.add(0);
                draws.add(0);
            }
            if (!players.contains(gameData[2])) {
                players.add(gameData[2]);
                wins.add(0);
                losses.add(0);
                draws.add(0);
            }
            String gameState = proxy.getGameState(Integer.parseInt(gameData[0]));
            switch (gameState) {
                case "1": {
```



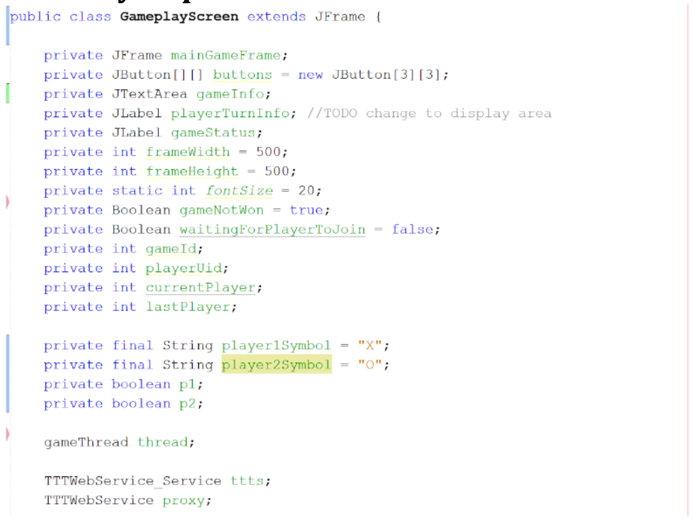
```java
                    int index = players.indexOf(gameData[1]);
                    int value = wins.get(index);
                    wins.set(index, ++value);
                    index = players.indexOf(gameData[2]);
                    value = losses.get(index);
                    losses.set(index, ++value);
                    break;
                }
                case "2": {
                    int index = players.indexOf(gameData[2]);
                    int value = wins.get(index);
                    wins.set(index, ++value);
                    index = players.indexOf(gameData[1]);
                    value = losses.get(index);
                    losses.set(index, ++value);
                    break;
                }
                case "3": {
                    int index = players.indexOf(gameData[2]);
                    int value = draws.get(index);
                    draws.set(index, ++value);
                    index = players.indexOf(gameData[1]);
                    value = draws.get(index);
                    draws.set(index, ++value);
                    break;
                }
            }
        }
    }
```



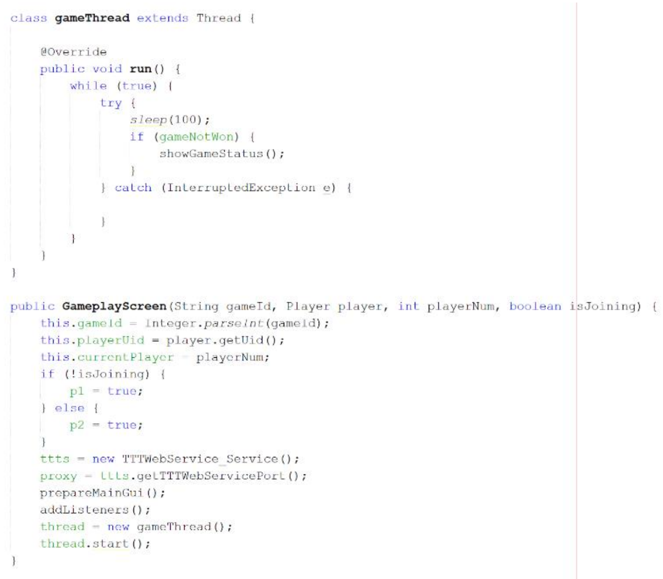
```
public String[][] formatTableData() {
    String[][] formattedTableData = new String[players.size()][4];

    for(int i=0; i<players.size(); i++){
        String[] playerData = new String[4];
        playerData[0] = players.get(0);

      formattedTableData[i][0] = players.get(i); // username
      formattedTableData[i][1] = String.valueOf(wins.get(i)); // wins
      formattedTableData[i][2] = String.valueOf(draws.get(i)); // draws
      formattedTableData[i][3] = String.valueOf(wins.get(i)); //losses
    }

    return formattedTableData;
}
```

The implementation of the users leaderboard is very similar to the user stats implementation. First three ArrayLists are defined to keep track of players, wins, losses and draws. Using the web service method leagueTable() to get all the games played we split them on a new line to access each game. Then iterating over all the games we check if player 1 and 2 in each game has already been added to the players list. if not then we add them to it. Then using a switch statement we check each game state and depending on the state we find the index of the player in the players list and add the corresponding win, draw or loss to their count.
After all the games have been iterated over we need to prepare the data to be put into a table. Using a for loop each player and their stats are added to individual arrays as part of a larger 2d array. This is then used to populate the table.
There is also a thread class that was supposed to be used to update the table but finding the correct piece of code to update without updating the entire GUI proved too difficult.

GamePlay Implementation

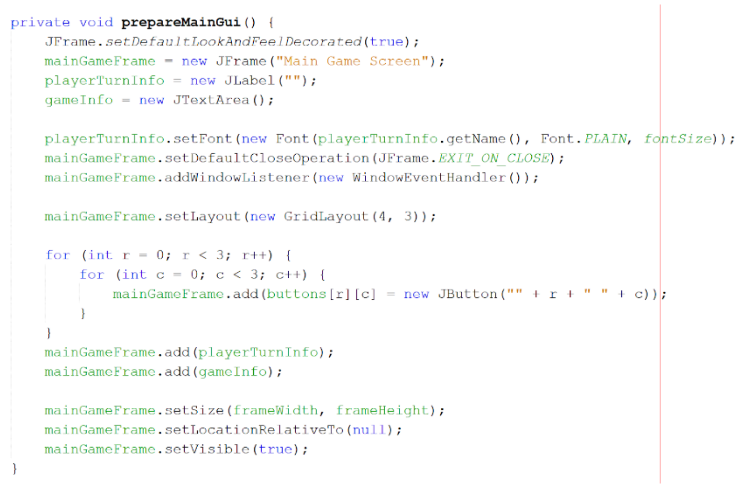


```
public class GameplayScreen extends JFrame {

    private JFrame mainGameFrame;
    private JButton[][] buttons = new JButton[3][3];
    private JTextArea gameInfo;
    private JLabel playerTurnInfo; //TODO change to display area
    private JLabel gameStatus;
    private int frameWidth = 500;
    private int frameHeight = 500;
    private static int fontSize = 20;
    private Boolean gameNotWon = true;
    private Boolean waitingForPlayerToJoin = false;
    private int gameId;
    private int playerUid;
    private int currentPlayer;
    private int lastPlayer;

    private final String player1Symbol = "X";
    private final String player2Symbol = "O";
    private boolean p1;
    private boolean p2;

    gameThread thread;

    TTTWebService_Service ttts;
    TTTWebService proxy;
```

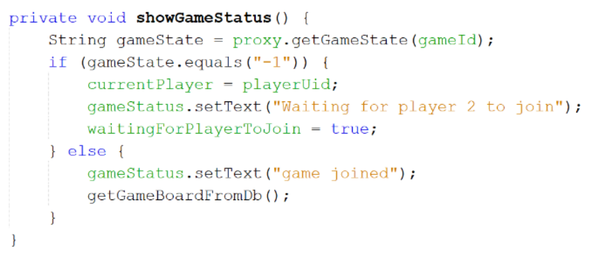


```
class gameThread extends Thread {

    @Override
    public void run() {
        while (true) {
            try {
                sleep(100);
                if (gameNotWon) {
                    showGameStatus();
                }
            } catch (InterruptedException e) {

            }
        }
    }
}

public GameplayScreen(String gameId, Player player, int playerNum, boolean isJoining) {
    this.gameId = Integer.parseInt(gameId);
    this.playerUid = player.getUid();
    this.currentPlayer = playerNum;
    if (!isJoining) {
        p1 = true;
    } else {
        p2 = true;
    }
    ttts = new TTTWebService_Service();
    proxy = ttts.getTTTWebServicePort();
    prepareMainGui();
    addListeners();
    thread = new gameThread();
    thread.start();
}
```

The implementation of the game is done through a 3x3 board of buttons.
There is also a thread class which checks at an interval if the game state is won or not and if not it calls the showGameStatus() method which tells the user if they are waiting for someone to join the game or not.

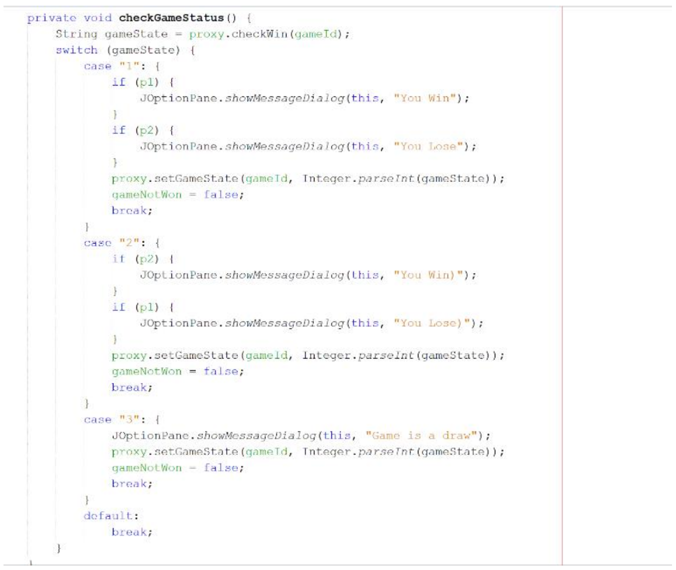


```
private void prepareMainGui() {
    JFrame.setDefaultLookAndFeelDecorated(true);
    mainGameFrame = new JFrame("Main Game Screen");
    playerTurnInfo = new JLabel("");
    gameInfo = new JTextArea();

    playerTurnInfo.setFont(new Font(playerTurnInfo.getName(), Font.PLAIN, fontSize));
    mainGameFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    mainGameFrame.addWindowListener(new WindowEventHandler());

    mainGameFrame.setLayout(new GridLayout(4, 3));

    for (int r = 0; r < 3; r++) {
        for (int c = 0; c < 3; c++) {
            mainGameFrame.add(buttons[r][c] = new JButton("" + r + " " + c));
        }
    }
    mainGameFrame.add(playerTurnInfo);
    mainGameFrame.add(gameInfo);

    mainGameFrame.setSize(frameWidth, frameHeight);
    mainGameFrame.setLocationRelativeTo(null);
    mainGameFrame.setVisible(true);
}
```



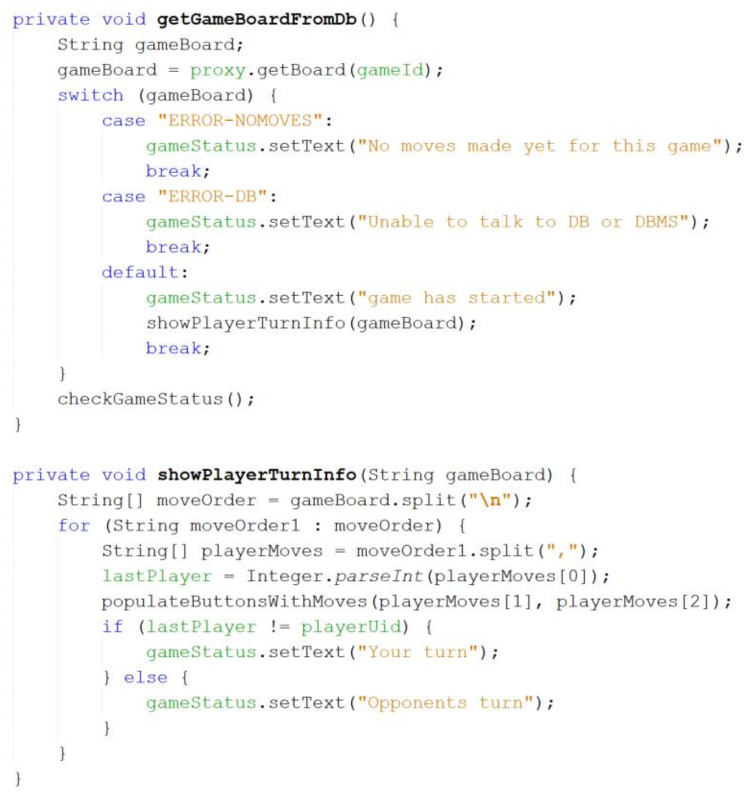
```
private void showGameStatus() {
    String gameState = proxy.getGameState(gameId);
    if (gameState.equals("-1")) {
        currentPlayer = playerUid;
        gameStatus.setText("Waiting for player 2 to join");
        waitingForPlayerToJoin = true;
    } else {
        gameStatus.setText("game joined");
        getGameBoardFromDb();
    }
}
```

Each time showGameStatus() is called it checks the game state to see if another user needs to join the game. If another user has joined then it gets the board configuration from the database checks the game board using getGameBoardFromDb().



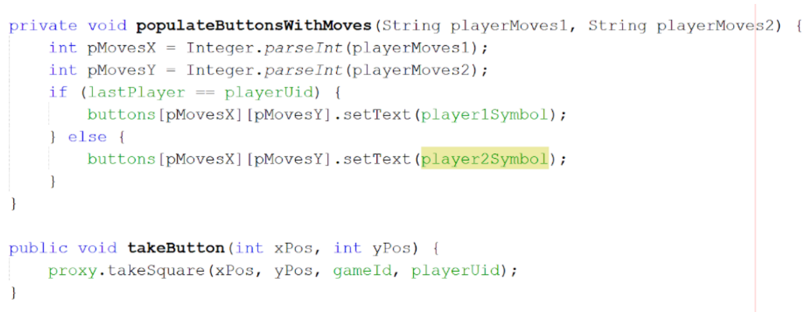
```
private void checkGameStatus() {
    String gameState = proxy.checkWin(gameId);
    switch (gameState) {
        case "1": {
            if (p1) {
                JOptionPane.showMessageDialog(this, "You Win");
            }
            if (p2) {
                JOptionPane.showMessageDialog(this, "You Lose");
            }
            proxy.setGameState(gameId, Integer.parseInt(gameState));
            gameNotWon = false;
            break;
        }
        case "2": {
            if (p2) {
                JOptionPane.showMessageDialog(this, "You Win)");
            }
            if (p1) {
                JOptionPane.showMessageDialog(this, "You Lose)");
            }
            proxy.setGameState(gameId, Integer.parseInt(gameState));
            gameNotWon = false;
            break;
        }
        case "3": {
            JOptionPane.showMessageDialog(this, "Game is a draw");
            proxy.setGameState(gameId, Integer.parseInt(gameState));
            gameNotWon = false;
            break;
        }
        default:
            break;
    }
```

The method checkGameStatus() uses a switch statement to iterate through the various states a game can take and using the result from the web service method checkWin() it displays the result to the user.



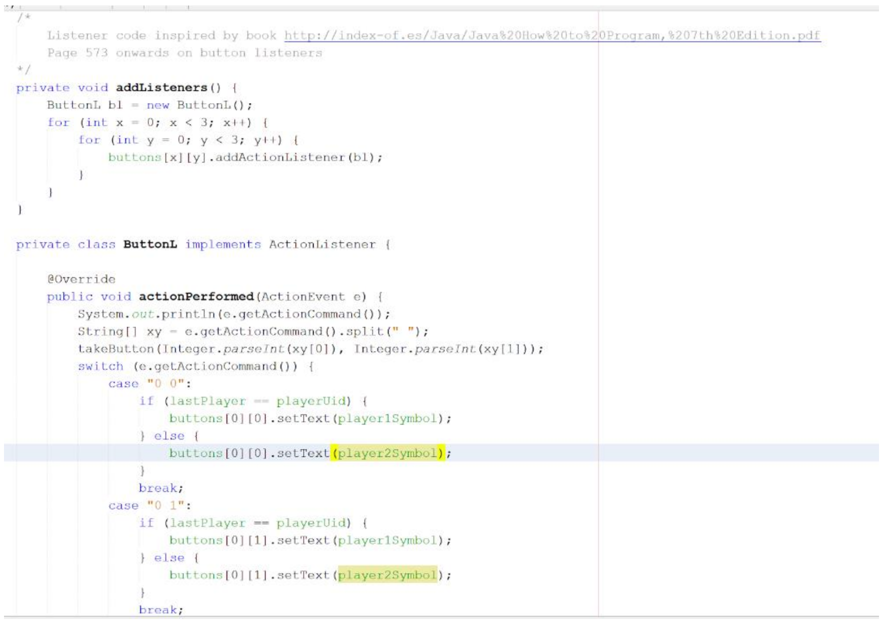
```
private void getGameBoardFromDb() {
    String gameBoard;
    gameBoard = proxy.getBoard(gameId);
    switch (gameBoard) {
        case "ERROR-NOMOVES":
            gameStatus.setText("No moves made yet for this game");
            break;
        case "ERROR-DB":
            gameStatus.setText("Unable to talk to DB or DBMS");
            break;
        default:
            gameStatus.setText("game has started");
            showPlayerTurnInfo(gameBoard);
            break;
    }
    checkGameStatus();
}

private void showPlayerTurnInfo(String gameBoard) {
    String[] moveOrder = gameBoard.split("\n");
    for (String moveOrder1 : moveOrder) {
        String[] playerMoves = moveOrder1.split(",");
        lastPlayer = Integer.parseInt(playerMoves[0]);
        populateButtonsWithMoves(playerMoves[1], playerMoves[2]);
        if (lastPlayer != playerUid) {
            gameStatus.setText("Your turn");
        } else {
            gameStatus.setText("Opponents turn");
        }
    }
}
```

The method getGameBoardFromDb() checks whether any moves have been made in the current game and if not calls showPlayerTurnInfo() to inform the user if it is their turn or not.

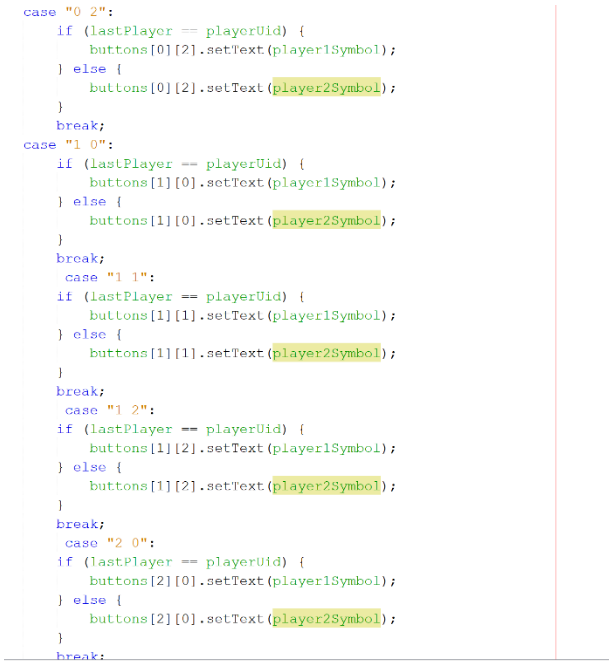


```
private void populateButtonsWithMoves(String playerMoves1, String playerMoves2) {
    int pMovesX = Integer.parseInt(playerMoves1);
    int pMovesY = Integer.parseInt(playerMoves2);
    if (lastPlayer == playerUid) {
        buttons[pMovesX][pMovesY].setText(player1Symbol);
    } else {
        buttons[pMovesX][pMovesY].setText(player2Symbol);
    }
}

public void takeButton(int xPos, int yPos) {
    proxy.takeSquare(xPos, yPos, gameId, playerUid);
}
```

The above method populates the buttons board when a game has saved moves in the database.
The takeButton() method updates the database using the takeSquare() web service method.



```
/*
    Listener code inspired by book http://index-of.es/Java/Java%20How%20to%20Program,%207th%20Edition.pdf
    Page 573 onwards on button listeners
*/
private void addListeners() {
    ButtonL bl = new ButtonL();
    for (int x = 0; x < 3; x++) {
        for (int y = 0; y < 3; y++) {
            buttons[x][y].addActionListener(bl);
        }
    }
}

private class ButtonL implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.println(e.getActionCommand());
        String[] xy = e.getActionCommand().split(" ");
        takeButton(Integer.parseInt(xy[0]), Integer.parseInt(xy[1]));
        switch (e.getActionCommand()) {
            case "0 0":
                if (lastPlayer == playerUid) {
                    buttons[0][0].setText(player1Symbol);
                } else {
                    buttons[0][0].setText(player2Symbol);
                }
                break;
            case "0 1":
                if (lastPlayer == playerUid) {
                    buttons[0][1].setText(player1Symbol);
                } else {
                    buttons[0][1].setText(player2Symbol);
                }
                break;
```



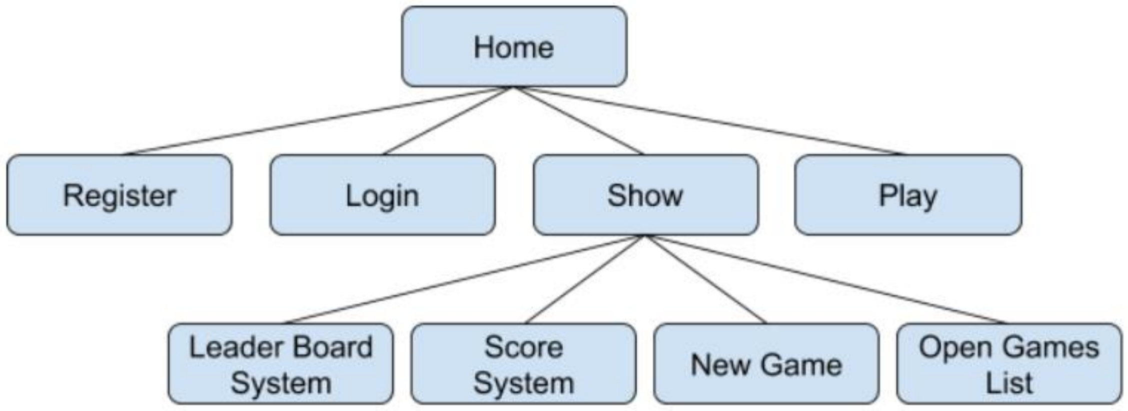
```
case "0 2":
    if (lastPlayer == playerUid) {
        buttons[0][2].setText(player1Symbol);
    } else {
        buttons[0][2].setText(player2Symbol);
    }
    break;
case "1 0":
    if (lastPlayer == playerUid) {
        buttons[1][0].setText(player1Symbol);
    } else {
        buttons[1][0].setText(player2Symbol);
    }
    break;
     case "1 1":
    if (lastPlayer == playerUid) {
        buttons[1][1].setText(player1Symbol);
    } else {
        buttons[1][1].setText(player2Symbol);
    }
    break;
     case "1 2":
    if (lastPlayer == playerUid) {
        buttons[1][2].setText(player1Symbol);
    } else {
        buttons[1][2].setText(player2Symbol);
    }
    break;
     case "2 0":
    if (lastPlayer == playerUid) {
        buttons[2][0].setText(player1Symbol);
    } else {
        buttons[2][0].setText(player2Symbol);
    }
    break;
```

The listener code above had to be implemented for all buttons using a switch statement and depending on the x and y co ordinate of the button being pressed and the player that pressed it the button text would be set to X or O to signify a move made by a user on the board.

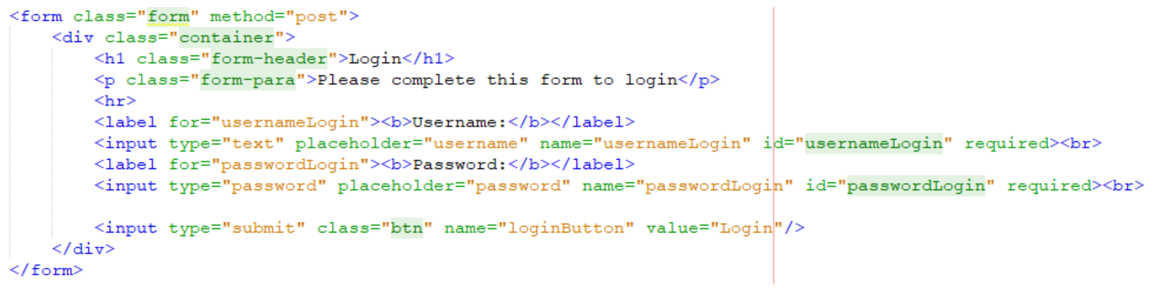
PHP Part

As for the PHP driven website, the implementation phase involves installing the provided Web Service into the development environment. The website can be accessed on a computer which has PHP and MySQL installed in it.

The main layout of this system is shown below.

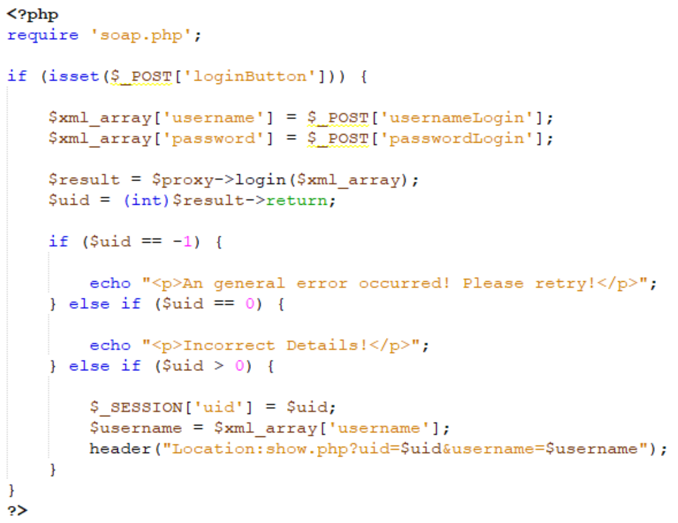


Impl for Login



```
<form class="form" method="post">
    <div class="container">
        <h1 class="form-header">Login</h1>
        <p class="form-para">Please complete this form to login</p>
        <hr>
        <label for="usernameLogin"><b>Username:</b></label>
        <input type="text" placeholder="username" name="usernameLogin" id="usernameLogin" required><br>
        <label for="passwordLogin"><b>Password:</b></label>
        <input type="password" placeholder="password" name="passwordLogin" id="passwordLogin" required><br>

        <input type="submit" class="btn" name="loginButton" value="Login"/>
    </div>
</form>
```

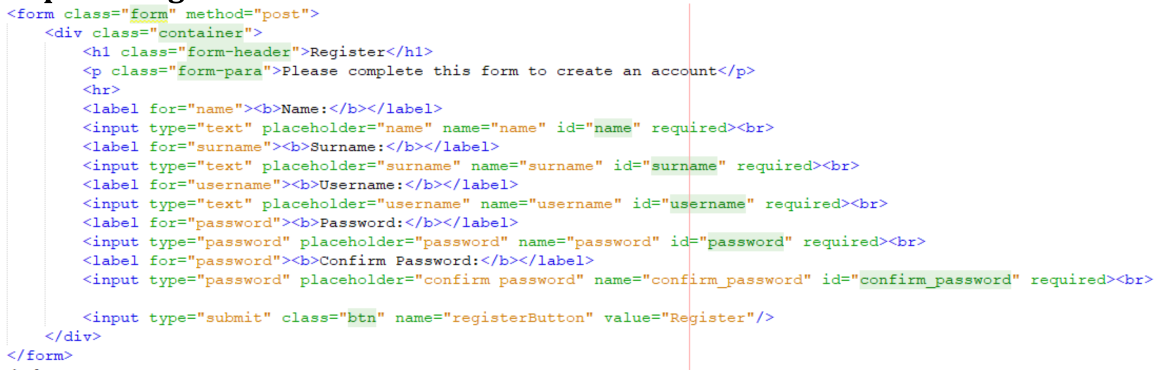
The Login page generally has two text input boxes with a button. These components are wrapped in a form tag. The contents of the user input can be passed through POST.



```
<?php
require 'soap.php';

if (isset($_POST['loginButton'])) {

    $xml_array['username'] = $_POST['usernameLogin'];
    $xml_array['password'] = $_POST['passwordLogin'];

    $result = $proxy->login($xml_array);
    $uid = (int)$result->return;

    if ($uid == -1) {

        echo "<p>An general error occurred! Please retry!</p>";
    } else if ($uid == 0) {

        echo "<p>Incorrect Details!</p>";
    } else if ($uid > 0) {

        $_SESSION['uid'] = $uid;
        $username = $xml_array['username'];
        header("Location:show.php?uid=$uid&username=$username");
    }
}
?>
```

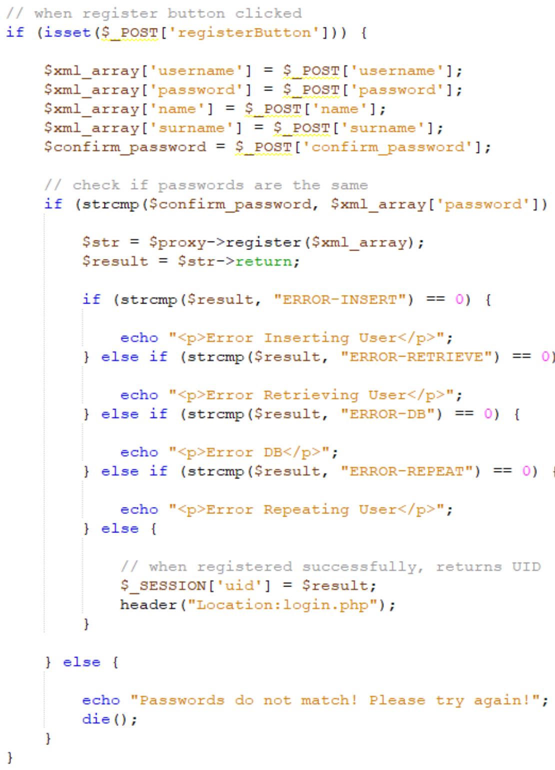
The PHP code for login is shown above, when the button clicked, the input values will be accessed and then using login webservice provided to save data into database. When the returned result > 0, it will direct to the Show page.

Impl for Registration



```
<form class="form" method="post">
    <div class="container">
        <h1 class="form-header">Register</h1>
        <p class="form-para">Please complete this form to create an account</p>
        <hr>
        <label for="name"><b>Name:</b></label>
        <input type="text" placeholder="name" name="name" id="name" required><br>
        <label for="surname"><b>Surname:</b></label>
        <input type="text" placeholder="surname" name="surname" id="surname" required><br>
        <label for="username"><b>Username:</b></label>
        <input type="text" placeholder="username" name="username" id="username" required><br>
        <label for="password"><b>Password:</b></label>
        <input type="password" placeholder="password" name="password" id="password" required><br>
        <label for="password"><b>Confirm Password:</b></label>
        <input type="password" placeholder="confirm password" name="confirm_password" id="confirm_password" required><br>

        <input type="submit" class="btn" name="registerButton" value="Register"/>
    </div>
</form>
```

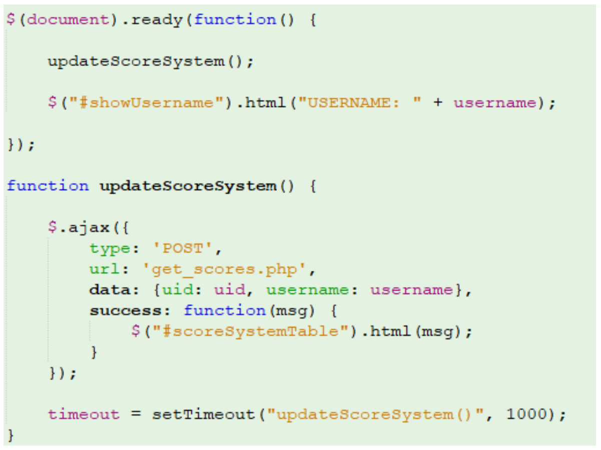
Similar to Login, Registration has more input text boxes, with a password confirmation technique.



```
// when register button clicked
if (isset($_POST['registerButton'])) {

    $xml_array['username'] = $_POST['username'];
    $xml_array['password'] = $_POST['password'];
    $xml_array['name'] = $_POST['name'];
    $xml_array['surname'] = $_POST['surname'];
    $confirm_password = $_POST['confirm_password'];

    // check if passwords are the same
    if (strcmp($confirm_password, $xml_array['password']) == 0) {

        $str = $proxy->register($xml_array);
        $result = $str->return;

        if (strcmp($result, "ERROR-INSERT") == 0) {

            echo "<p>Error Inserting User</p>";
        } else if (strcmp($result, "ERROR-RETRIEVE") == 0) {

            echo "<p>Error Retrieving User</p>";
        } else if (strcmp($result, "ERROR-DB") == 0) {

            echo "<p>Error DB</p>";
        } else if (strcmp($result, "ERROR-REPEAT") == 0) {

            echo "<p>Error Repeating User</p>";
        } else {

            // when registered successfully, returns UID
            $_SESSION['uid'] = $result;
            header("Location:login.php");
        }

    } else {

        echo "Passwords do not match! Please try again!";
        die();
    }
}
```

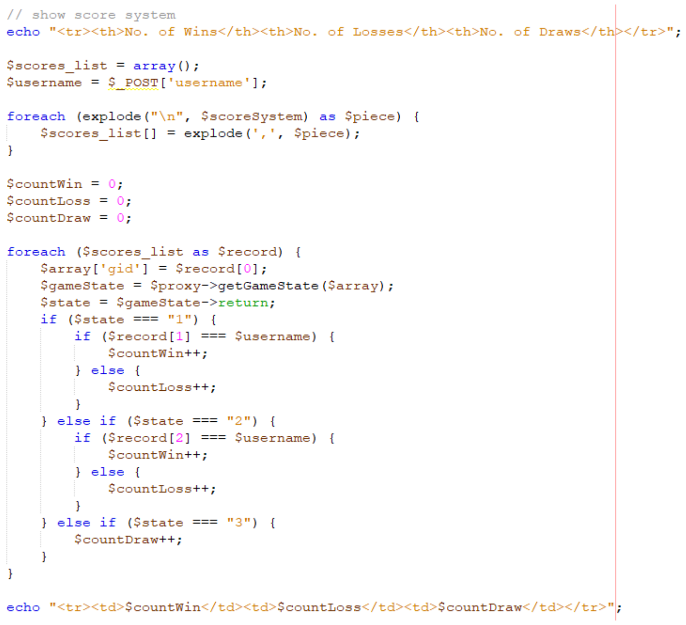
When the Register button clicked, it will compare the whether the two passwords entered are the same. If yes, register webservice will be executed, and the system will go to Login page to ask user to login to the system when no error occurs.

Impl for Score System



```
$(document).ready(function() {

    updateScoreSystem();

    $("#showUsername").html("USERNAME: " + username);

});

function updateScoreSystem() {

    $.ajax({
        type: 'POST',
        url: 'get_scores.php',
        data: {uid: uid, username: username},
        success: function(msg) {
            $("#scoreSystemTable").html(msg);
        }
    });

    timeout = setTimeout("updateScoreSystem()", 1000);
}
```

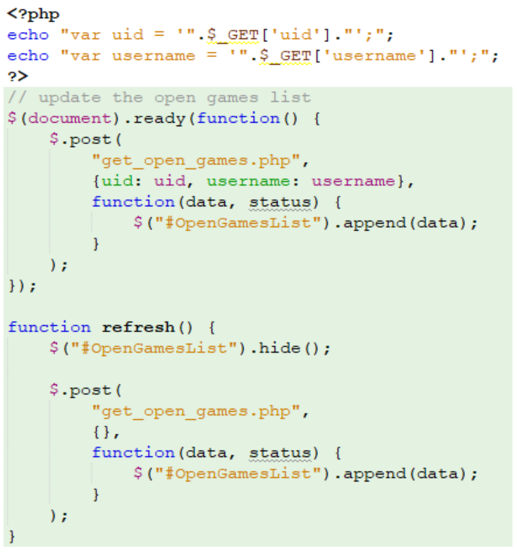
The main method used to implement score system is to refresh the score system per second. The setTimeout() method defines a method to execute every second. Ajax is used to get the scores and generate table for the score system.



```
// show score system
echo "<tr><th>No. of Wins</th><th>No. of Losses</th><th>No. of Draws</th></tr>";

$scores_list = array();
$username = $_POST['username'];

foreach (explode("\n", $scoreSystem) as $piece) {
    $scores_list[] = explode(',', $piece);
}

$countWin = 0;
$countLoss = 0;
$countDraw = 0;

foreach ($scores_list as $record) {
    $array['gid'] = $record[0];
    $gameState = $proxy->getGameState($array);
    $state = $gameState->return;
    if ($state === "1") {
        if ($record[1] === $username) {
            $countWin++;
        } else {
            $countLoss++;
        }
    } else if ($state === "2") {
        if ($record[2] === $username) {
            $countWin++;
        } else {
            $countLoss++;
        }
    } else if ($state === "3") {
        $countDraw++;
    }
}

echo "<tr><td>$countWin</td><td>$countLoss</td><td>$countDraw</td></tr>";
```

A list of strings are obtained by applying showAllMyGames() webservice, here we use explode() method to split the string into pieces and save data into an array. We can see the first element of each record is the gid, then we use getGameState() to obtain the status of the game with this gid. If the returned state is 1, then find out if the login user is player1 or player2. If the current user is player1, then increment win, otherwise, increment loss. And vice versa…

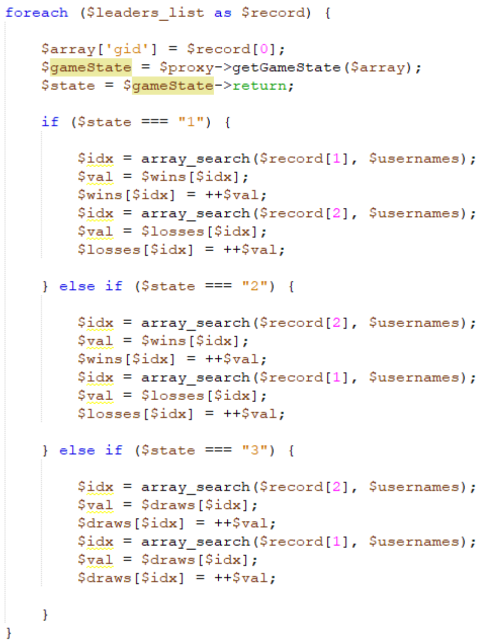
Impl for Open Game List



```
<?php
echo "var uid = '".$_GET['uid']."';";
echo "var username = '".$_GET['username']."';";
?>
// update the open games list
$(document).ready(function() {
    $.post(
        "get_open_games.php",
        {uid: uid, username: username},
        function(data, status) {
            $("#OpenGamesList").append(data);
        }
    );
});

function refresh() {
    $("#OpenGamesList").hide();

    $.post(
        "get_open_games.php",
        {},
        function(data, status) {
            $("#OpenGamesList").append(data);
        }
    );
}
```

Similar to the score system, the implementation of open game list also used a timeout method to refresh the screen. And Ajax is used to get the generated list.

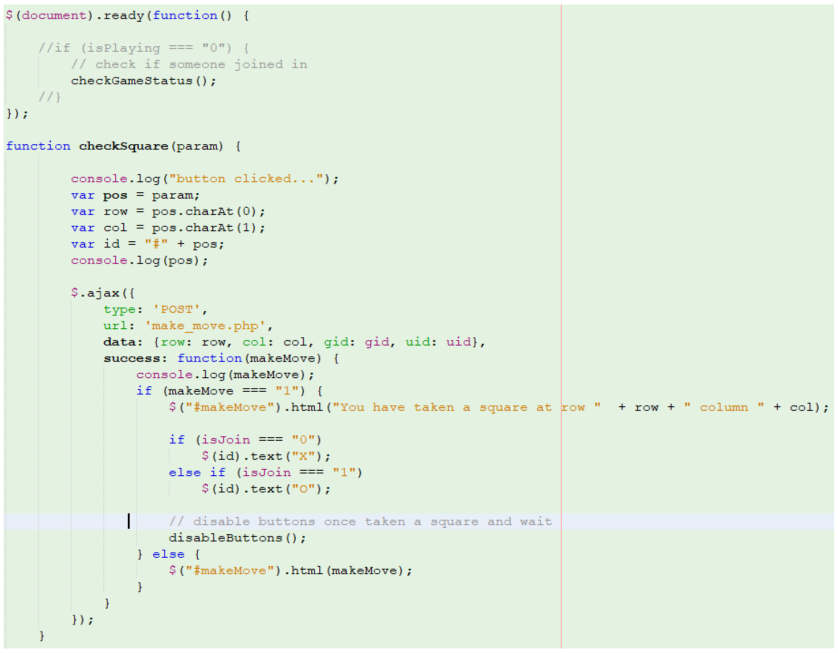
Impl for Leader Board System



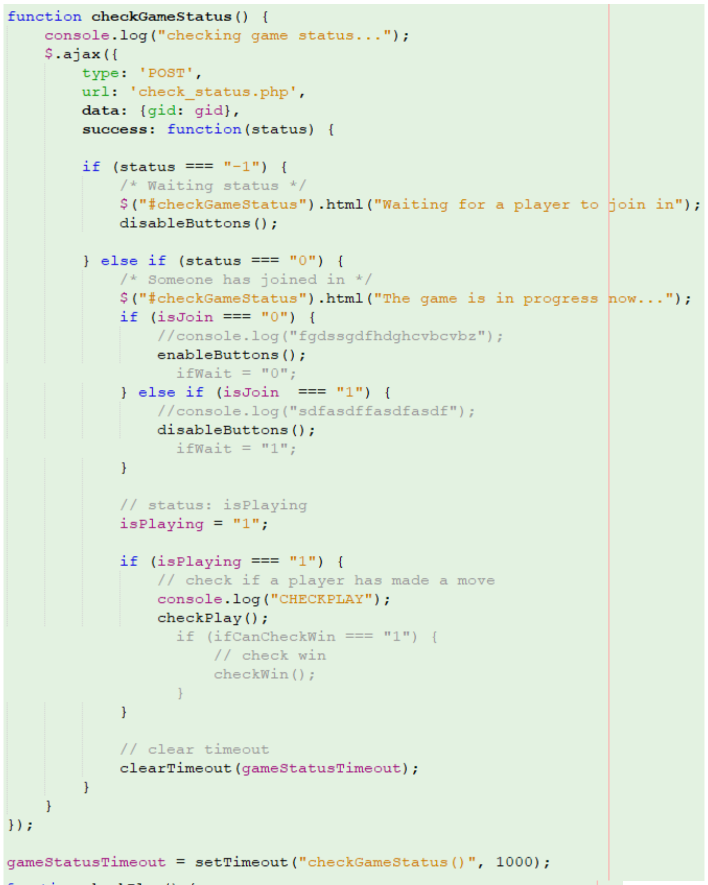
```
foreach ($leaders_list as $record) {

    $array['gid'] = $record[0];
    $gameState = $proxy->getGameState($array);
    $state = $gameState->return;

    if ($state === "1") {

        $idx = array_search($record[1], $usernames);
        $val = $wins[$idx];
        $wins[$idx] = ++$val;
        $idx = array_search($record[2], $usernames);
        $val = $losses[$idx];
        $losses[$idx] = ++$val;

    } else if ($state === "2") {

        $idx = array_search($record[2], $usernames);
        $val = $wins[$idx];
        $wins[$idx] = ++$val;
        $idx = array_search($record[1], $usernames);
        $val = $losses[$idx];
        $losses[$idx] = ++$val;

    } else if ($state === "3") {

        $idx = array_search($record[2], $usernames);
        $val = $draws[$idx];
        $draws[$idx] = ++$val;
        $idx = array_search($record[1], $usernames);
        $val = $draws[$idx];
        $draws[$idx] = ++$val;

    }
}
```

Similar to score system, the leader board system also has a timeout which allows the system to update the board regularly. The code snippet above shows the main implementation for the leader board. Firstly, the leagueTable() webservice is applied to obtain a list of strings. And then we can use explode() method to split string into pieces. All the registered users can be easily accessed from the webservice. Then use getGameState() to check the status of the game. When the state is 1, that means for a specific game when the current user was player 1, he/she won, if the current user played as second player, he lost the game. And similar for the state 2 and 3.

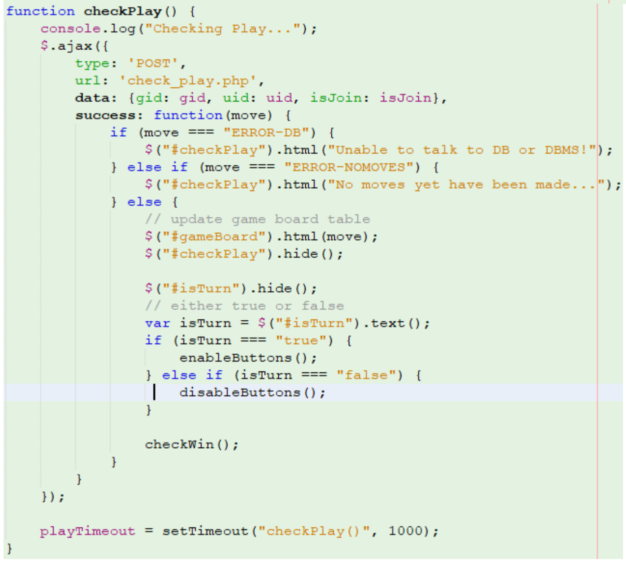
Impl for Play



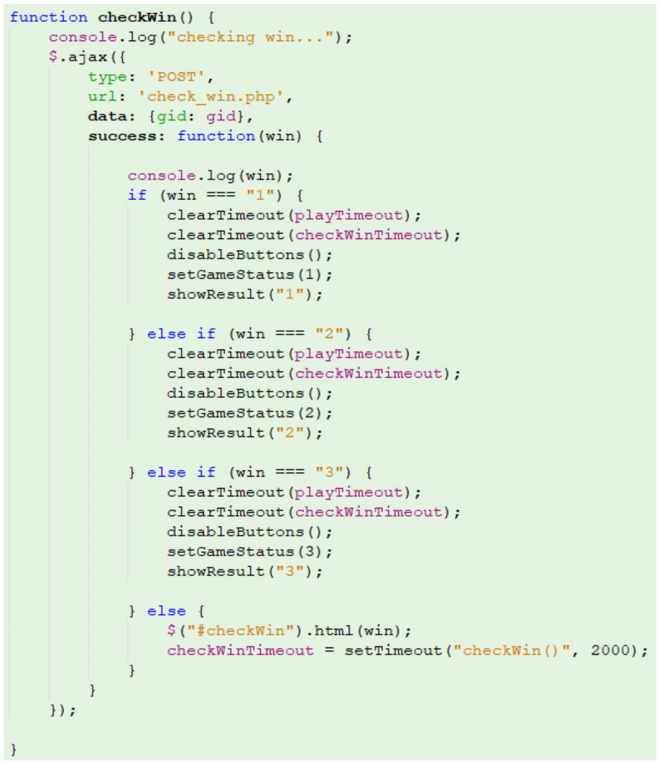
```
$(document).ready(function() {

    //if (isPlaying === "0") {
        // check if someone joined in
        checkGameStatus();
    //}
});

function checkSquare(param) {

        console.log("button clicked...");
        var pos = param;
        var row = pos.charAt(0);
        var col = pos.charAt(1);
        var id = "#" + pos;
        console.log(pos);

        $.ajax({
            type: 'POST',
            url: 'make_move.php',
            data: {row: row, col: col, gid: gid, uid: uid},
            success: function(makeMove) {
                console.log(makeMove);
                if (makeMove === "1") {
                    $("#makeMove").html("You have taken a square at row "  + row + " column " + col);

                    if (isJoin === "0")
                        $(id).text("X");
                    else if (isJoin === "1")
                        $(id).text("O");

                    // disable buttons once taken a square and wait
                    disableButtons();
                } else {
                    $("#makeMove").html(makeMove);
                }
            }
        });
    }
```



```
function checkGameStatus() {
    console.log("checking game status...");
    $.ajax({
        type: 'POST',
        url: 'check_status.php',
        data: {gid: gid},
        success: function(status) {

        if (status === "-1") {
            /* Waiting status */
            $("#checkGameStatus").html("Waiting for a player to join in");
            disableButtons();

        } else if (status === "0") {
            /* Someone has joined in */
            $("#checkGameStatus").html("The game is in progress now...");
            if (isJoin === "0") {
                //console.log("fgdssgdfhdghcvbcvbz");
                enableButtons();
                  ifWait = "0";
            } else if (isJoin  === "1") {
                //console.log("sdfasdffasdfasdf");
                disableButtons();
                  ifWait = "1";
            }

            // status: isPlaying
            isPlaying = "1";

            if (isPlaying === "1") {
                // check if a player has made a move
                console.log("CHECKPLAY");
                checkPlay();
                  if (ifCanCheckWin === "1") {
                      // check win
                      checkWin();
                  }
            }

            // clear timeout
            clearTimeout(gameStatusTimeout);
        }
    }
});

gameStatusTimeout = setTimeout("checkGameStatus()", 1000);
```



```
function checkPlay() {
    console.log("Checking Play...");
    $.ajax({
        type: 'POST',
        url: 'check_play.php',
        data: {gid: gid, uid: uid, isJoin: isJoin},
        success: function(move) {
            if (move === "ERROR-DB") {
                $("#checkPlay").html("Unable to talk to DB or DBMS!");
            } else if (move === "ERROR-NOMOVES") {
                $("#checkPlay").html("No moves yet have been made...");
            } else {
                // update game board table
                $("#gameBoard").html(move);
                $("#checkPlay").hide();

                $("#isTurn").hide();
                // either true or false
                var isTurn = $("#isTurn").text();
                if (isTurn === "true") {
                    enableButtons();
                } else if (isTurn === "false") {
                    disableButtons();
                }

                checkWin();
            }
        }
    });

    playTimeout = setTimeout("checkPlay()", 1000);
}
```



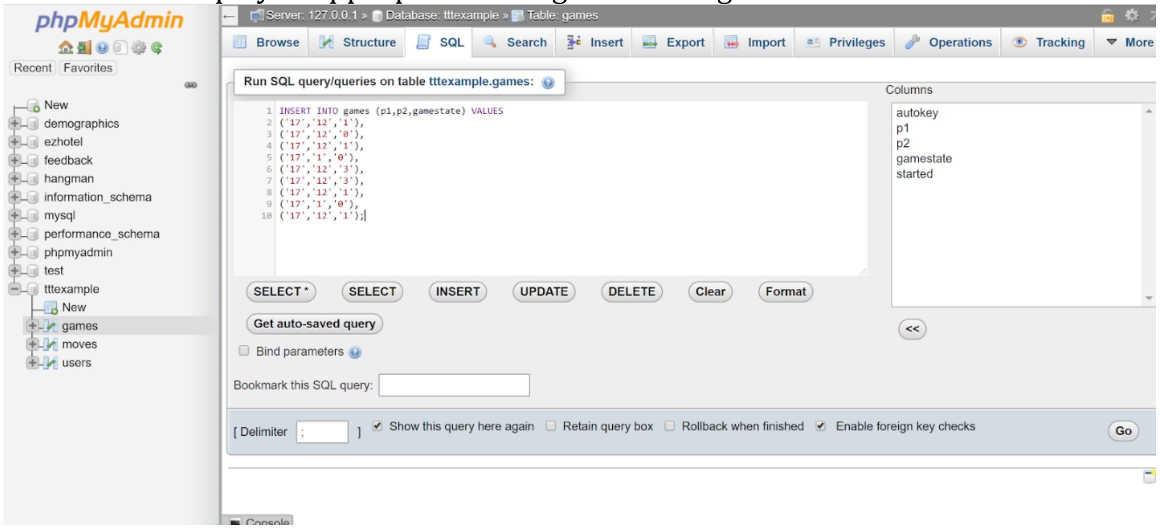
```
function checkWin() {
    console.log("checking win...");
    $.ajax({
        type: 'POST',
        url: 'check_win.php',
        data: {gid: gid},
        success: function(win) {

            console.log(win);
            if (win === "1") {
                clearTimeout(playTimeout);
                clearTimeout(checkWinTimeout);
                disableButtons();
                setGameStatus(1);
                showResult("1");

            } else if (win === "2") {
                clearTimeout(playTimeout);
                clearTimeout(checkWinTimeout);
                disableButtons();
                setGameStatus(2);
                showResult("2");

            } else if (win === "3") {
                clearTimeout(playTimeout);
                clearTimeout(checkWinTimeout);
                disableButtons();
                setGameStatus(3);
                showResult("3");

            } else {
                $("#checkWin").html(win);
                checkWinTimeout = setTimeout("checkWin()", 2000);
            }
        }
    });

}
```

The implementation for Play Game part is a bit tricky compared to the previous ones. When the Play page is ready, we execute a function called checkGameStatus() and this function will keep checking if another user has joined in this game by using timeout. When the system noticed a user has joined in the current game, it will kill the checkGameStatus timeout and move to another one called checkPlay(). This method is to check if a user has taken a move or not. In the checkPlay() method, we invoke getBoard() webservice to update the game board on both sides. And the system will also check if it is the time for the current user to play. If yes, the game board will be unlocked for the current user; otherwise, the board is locked and the user needs to wait. When a user wants to take a move, a checkSquare() function will be executed to check if the current square has been taken or not. If the square has been taken, an error message will be shown and ask user to try another one. After the system has detected one move has been made for the game, a checkWin() timeout will be executed. This method keeps checking if either side wins the game or draws (when no one can move anymore). If the result can be declared, the checkWin() timeout will be cleared and the board will be disabled for both users, all rests of timeouts will be cleared as well. The game terminates. Then the system will use setGameState() to record the result into database.

Test

Java Part

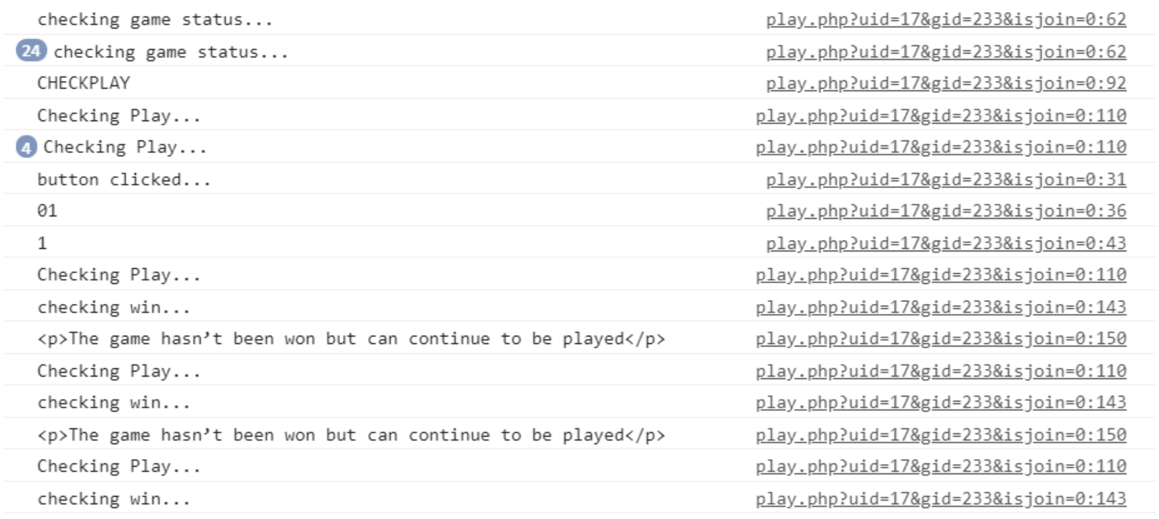
The testing for the Java implementation mainly involved manually adding SQL statements to insert dummy data into the database and checking if certain data appeared in the tables. The dummy data helped to check if the login and registration worked and displayed appropriate dialog box messages.



```sql
INSERT INTO games (p1,p2,gamestate) VALUES
('17','12','1'),
('17','12','0'),
('17','12','1'),
('17','1','0'),
('17','12','3'),
('17','12','3'),
('17','12','1'),
('17','1','0'),
('17','12','1');
```

PHP Part

The test procedure for PHP part mainly uses console log built in JavaScript. In order to test if a function or a piece of code can be executed, we can use this way to test it. And to test whether a parameter is successfully passed, simply put the parameter to be tested into the log function and check its output in the console.



```
   checking game status...                                      play.php?uid=17&gid=233&isjoin=0:62
24 checking game status...                                      play.php?uid=17&gid=233&isjoin=0:62
   CHECKPLAY                                                    play.php?uid=17&gid=233&isjoin=0:92
   Checking Play...                                             play.php?uid=17&gid=233&isjoin=0:110
4  Checking Play...                                             play.php?uid=17&gid=233&isjoin=0:110
   button clicked...                                            play.php?uid=17&gid=233&isjoin=0:31
   01                                                           play.php?uid=17&gid=233&isjoin=0:36
   1                                                            play.php?uid=17&gid=233&isjoin=0:43
   Checking Play...                                             play.php?uid=17&gid=233&isjoin=0:110
   checking win...                                              play.php?uid=17&gid=233&isjoin=0:143
   <p>The game hasn’t been won but can continue to be played</p> play.php?uid=17&gid=233&isjoin=0:150
   Checking Play...                                             play.php?uid=17&gid=233&isjoin=0:110
   checking win...                                              play.php?uid=17&gid=233&isjoin=0:143
   <p>The game hasn’t been won but can continue to be played</p> play.php?uid=17&gid=233&isjoin=0:150
   Checking Play...                                             play.php?uid=17&gid=233&isjoin=0:110
   checking win...                                              play.php?uid=17&gid=233&isjoin=0:143
```