DALManager Class:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

import com.andrew.db.DBUtil;
import com.andrew.model.Game;
import com.andrew.model.Question;
import com.andrew.model.Team;

public class DALManager {
	static Team aTeam = new Team();
	public static void saveQuestion(Question question, String process) {
		// deletes, adds, or edits the question base on the String
process
		Connection connection = DBUtil.getDbConnection();
		DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd
HH:mm:ss");
		DateTimeFormatter yr = DateTimeFormatter.ofPattern("yyyy");
		LocalDateTime now = LocalDateTime.now();
		if (process.equalsIgnoreCase("add")) {
			if (connection != null) {
				try {
					Statement statement =
connection.createStatement();
					String sql = "insert into
question(question_id, question_description, question_choiceA,
question_choiceB, question_choiceC, question_choiceD, correct_answer,
time_limit, explaination, last_update_time, year, section_number, showAnswer,
ID)values('"+question.getQuestionOrder()+"','"+question.getQuestion()+"','"+qu
estion.getAnswerchoiceA()+"','"+question.getAnswerchoiceB()+"','"+question.get
AnswerchoiceC()+"','"+question.getAnswerchoiceD()+"','"+question.getCorrectAns
wer()+"','"+question.getTimelimit()+"','"+question.getExplaination()+"','"+dtf
.format(now)+"','"+yr.format(now)+"','"+question.getTimelimit()+"','"+0+"','"+
question.getQuestionID()+"')";
					int rlt = statement.executeUpdate(sql);
				} catch (SQLException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
			}
		}
		else if (process.equalsIgnoreCase("delete")) {
			System.out.println(question.getQuestionID());
			if (connection != null) {
				try {
```

```
                                String sql = "delete from question where
ID=?";
                                PreparedStatement ps =
connection.prepareStatement(sql);
                                ps.setDouble(1, question.getQuestionID());
                                int rlt = ps.executeUpdate();

                          } catch (SQLException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                          }
                    }
              }
              else if (process.equalsIgnoreCase("edit")) {
                    try {
                          String sql = "UPDATE question SET
question_description=?, question_choiceA=?, question_choiceB=?,
question_choiceC=?, question_choiceD=?,
correct_answer='"+question.getCorrectAnswer()+"', time_limit=?,
explaination=?, year=?, Last_update_time=?, showAnswer="+0+", question_id=?"+"
WHERE ID=?";
                          PreparedStatement ps =
connection.prepareStatement(sql);
                          System.out.println("updating");
                          ps.setString(1, question.getQuestion());
                          ps.setString(2, question.getAnswerchoiceA());
                          ps.setString(3, question.getAnswerchoiceB());
                          ps.setString(4, question.getAnswerchoiceC());
                          ps.setString(5, question.getAnswerchoiceD());
                          ps.setInt(6, question.getTimelimit());
                          ps.setString(7, question.getExplaination());
                          ps.setString(8, yr.format(now));
                          ps.setString(9, dtf.format(now));
                          ps.setInt(10, question.getQuestionOrder());
                          ps.setDouble(11, question.getQuestionID());
                          int rlt = ps.executeUpdate();
                    }
                    catch (SQLException e) {
                          // TODO Auto-generated catch block
                          e.printStackTrace();
                    } finally {
                          if(connection!=null) {
                                try {
                                      connection.close();
                                } catch (SQLException e) {
                                      // TODO Auto-generated catch block
                                      e.printStackTrace();
                                }
                          }
                    }
              }
        }

        public static Question getQuestionByID(int num) {
              // gets the quesiton the quiz is currently on
```

```
		Question aquestion = new Question();
		DateTimeFormatter yr = DateTimeFormatter.ofPattern("yyyy");
		LocalDateTime now = LocalDateTime.now();
		String current = yr.format(now);
		Connection connection = DBUtil.getDbConnection();
		System.out.println("int num="+num);
		if(connection!=null) {
			try {
				String sql = "SELECT * FROM question WHERE
question_id="+num+" and year="+current;
				PreparedStatement ps =
connection.prepareStatement(sql);
				ResultSet rlt = ps.executeQuery();
				while(rlt.next()) {
					String desc =
rlt.getString("question_description");
					aquestion.setQuestionID(num);
					aquestion.setQuestion(desc);

	aquestion.setAnswerchoiceA(rlt.getString("question_choiceA"));

	aquestion.setAnswerchoiceB(rlt.getString("question_choiceB"));

	aquestion.setAnswerchoiceC(rlt.getString("question_choiceC"));

	aquestion.setAnswerchoiceD(rlt.getString("question_choiceD"));

	aquestion.setCorrectAnswer(rlt.getString("correct_answer").charAt(0));

	aquestion.setExplaination(rlt.getString("explaination"));

	aquestion.setShowExplaination(rlt.getInt("showAnswer"));

	aquestion.setTimelimit(rlt.getInt("time_limit"));
					break;
				}
			} catch (SQLException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			} finally {
				if(connection!=null) {
					try {
						connection.close();
					} catch (SQLException e) {
						// TODO Auto-generated catch block
						e.printStackTrace();
					}
				}
			}
		}
		return aquestion;
	}

	public static List<Question> loadQuestions() {	// returns list of
an arraylist of the object Question
```

```
            List<Question> list = new ArrayList<>();
            DateTimeFormatter yr = DateTimeFormatter.ofPattern("yyyy");
            LocalDateTime now = LocalDateTime.now();
            String current = yr.format(now);
            Connection connection = DBUtil.getConnection(); // establishes
connection with database
            if(connection!=null) { // if connection is successful
                  try {
                        String sql = "SELECT * FROM question WHERE
year="+current; // Select everything from the 'question' table for this year
                        PreparedStatement ps;
                        ps = connection.prepareStatement(sql);
                        ResultSet rlt = ps.executeQuery();
                        while(rlt.next()) {  // whilst there are more
questions
                              Question aquestion = new Question();
                              aquestion.setQuestionID(rlt.getDouble("ID"));

      aquestion.setQuestionOrder(rlt.getInt("question_id"));

      aquestion.setQuestion(rlt.getString("question_description"));

      aquestion.setAnswerchoiceA(rlt.getString("question_choiceA"));

      aquestion.setAnswerchoiceB(rlt.getString("question_choiceB"));

      aquestion.setAnswerchoiceC(rlt.getString("question_choiceC"));

      aquestion.setAnswerchoiceD(rlt.getString("question_choiceD"));

      aquestion.setCorrectAnswer(rlt.getString("correct_answer").charAt(0));

      aquestion.setTimelimit(rlt.getInt("time_limit"));

      aquestion.setExplaination(rlt.getString("explaination"));
                              list.add(aquestion);
                        }
                  } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                  } finally {
                        if(connection!=null) {
                              DBUtil.closeConnection(); // lastly, close
connection with database
                        }
                  }
            }
            return list;
      }

      public static List<Question> loadQuestionsFromPast() {     // returns
list of an arraylist of the object Question
            List<Question> list = new ArrayList<>();
            Connection connection = DBUtil.getConnection(); // establishes
connection with database
```

```
		if(connection!=null) { // if connection is successful
			try {
				String sql = "SELECT * FROM question"; // Select
everything from the 'question' table
				PreparedStatement ps;
				ps = connection.prepareStatement(sql);
				ResultSet rlt = ps.executeQuery();
				while(rlt.next()) {  // whilst there are more
questions
					Question aquestion = new Question();
					aquestion.setQuestionID(rlt.getInt("year"));

	aquestion.setQuestion(rlt.getString("question_description"));

	aquestion.setAnswerchoiceA(rlt.getString("question_choiceA"));

	aquestion.setAnswerchoiceB(rlt.getString("question_choiceB"));

	aquestion.setAnswerchoiceC(rlt.getString("question_choiceC"));

	aquestion.setAnswerchoiceD(rlt.getString("question_choiceD"));

	aquestion.setCorrectAnswer(rlt.getString("correct_answer").charAt(0));

	aquestion.setTimelimit(rlt.getInt("time_limit"));

	aquestion.setExplaination(rlt.getString("explaination"));
					list.add(aquestion);
				}
			} catch (SQLException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			} finally {
				if(connection!=null) {
					DBUtil.closeConnection(); // lastly, close
connection with database
				}
			}
		}
		return list;
	}

	public static int loadGame() {
		Connection connection = DBUtil.getConnection();
		Game game = new Game(); // creates new Game object
		List<Game> list = new ArrayList<>();
		int id = GameManagement.generateGameCode(); // generates game
code
		DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd
HH:mm:ss"); // gets current date and time
		DateTimeFormatter yr = DateTimeFormatter.ofPattern("yyyy"); //
gets current year
		LocalDateTime now = LocalDateTime.now();
		String name = "TriviaNight"+yr.format(now);
		if(connection!=null) {
```

```
                        try { // adds game data into database
                                String sql = "insert into game(Game_ID,
Game_Name, Start_Time,
Game_Process)values('"+id+"','"+name+"','"+dtf.format(now)+"','"+0+"')";
                                Statement statement =
connection.createStatement();
                                int rlt = statement.executeUpdate(sql);
                        } catch (SQLException e) {
                                e.printStackTrace();
                        } finally {
                                if(connection!=null) {
                                        DBUtil.closeConnection();
                                }
                        }
                }
                return id;
        }

        public static Game loadGameInfo(int id) {
                // returns the Game object of the current game
                Game aGame = new Game();
                Connection connection = DBUtil.getConnection();
                if(connection!=null) {
                        try {
                                String sql = "SELECT * FROM game WHERE Game_ID="+id;
                                PreparedStatement ps =
connection.prepareStatement(sql);
                                ResultSet rlt = ps.executeQuery();
                                aGame.setGameID(rlt.getInt("Game_ID"));
                                aGame.setGameName(rlt.getString("Game_Name"));
                                aGame.setGameStartTime(rlt.getString("Start_Time"));
                                aGame.setHasGameStarted(rlt.getInt("Game_Process"));

                        } catch (SQLException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } finally {
                                if(connection!=null) {
                                        DBUtil.closeConnection();
                                }
                        }
                }
                return aGame;
        }

        public static int TeamEnter(String team, int gamecode) {
                // when a team joins the quiz, adds the team into database and
edits the static team object
                Connection connection = DBUtil.getConnection();
                Random rnd = new Random();
                int n = 10000 + rnd.nextInt(900000);
                int teamid = gamecode+n;
                if(connection!=null) {
                        try {
```

```
                                Statement statement =
connection.createStatement();
                                Statement statement2 =
connection.createStatement();
                                String sql = "insert into Team(Team_ID,
Team_Name)values('"+teamid+"','"+team+"')"; // adds team into
                                String sql2 = "insert into
Game_Team_bridge(Game_ID, Team_ID,
Team_points)values('"+gamecode+"','"+teamid+"','"+0+"')";
                                int rlt = statement.executeUpdate(sql);
                                int rlt2 = statement.executeUpdate(sql2);
                                aTeam.setTeam_ID(teamid);
                                System.out.println("printed");
                        } catch (SQLException e) {
                                e.printStackTrace();
                        } finally {
                                if(connection!=null) {
                                        DBUtil.closeConnection();
                                }
                        }
                }
                return teamid;
        }

        public static boolean codecheck(int code) {
                // checks if the code is valid
                Connection connection = DBUtil.getConnection();
                if(connection!=null) {
                        try {
                                String sql = "SELECT * FROM Game";
                                PreparedStatement ps;
                                ps = connection.prepareStatement(sql);
                                ResultSet rlt = ps.executeQuery();
                                while(rlt.next()) {
                                        if (code == rlt.getInt("Game_ID")) {
                                                return true;
                                        }
                                }
                                return false;
                        } catch (SQLException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } finally {
                                if(connection!=null) {
                                        DBUtil.closeConnection();
                                }
                        }
                }
                return false;
        }

        public static void getNextQuestion(int id) {
                // sets the game to the next question
                Connection connection = DBUtil.getDbConnection();
                if(connection!=null) {
```

```
                        try {
                                String sql = "UPDATE Game SET
Game_Process=Game_Process+1 WHERE Game_ID="+id;
                                PreparedStatement ps =
connection.prepareStatement(sql);
                                int rlt = ps.executeUpdate();
                        } catch (SQLException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } finally {
                                if(connection!=null) {
                                        try {
                                                connection.close();
                                        } catch (SQLException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        };
                                }
                        }
                }
        }

        public static void getPrevQuestion(int id) {
                int q = getCurrentQuestion(findCurrentGame());
                Connection connection = DBUtil.getDbConnection();
                if(connection!=null) {
                        try {
                                String sql = "UPDATE Game SET
Game_Process=Game_Process-1 WHERE Game_ID="+id;
                                String sql2 =  "UPDATE question SET
showAnswer=0 WHERE question_id="+q+"-1";
                                PreparedStatement ps =
connection.prepareStatement(sql);
                                int rlt = ps.executeUpdate();
                                PreparedStatement ps2 =
connection.prepareStatement(sql2);
                                int rlt2 = ps2.executeUpdate();
                        } catch (SQLException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } finally {
                                if(connection!=null) {
                                        try {
                                                connection.close();
                                        } catch (SQLException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        };
                                }
                        }
                }
        }

        public static int getCurrentQuestion(int id) {
                // returns the current question the quiz is on
```

```
		int currentQuestion =-1;
		Connection connection = DBUtil.getDbConnection();
		if(connection!=null) {
			try {
					String sql = "SELECT Game_Process FROM Game
WHERE Game_ID="+id;
					PreparedStatement ps =
connection.prepareStatement(sql);
					ResultSet rlt = ps.executeQuery();
					while(rlt.next()) {
						currentQuestion =
rlt.getInt("Game_Process");
						break;
					}
			} catch (SQLException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			} finally {
				if(connection!=null) {
					try {
						connection.close();
					} catch (SQLException e) {
						// TODO Auto-generated catch block
						e.printStackTrace();
					}
				}
			}
		}
		return currentQuestion;


	}

	public static int findCurrentGame() {
		// finds the current active game
		int game = -1;
		Connection dbConnection = DBUtil.getDbConnection();
		if(dbConnection!=null) {
			try {
					String sql = "SELECT Game_ID FROM Game WHERE
Game_Process>-1";
					PreparedStatement ps =
dbConnection.prepareStatement(sql);
					ResultSet rlt = ps.executeQuery();
					while(rlt.next()) {
						game = rlt.getInt("Game_ID");
						break;
					}

					rlt.close();
					ps.close();
			} catch (SQLException e) {
				e.printStackTrace();
			} finally {
				if(dbConnection!=null) {
```

```
                                try {
                                        dbConnection.close();
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                        }
                }
        }
        return game;
    }

    public static void endGame() {
        // ends the current game
        Connection connection = DBUtil.getDbConnection();
        System.out.println("code aquired");
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd
HH:mm:ss");
        LocalDateTime now = LocalDateTime.now();
        if(connection!=null) {
                try {
                                String sql = "UPDATE Game SET Game_Process=-
1, End_Time='"+dtf.format(now)+"'WHERE Game_Process!=-1";
                                PreparedStatement ps =
connection.prepareStatement(sql);
                                int rlt= ps.executeUpdate();
                                String sql2 = "UPDATE question SET
showAnswer=0";
                                PreparedStatement ps2 =
connection.prepareStatement(sql2);
                                int rlt2 = ps2.executeUpdate();
                } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } finally {
                        if(connection!=null) {
                                try {
                                        connection.close();
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                        }
                }
        }
    }

    public static void saveTeamInfo(int team_ID, String team_Name) {
        aTeam.setTeam_ID(team_ID);
        aTeam.setTeam_Name(team_Name);
    }

    public static Team getTeamInfo() {
        return aTeam;
    }

    public static void showAnswer() {
```

```
		// shows answer to the current question
		int q = findCurrentQuestion(findCurrentGame());
		Connection connection = DBUtil.getConnection();
		System.out.println(q);
		if(connection!=null) {
			try {
					String sql = "UPDATE question SET
showAnswer=1 WHERE question_id="+q;
					PreparedStatement ps =
connection.prepareStatement(sql);
					int rlt = ps.executeUpdate();
			} catch (SQLException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			} finally {
				if(connection!=null) {
					DBUtil.closeConnection();
				}
			}
		}
	}

	public static void addPoints() {
		// add points to the team for answering correctly
		int currentGame = findCurrentGame();
		Team team = DALManager.getTeamInfo();
		Connection connection = DBUtil.getConnection();
		if (connection!=null) {
			try {
				System.out.println("gamecode="+currentGame+",
Teamcode="+team.getTeam_ID());
				String sql = "UPDATE Game_Team_bridge SET
Team_points=Team_points+100 WHERE Game_ID="+currentGame+" and
Team_ID="+team.getTeam_ID();
				PreparedStatement ps =
connection.prepareStatement(sql);
				int rlt = ps.executeUpdate();
			} catch (SQLException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			} finally {
				if(connection!=null) {
					DBUtil.closeConnection();
				}
			}
		}
	}
}
```

GameManagement class:

```
import java.util.Random;
```

```
public class GameManagement {
	// generates game code when game is hosted
	public static int generateGameCode() {
		Random rnd = new Random();
		int n = 10000 + rnd.nextInt(900000);
		return n;
	}
}
```

QuestionManagement class:

```
import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.List;


import com.andrew.db.DBUtil;

import com.andrew.model.Question;


public class QuestionManagement {


	public static void createQuestion(double qID, int id, String q, String a, String b, String c,
String d, int time, char answer, String explaination) {

		Question question = new Question();

		question.setQuestionID(qID);

		question.setQuestionOrder(id);

		question.setQuestion(q);

		question.setAnswerchoiceA(a);
```

```
		question.setAnswerchoiceB(b);
		question.setAnswerchoiceC(c);
		question.setAnswerchoiceD(d);
		question.setTimelimit(time);
		question.setCorrectAnswer(answer);
		question.setExplaination(explaination);
		System.out.println(b);
		DALManager.saveQuestion(question,"add");
		}

	public static void updateUser(double qID, int id, String q, String a, String b, String c,
String d, int time, char answer, String explaination) {
		Question question = new Question();
		question.setQuestionID(qID);
		question.setQuestionOrder(id);
		question.setQuestion(q);
		question.setAnswerchoiceA(a);
		question.setAnswerchoiceB(b);
		question.setAnswerchoiceC(c);
		question.setAnswerchoiceD(d);
		question.setTimelimit(time);
		question.setCorrectAnswer(answer);
		question.setExplaination(explaination);
		System.out.print("HELPME");
		DALManager.saveQuestion(question, "edit");
	}

	public static void deleteQuestion(double id) {
		DALManager manager = new DALManager();
```

```
                List<Question> list = manager.loadQuestions();
                int i=0;
                while(list.get(i).getQuestionID()!=id) {
                        i++;
                }
                DALManager.saveQuestion(list.get(i), "delete");
        }

        public static boolean validateGame() {
                List<Integer> order = new ArrayList();
                DateTimeFormatter yr = DateTimeFormatter.ofPattern("yyyy");
                LocalDateTime now = LocalDateTime.now();
                String current = yr.format(now);
                Connection connection = DBUtil.getConnection(); // establishes connection with
database
                if(connection!=null) {
                        try {
                                String sql = "SELECT question_ID FROM question WHERE
year="+current;
                                PreparedStatement ps = connection.prepareStatement(sql);
                                ResultSet rlt = ps.executeQuery();
                                while (rlt.next()) {
                                        int i=rlt.getInt("question_ID");
                                        System.out.println(i);
                                        order.add(i);
                                }
                                Collections.sort(order);
                                for (int j=0;j<order.size()-1;j++) {
                                        if
(order.get(j)==order.get(j+1)||order.get(j)!=order.get(j+1)-1) {
```

```
                                        return false;
                                }
                        }
                        return true;

                } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } finally {
                        if(connection!=null) {
                                DBUtil.closeConnection();
                        }
                }
        }
        return false;
}

public static double getID(int order) {
        DateTimeFormatter yr = DateTimeFormatter.ofPattern("yyyy");
        LocalDateTime now = LocalDateTime.now();
        String current = yr.format(now);
        Connection connection = DBUtil.getConnection(); // establishes connection with
database
        System.out.println(current);
        System.out.println(order);
        if(connection!=null) {
                try {
                        String sql = "SELECT ID FROM question WHERE year="+current+"
and question_id="+order;
```

```
                        PreparedStatement ps = connection.prepareStatement(sql);
                        ResultSet rlt = ps.executeQuery();
                        return rlt.getDouble("ID");
                    } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    } finally {
                        if(connection!=null) {
                            DBUtil.closeConnection();
                        }
                    }
                }
                return 0;
            }

            public static int LastQ() {
                DateTimeFormatter yr = DateTimeFormatter.ofPattern("yyyy");
                LocalDateTime now = LocalDateTime.now();
                String current = yr.format(now);
                Connection connection = DBUtil.getDbConnection();; // establishes connection
with database
                if(connection!=null) {
                    try {
                        String sql = "SELECT MAX(question_ID) FROM question WHERE
year="+current;
                        PreparedStatement ps = connection.prepareStatement(sql);
                        ResultSet rlt = ps.executeQuery();
                        return rlt.getInt("MAX(question_ID)");
                    } catch (SQLException e) {
```

```
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } finally {
                    if(connection!=null) {
                        try {
                            connection.close();
                        } catch (SQLException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        };
                    }
                }
            }
            return 1;
        }

        public static ArrayList<String> returnNames(ArrayList<Integer>list){
            ArrayList<String>Ranking = new ArrayList<String>();
            for (int i:list) {
                Connection connection = DBUtil.getConnection(); // establishes
connection with database
                if(connection!=null) {
                    try {
                        String sql = "SELECT Team_Name FROM TEAM WHERE
Team_ID="+i;
                        PreparedStatement ps =
connection.prepareStatement(sql);
                        ResultSet rlt = ps.executeQuery();
                        Ranking.add(rlt.getString("Team_Name"));
                    } catch (SQLException e) {
```

```
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                } finally {
                                        if(connection!=null) {
                                                DBUtil.closeConnection();
                                        }
                                }
                        }
                }
                return Ranking;
        }
```

TeamManagement Class:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;

import com.andrew.db.DBUtil;

public class TeamManagement {
        public static ArrayList<String> returnNames(ArrayList<Integer>list){
                ArrayList<String>Ranking = new ArrayList<String>();
                for (int i:list) {
```

```
                    Connection connection = DBUtil.getConnection(); // establishes connection with database
                    if(connection!=null) {
                        try {
                            String sql = "SELECT Team_Name FROM TEAM WHERE Team_ID="+i;
                            PreparedStatement ps = connection.prepareStatement(sql);
                            ResultSet rlt = ps.executeQuery();
                            Ranking.add(rlt.getString("Team_Name"));
                        } catch (SQLException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        } finally {
                            if(connection!=null) {
                                DBUtil.closeConnection();
                            }
                        }
                    }
                }
                return Ranking;
            }

            public static ArrayList<ArrayList<Integer>> displayRanking() { // returns an arraylist of arraylists
                int id = DALManager.findCurrentGame();
                Connection connection = DBUtil.getConnection();
                HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>(); // HashMap mapping TeamID to Points
                ArrayList<ArrayList<Integer>> leaderboard = new ArrayList<ArrayList<Integer>>();
```

```
                if(connection!=null) {
                        try {
                                String sql = "SELECT * FROM Game_Team_bridge WHERE
Game_ID="+id;
                                PreparedStatement ps;
                                ps = connection.prepareStatement(sql);
                                ResultSet rlt = ps.executeQuery();
                                while(rlt.next()) {

        hm.put(rlt.getInt("Team_ID"),rlt.getInt("Team_points"));
                                }
                                ArrayList<Integer> temp = new ArrayList<Integer>(hm.keySet());
// temporary arry that holds the id of teams
                                ArrayList<Integer> points = new
ArrayList<Integer>(hm.values()); // array that holds the amount of points each team has
                                ArrayList<Integer> teams = new ArrayList<Integer>(); // empty
array
                                Collections.sort(points); // points sorted in ascending order
                                while (!temp.isEmpty()) {
                                        for (int i=0;i<points.size();i++) {  // enclosed for loop to
try match the teams with the points that is sorted
                                                for (int j=0; j<temp.size();j++){
                                                        if (hm.get(temp.get(j))==points.get(i)) {
                                                                teams.add(temp.get(j));
                                                                temp.remove(j);
                                                                break;
                                                        }
                                                }
                                        }
                                }
                                leaderboard.add(teams);
```

```
                    leaderboard.add(points);


                } catch (SQLException e) {
                    e.printStackTrace();
                } finally {
                    if(connection!=null) {
                        DBUtil.closeConnection();
                    }
                }
            }
            return leaderboard; // returns arraylist of arraylists with the first arrylist being the teams and second being points, both in
                                                    // ascending order of ranking
        }


}
```

DBUtil class:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;


public class DBUtil {
    public static Connection connection = null;
    public static Connection getConnection() {
        if(connection!=null) {
            try {
                if (!connection.isClosed()) {
                    return connection;
```

```
                }
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }



        try {
            Class.forName("org.sqlite.JDBC");
            connection = DriverManager.getConnection("jdbc:sqlite:Trivia.db");
            System.out.println("success");
        } catch (ClassNotFoundException | SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return connection;
}



public static Connection getDbConnection() {
        Connection dnconnection = null;
        try {
            Class.forName("org.sqlite.JDBC");
            dnconnection = DriverManager.getConnection("jdbc:sqlite:Trivia.db");
            System.out.println("success");
        } catch (ClassNotFoundException | SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
```

```
            }
            return dnconnection;
        }

        public static void closeConnection() {
            if(connection!=null) {
                try {
                    connection.close();
                } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
```

Game Class:

```
public class Game {
	int gameID;
	String gameName;
	String gameStartTime;
	String gameEndTIme;
	int hasGameStarted;
	public int getGameID() {
		return gameID;
	}
	public void setGameID(int gameID) {
		this.gameID = gameID;
	}
	public String getGameName() {
		return gameName;
	}
	public void setGameName(String gameName) {
		this.gameName = gameName;
	}
	public String getGameStartTime() {
		return gameStartTime;
	}
	public void setGameStartTime(String gameStartTime) {
		this.gameStartTime = gameStartTime;
	}
```

```
	public String getGameEndTIme() {
		return gameEndTIme;
	}
	public void setGameEndTIme(String gameEndTIme) {
		this.gameEndTIme = gameEndTIme;
	}
	public int getHasGameStarted() {
		return hasGameStarted;
	}
	public void setHasGameStarted(int hasGameStarted) {
		this.hasGameStarted = hasGameStarted;
	}
}
```

Team class:

```
public class Team {
	int team_ID;
	String team_Name;
	public int getTeam_ID() {
		return team_ID;
	}
	public void setTeam_ID(int team_ID) {
		this.team_ID = team_ID;
	}
	public String getTeam_Name() {
		return team_Name;
	}
	public void setTeam_Name(String team_Name) {
		this.team_Name = team_Name;
	}
}
```

Question Class:

```
import com.andrew.SRV.DALManager;

public class Question {
	private double questionID;
	private String question;
	private String answerchoiceA;
	private String answerchoiceB;
	private String answerchoiceC;
	private String answerchoiceD;
	private char correctAnswer;
	private String explaination;
	private int timelimit;
	private int showExplaination;
	private int questionOrder;
	public int getQuestionOrder() {
		return questionOrder;
	}
	public void setQuestionOrder(int questionOrder) {
```

```
		this.questionOrder = questionOrder;
	}

	public int getShowExplaination() {
		return showExplaination;
	}
	public void setShowExplaination(int showExplaination) {
		this.showExplaination = showExplaination;
	}
	public String getQuestion() {
		return question;
	}
	public void setQuestion(String question) {
		this.question = question;
	}
	public String getAnswerchoiceA() {
		return answerchoiceA;
	}
	public void setAnswerchoiceA(String answerchoiceA) {
		this.answerchoiceA = answerchoiceA;
	}
	public String getAnswerchoiceB() {
		return answerchoiceB;
	}
	public void setAnswerchoiceB(String answerchoiceB) {
		this.answerchoiceB = answerchoiceB;
	}
	public String getAnswerchoiceC() {
		return answerchoiceC;
	}
	public void setAnswerchoiceC(String answerchoiceC) {
		this.answerchoiceC = answerchoiceC;
	}
	public String getAnswerchoiceD() {
		return answerchoiceD;
	}
	public void setAnswerchoiceD(String answerchoiceD) {
		this.answerchoiceD = answerchoiceD;
	}
	public char getCorrectAnswer() {
		return correctAnswer;
	}
	public void setCorrectAnswer(char correctAnswer) {
		this.correctAnswer = correctAnswer;
	}
	public String getExplaination() {
		return explaination;
	}
	public void setExplaination(String explaination) {
		this.explaination = explaination;
	}
	public int getTimelimit() {
		return timelimit;
	}
	public void setTimelimit(int timelimit) {
```

```
            this.timelimit = timelimit;
      }
      public double getQuestionID() {
            return questionID;
      }
      public void setQuestionID(double questionID) {
            this.questionID = questionID;
      }
      public void showAnswer() {
            Question question = new Question();
            int id = DALManager.findCurrentGame();
            question
=DALManager.getQuestionByID(DALManager.getCurrentQuestion(id));
            question.setQuestionID(1);
            DALManager.saveQuestion(question, "edit");
      }
}
```

AdminServlet Class:

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.andrew.SRV.DALManager;

/**
 * Servlet implementation class AdminServlet
 */
@WebServlet("/Admindo")
public class AdminServlet extends HttpServlet {
      private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet(/Admindo)
     */
    public AdminServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

      /**
       * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
       */
      protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            String act = request.getParameter("action");
            System.out.println(act);
            if ("Next Question".equalsIgnoreCase(act)) { // if the next
button is pressed
```

```
                        DALManager.getNextQuestion(DALManager.findCurrentGame());
// moves to next q for the current game
                        response.sendRedirect("admin.html"); // refreshes page

                }
                else if("End Game".equalsIgnoreCase(act)) { // if end game button
is pressed
                        response.sendRedirect("Leaderboard.html"); // redirects to
the Leaderboard page
                }
                else if ("Show Explaination".equalsIgnoreCase(act)) { // if show
explaination is pressed
                        DALManager.showAnswer(); // shows answer
                        response.sendRedirect("admin.html"); // refreshes page
                }
        }

        /**
         * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
         */
        protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
                String username = request.getParameter("user");
                String pwd = request.getParameter("password");
                int code = Integer.parseInt(request.getParameter("code"));
                if("interact".equals(username) && "god".equals(pwd)&&
DALManager.codecheck(code)) { // if username and password are correct
                        response.sendRedirect("admin.html"); // redirects to new
page
                }else {
                        response.sendRedirect("LoginPage.html"); // nothing
happens
                }
        }
```

GameControlServlet class:

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.andrew.SRV.DALManager;
import com.andrew.model.Question;

/**
 * Servlet implementation class GameControlServlet
 */
@WebServlet("/adminEnd")
public class GameControlServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;
```

```
    /**
     * @see HttpServlet#HttpServlet()
     */
    public GameControlServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		DALManager.endGame(); //ends game
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		DALManager.showAnswer(); // shows correct answer
		response.sendRedirect("admin.html"); // redirects
	}
}
```

GameHolder Servlet Class:

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.andrew.SRV.DALManager;
import com.andrew.SRV.QuestionManagement;
import com.andrew.model.Question;
import com.google.gson.Gson;

/**
 * Servlet implementation class GameHolderServlet
 */
@WebServlet("/gameholder")
public class GameHolderServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;
```

```
    /**
     * @see HttpServlet#HttpServlet()
     */
    public GameHolderServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		response.setContentType("application/json; charset=utf-8");
		response.setCharacterEncoding("UTF-8");
		Question aquestion = new Question();
		int id = DALManager.findCurrentGame(); // gets current question
number
		int questionNum = DALManager.getCurrentQuestion(id); // gets last
question number
		if (questionNum>QuestionManagement.LastQ()){  // if question
number exceeds the last q
			DALManager.endGame(); // end game
		}
				aquestion = DALManager.getQuestionByID(questionNum);
				Gson agson = new Gson();
				String jsonObj = agson.toJson(aquestion);
				PrintWriter out = response.getWriter();
				out.print(jsonObj);
				out.flush();
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		System.out.println("adding points");
		DALManager.addPoints(); // adds points if answered correctly
	}

}
```

GameServlet Class:

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
```

```java
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.andrew.SRV.DALManager;
import com.andrew.SRV.GameManagement;
import com.andrew.TriviaAPP.ResponseUtil;
import com.andrew.model.Game;
import com.andrew.model.Question;
import com.andrew.model.Team;
import com.google.gson.Gson;

/**
 * Servlet implementation class GameServlet
 */
@WebServlet("/Gamedo")
public class GameServlet extends HttpServlet {
	static Team aTeam = new Team();
	private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public GameServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		response.setContentType("application/json; charset=utf-8");
		response.setCharacterEncoding("UTF-8");
		DALManager mal = new DALManager();
		Game game = mal.loadGameInfo(mal.loadGame()); // loads game
		ResponseUtil rsp = new ResponseUtil();
		rsp.setCode(200);
		rsp.setData(game);
		rsp.setMessage("sucess.");

		Gson agson = new Gson();
		String jsonObj = agson.toJson(rsp);
		PrintWriter out = response.getWriter();
		out.print(jsonObj);
		out.flush();

	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
	 */
```

```
	protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		String team = request.getParameter("team");
		int game = Integer.parseInt(request.getParameter("code"));
		DALManager mal = new DALManager();
		if(mal.codecheck(game)) { // if gamecode is linked to an active
game, the team is entered into the game
			DALManager.saveTeamInfo(mal.TeamEnter(team, game), team);
			response.sendRedirect("ContestantPage.html");
		}
		else {
			response.sendRedirect("GameCodeError.html"); // else
redirected to error page
		}
	}

}
```

LoginServlet Class:

```
import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


/**

 * Servlet implementation class LoginServlet

 */

@WebServlet("/Logindo")

public class LoginServlet extends HttpServlet {

	private static final long serialVersionUID = 1L;


  /**

   * @see HttpServlet#HttpServlet()

   */
```

```
    public LoginServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String username = request.getParameter("user");
        String pwd = request.getParameter("password");
        if("interact".equals(username) && "god".equals(pwd)) {  // if username and password matches preset details
            response.sendRedirect("quiz.html");
        }else {
            response.sendRedirect("LoginPage.html");
        }
    }
}
```

QuestionServlet Class:

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.andrew.SRV.DALManager;
import com.andrew.SRV.QuestionManagement;
import com.andrew.TriviaAPP.ResponseUtil;
import com.andrew.model.Question;
import com.google.gson.Gson;

/**
 * Servlet implementation class QuestionServlet
 */
@WebServlet("/Questiondo")
public class QuestionServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public QuestionServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		response.setContentType("application/json; charset=utf-8");
		response.setCharacterEncoding("UTF-8");
		ResponseUtil rsp = new ResponseUtil();
		DALManager manager = new DALManager();
		String act = request.getParameter("act");
		if ("past".equalsIgnoreCase(act)) {
			List<Question> listQuestion =
manager.loadQuestionsFromPast();
			rsp.setCode(200); // HTTP status code - 200 = success
			rsp.setData(listQuestion); // Data in list put into
responseUtil
			rsp.setMessage("sucess.");
		}
		else {
```

```
                    List<Question> listQuestion = manager.loadQuestions(); //
returns list of all questions in SQLite dationabase
                    rsp.setCode(200); // HTTP status code - 200 = success
                    rsp.setData(listQuestion); // Data in list put into
responseUtil
                    rsp.setMessage("sucess.");
            }

            Gson agson = new Gson(); //creates new Gson object
            String jsonObj = agson.toJson(rsp); // changes Gson to Json
            PrintWriter out = response.getWriter();
            out.print(jsonObj);
            out.flush(); // printwriter gets sent
      }

      /**
       * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
       */
      protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            String act = request.getParameter("act");
            if ("add".equalsIgnoreCase(act)) { // add new question
                    int id =
Integer.parseInt(request.getParameter("questionOrder"));
                    double qID = Math.random() * (9999999 - 1000000) +
1000000;
                    String question = request.getParameter("description");
                    String choiceA = request.getParameter("choiceA");
                    String choiceB = request.getParameter("choiceB");
                    System.out.println("choiceB");
                    String choiceC = request.getParameter("choiceC");
                    String choiceD = request.getParameter("choiceD");
                    int timelimit =
Integer.parseInt(request.getParameter("time"));
                    char correctChoice =
request.getParameter("correctanswer").charAt(0);
                    String explaination =
request.getParameter("explaination");
                    QuestionManagement.createQuestion(qID, id, question,
choiceA, choiceB, choiceC, choiceD, timelimit, correctChoice, explaination);
            }else if("edit".equalsIgnoreCase(act)){ // edit existing question
                    int id =
Integer.parseInt(request.getParameter("questionOrder"));
                    double qID =
Double.parseDouble(request.getParameter("id"));
                    System.out.println(qID);
                    String question = request.getParameter("description");
                    String choiceA = request.getParameter("choiceA");
                    String choiceB = request.getParameter("choiceB");
                    String choiceC = request.getParameter("choiceC");
                    String choiceD = request.getParameter("choiceD");
                    int timelimit =
Integer.parseInt(request.getParameter("time"));
```

```
                    char correctChoice =
request.getParameter("correctanswer").charAt(0);
                    String explaination =
request.getParameter("explaination");
                    QuestionManagement.updateUser(qID, id, question, choiceA,
choiceB, choiceC, choiceD, timelimit, correctChoice, explaination);
             }else if("delete".equalsIgnoreCase(act)) { // delete existing
question
                    System.out.println(request.getParameter("questionOrder"));
                    double id =
Double.parseDouble(request.getParameter("questionOrder"));
                    QuestionManagement.deleteQuestion(id);
             }
             response.sendRedirect("Question.html"); // refreshes page
             //doGet(request, response);
       }

}
```

TeamServlet class:

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.andrew.SRV.DALManager;
import com.andrew.SRV.QuestionManagement;
import com.andrew.SRV.TeamManagement;
import com.andrew.TriviaAPP.ResponseUtil;
import com.andrew.model.Game;
import com.andrew.model.Question;
import com.andrew.model.Team;
import com.google.gson.Gson;

/**
 * Servlet implementation class TeamServlet
 */
@WebServlet("/loadTeam")
public class TeamServlet extends HttpServlet {
       private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public TeamServlet() {
        super();
```

```
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            response.setContentType("application/json; charset=utf-8");
            response.setCharacterEncoding("UTF-8");
            ResponseUtil rsp = new ResponseUtil();
            DALManager manager = new DALManager();
            ArrayList<ArrayList<Integer>> rank =
TeamManagement.displayRanking();
            ArrayList<String> teams=TeamManagement.returnNames(rank.get(0));
            ArrayList<ArrayList<Object>> ranking = new
ArrayList<ArrayList<Object>>(); // arraylist of object arraylitss to store
both String and int
            ArrayList<Object> team = new ArrayList<Object>();
            ArrayList<Object> points = new ArrayList<Object>();
            for (String i:teams) {
                  team.add(i);
            }
            for (int i:rank.get(1)) {
                  points.add(i);
            }
            ranking.add(team);
            ranking.add(points);

            rsp.setCode(200); // HTTP status code - 200 = success
            rsp.setData(ranking); // Data in list put into responseUtil
            rsp.setMessage("sucess.");

            Gson agson = new Gson();
            String jsonObj = agson.toJson(rsp);
            PrintWriter out = response.getWriter();
            out.print(jsonObj);
            out.flush();
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            DALManager.addPoints(); // adds points
    }

}
```

ValidationServlet Class:

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.andrew.SRV.DALManager;
import com.andrew.SRV.QuestionManagement;

/**
 * Servlet implementation class ValidationServlet
 */
@WebServlet("/Validation")
public class ValidationServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ValidationServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

	/**
```

```
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		// TODO Auto-generated method stub
		PrintWriter pw=response.getWriter();
		if (!QuestionManagement.validateGame()) {
			System.out.println("vad fail");
			pw.print("errorQuestion.html");
			pw.flush();
		}
		else {
			System.out.println("vad suc");
			pw.print("Game.html");
			pw.flush();
		}
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		DALManager.endGame();
	}
}
```

addq.html:

```
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="Questiondo" method="post">
	<input type="text" name="questionID" placeholder="ID"></input>
	<input type="text" name="description" placeholder="description"></input>
	<input type="text" name="choiceA" placeholder="choice A"></input>
	<input type="text" name="choiceA" placeholder="Choice B"></input>
	<input type="text" name="choiceC" placeholder="Choice C"></input>
	<input type="text" name="choiceD" placeholder="Choice D"></input>
	<input type="text" name="time" placeholder="time"></input>
	<input type="text" name="correctanswer" placeholder="correct
Answer"></input>
	<input type="text" name="explaination"
placeholder="explaination"></input>
	<button>add</button>
</form>
</body>
</html>
```

admin.html:

```
<!DOCTYPE html>
<html>
<style type="text/css">
 html,
body {
	height: 100%;
	color: white;
}
body {
	margin: 0;
	background: #1A1A1D;
	font-family: sans-serif;
	font-weight: 100;
	text-align: center;
}

h1{
	font-size:50px;
}

#current_num{
	font-size:40px;

}
table {
	width: 1100px;
	border-collapse: collapse;
	overflow: hidden;
```

```
        box-shadow: 0 0 20px rgba(0,0,0,0.1);

}

th {
        padding: 15px;
        background-color: #C3073F;
        color: #fff;
}

td{
        padding: 15px;
        background-color: #6F2232;
        color: #fff;
}

th {
        text-align: left;
}

thead {
        th {
                background-color: #55608f;
        }
}

 input[type=submit]{
        background-color: #950740;
        border: none;
        color: white;
        text-align: center;
        text-decoration: none;
        font-size: 25px;
        border: 1px solid black;
}

input[type=submit]:hover{
        transition-duration: 0.2s;
        background-color: #C3073F;
        color: white;
}
</style>
<head>
<meta charset="ISO-8859-1">
<title>Admin</title>
<script src="jquery-3.5.1.js"></script>
<script type="text/javascript">

function getCurrentQuestion(){
        $.get("gameholder", function(result, status){
                console.log(result)
                console.log(result.questionID)
                $("#current_num").text(result.question)
        });
}
```

```
function display_c(){
	var refresh=1000; // Refresh rate in milli seconds
	mytime=setTimeout('display_ct()',refresh)
	}

function display_ct() {
	var x = new Date()
	document.getElementById('ct').innerHTML = x;
	display_c();
	 }

	function loadTeamRanking(){
		$.get("loadTeam", function(result, status){
			var q=1
			console.log(result.data)

			var  team = result.data[0]
			var score = result.data[1]
			for(i=team.length-1;i>=0;i--)
			{
				console.log(result.data[i])

	$('#tbl').append("<tr><td>"+q+"</td><td>"+team[i]+"</td><td>"+score[i]+"
</td></tr>");
				q++
			}
		});
	}
</script>
</head>
<body  onload="loadTeamRanking();getCurrentQuestion(); display_ct();">
	<form action="Admindo" method="get">
		<input type="submit" name="action" value="Next Question">
		<input type="submit" name="action" value="Show Explaination">
		<input type="submit" name="action" value="End Game">
	</form>
	<h1>Current Question:</h1>
	<div>
		<span  id="current_num">No question currently being
displayed</span>
	</div>
	<br><br><br><br>
	<span id='ct' ></span>
	<br><br><br><br><br><br>
	<div id="userListDiv">
		<table id="tbl" border="1" align="center">
		<thead>
		<th>Ranking</th>
		<th>Team</th>
		<th>Points</th>
		</thead>
		</table>
	</div>
</body>
</html>
```

AdminLogin.html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Admin Login</title>
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<form action="Admindo" method="post">
       <div class="center">
       <input type="text" name="user" id= "verticalspacelogin" class="color1
stack verticalspacecenter verticalspace"  placeholder="username"/>
        <input type="password" name="password" id= "password" class="color2
stack verticalspace" placeholder="password"/>
        <input type="text" name = "code" id="code" class="color2 stack
verticalspace" placeholder="gamecode"/>
        <button id="loginButton">Login</button>
    </div>
</form>
</body>
</html>
```

Code.html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Join Game</title>
</head>
<body>
       <form>
              <input type="text" name="questionID" placeholder="Enter
game"></input>
              <button>enter</button>
       </form>
</body>
</html>
```

comp.html:

```
<!DOCTYPE html>
<html>
<style type="text/css">
 html,
body {
       height: 100%;
       color: white;
}
body {
       margin: 0;
       background: #1A1A1D;
       font-family: sans-serif;
       font-weight: 100;
```

```
        text-align: center;
}

h1{
        font-size:50px;
}

h2{
        font-size:30px;
}

span{
        font-size: 40px;
}
button{
        padding: 60px 170px;
}

.divider{
    width:40px;
    height:auto;
    display:inline-block;
}
#a{
        background-color:#C3073F;
}
#b{
        background-color:#6F2232;
}
#c{
        background-color:#950740;
}
#d{
        background-color:#FF69B4;
}
#a:hover{
        background-color:Orange;
}
#b:hover{
        background-color:Orange;
}
#c:hover{
        background-color:Orange;
}
#d:hover{
        background-color:Orange;
}
#a[disabled]:hover {
  background-color:#C3073F;
}
#b[disabled]:hover{
        background-color:#6F2232;
}
#c[disabled]:hover{
        background-color:#950740;
```

```
}
#d[disabled]:hover{
	background-color:#FF69B4;
}
</style>
<script src="jquery-3.5.1.js"></script>
<script type="text/javascript">
var seconds;
var q=0;
var intervalID
	function checkQuestion(){
		setInterval("getCurrentQuestion()",1000)

	}

	function loadTeam(){
		$.get("loadTeam",function(result,status){
			console.log(result)
		});
	}

	function getCurrentQuestion(){
		$.get("gameholder", function(result, status){
			console.log(result)
			console.log(result.questionID)
			if (result.questionID==0){
				window.location.replace("end.html")
			}
			if(q!=result.questionID){
				seconds=result.timelimit
				q++
				test()
				$(":button").attr("disabled", false);
			}
			$("#current_num").text(result.question)
			$("#a").text(result.answerchoiceA)
			$("#b").text(result.answerchoiceB)
			$("#c").text(result.answerchoiceC)
			$("#d").text(result.answerchoiceD)
			if(result.showExplaination==1){
				$("#explain").text(result.explaination)
			}
			else{
				$("#explain").text("")
			}
		});
	}

	function addPoints(){
		console.log("add");
		$.post("gameholder");
	}

		function myMain(id) {
			$.get("gameholder", function(result, status){
```

```
                        console.log(id);
                          if (id==result.correctAnswer){
                                addPoints();
                          }
                        });
                }


             function Fuction() {
                      document.getElementById('answer').disabled = 'disabled';
                    }

             function disbaleOptions(){
            alert("your option is submited");
            $(":button").attr("disabled", true);
        }

             function countdown() {
            console.log(seconds);
            $("#timer").html(seconds);
            if (seconds == 0) {
                $(":button").attr("disabled", true);
                clearInterval(intervalID);
                return;
            }
            seconds = seconds - 1;

        };

             function test() {
            intervalID = setInterval(countdown, 1000)
        }
</script>
<head>
<meta charset="ISO-8859-1">
<title>welcome</title>
</head>
<body onload="checkQuestion();loadTeam();">
       <div>
              <br><br>
              <span id="timer"></span><br>
              <br><br><br><br>
              <h1 id="current_num"></h1><br><br><br><br>
              <button id="a" onclick="myMain(this.id);
disbaleOptions()"></button>
              <div class="divider"></div>
              <button id="b" onclick="myMain(this.id);
disbaleOptions()"></button><br><br><br><br>
              <button id="c" onclick="myMain(this.id);
disbaleOptions()"></button>
              <div class="divider"></div>
              <button id="d" onclick="myMain(this.id);
disbaleOptions()"></button>
              <br>
              <h2 id="explain"></h2>
```

```
	</div>
</body>
</html>
```

ContestantPage.html:

```
<!DOCTYPE html>
<html>
<style type="text/css">
 html,
body {
	height: 100%;
	color: white;
}
body {
	margin: 0;
	background: #1A1A1D;
	font-family: sans-serif;
	font-weight: 100;
	text-align: center;
}

h1{
	font-size:50px;
}
</style>
<script src="jquery-3.5.1.js"></script>
<script type="text/javascript">
function checkQuestion(){
	setInterval("getCurrentQuestion()",2000)

}
	function getCurrentQuestion(){
		$.get("gameholder", function(result, status){
			console.log(result)
			console.log(result.questionID)
			$("#current_num").text(result.description)
			if(result.questionID ==1){
				window.location.assign("comp.html")
			}
		});
	}
</script>
<head>
<meta charset="ISO-8859-1">
<title>welcome</title>
</head>
<body onload="checkQuestion()">
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<h1>Welcome! You are in game!</h1>
<div id="current_num"></div>
</body>
</html>
```

end.html:

```
<!DOCTYPE html>
<html>
<style type="text/css">
 html,
body {
	height: 100%;
	color: white;
}
body {
	margin: 0;
	background: #1A1A1D;
	font-family: sans-serif;
	font-weight: 100;
	text-align: center;
}

h1{
	font-size:50px;
}
</style>
<head>
<meta charset="ISO-8859-1">
<title>Game ended</title>
</head>
<body>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<h1>Thats it folks! Thank you all for attending years quiz! The winners will
be annouced shortly!</h1>
</body>
</html>
```

errorQuestion.html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
<style type="text/css">
 html,
body {
	height: 100%;
	color: white;
}

body {
	margin: 0;
	background: #1A1A1D;
	font-family: sans-serif;
	font-weight: 100;
}

.container {
	position: absolute;
	top: 50%;
```

```
        left: 50%;
        transform: translate(-50%, -50%);
}

table {
        width: 1100px;
        border-collapse: collapse;
        overflow: hidden;
        box-shadow: 0 0 20px rgba(0,0,0,0.1);
}

th {
        padding: 15px;
        background-color: #C3073F;
        color: #fff;
}

td{
        padding: 15px;
        background-color: #6F2232;
        color: #fff;
}

th {
        text-align: left;
}

thead {
        th {
                background-color: #55608f;
        }
}

 button{
        background-color: #950740;
        border: none;
        color: white;
        text-align: center;
        text-decoration: none;
        font-size: 16px;
        border: 1px solid black;
}

button:hover{
        transition-duration: 0.2s;
        background-color: #C3073F;
        color: white;
}

dialog{
        background-color: #C3073F;
}

input {
        color: white;
```

```
	background-color:#950740;
}
 </style>
<script src="jquery-3.5.1.js"></script>
<script type="text/javascript">
	function loadQuestionList() {
		$.get("Questiondo", function(result, status){
			var q
			console.log(result.data)
			for(i=0; i<result.data.length; i++)
			{
				console.log(result.data[i].question)

	$('#tbl').append("<tr><td>"+result.data[i].questionOrder+"</td><td>"+res
ult.data[i].question+"</td><td>"+result.data[i].answerchoiceA+"</td><td>"+resu
lt.data[i].answerchoiceB+"</td><td>"+result.data[i].answerchoiceC+"</td><td>"+
result.data[i].answerchoiceD+"</td><td>"+result.data[i].timelimit+"</td><td>"+
result.data[i].correctAnswer+"</td><td>"+result.data[i].explaination+
"</td><td><button
onclick=\"showEditDlg('edit','"+result.data[i].questionOrder+"','"+result.data
[i].question+"','"+result.data[i].answerchoiceA+"','"+result.data[i].answercho
iceB+"','"+result.data[i].answerchoiceC+"','"+result.data[i].answerchoiceD+"',
'"+result.data[i].timelimit+"','"+result.data[i].correctAnswer+"','"+result.da
ta[i].explaination+"','"+result.data[i].questionID+"')\">Edit</button>"+"<butt
on
onclick=\"deleteQuestion('"+result.data[i].questionID+"')\">X</button>"+"</td>
</tr>");
			}
		});
	}

	function showDlg(act) {
		document.getElementById("act").value = act;
		document.getElementById("dlg").open = true;
	}

	function showEditDlg(act, id, question, choiceA, choiceB, choiceC,
choiceD, time, correctAns,explaination,qID) {
		document.getElementById("act").value=act;
		document.getElementById("questionOrder").value = id;
		document.getElementById("description").value=question;
		document.getElementById("choiceA").value=choiceA;
		document.getElementById("choiceB").value=choiceB;
		document.getElementById("choiceC").value=choiceC;
		document.getElementById("choiceD").value=choiceD;
		document.getElementById("time").value=time;
		document.getElementById("correct").value=correctAns;
		document.getElementById("explain").value=explaination;
		document.getElementById("questionID").value=qID;
		document.getElementById("dlg").open = true;
	}

	function saveQuestion() {
		$.post("Questiondo",
			$("#fm").serialize(),
```

```
                        function(data,status) {
                                document.getElementById("dlg").open = false;
                                loadQuestionList();
                        }
                );
        }

        function deleteQuestion(QuestionID) {
                if(confirm("Are you sure you want to remove this question?")) {
                        $.post("Questiondo",
                                        {"questionOrder":QuestionID,
                                "act":"delete"},
                        );
                }
                location.reload();
        }

        function goBack() {
                window.location.assign("Welcome.html")
        }

</script>
</head>
<body onload="loadQuestionList()">
        <button onclick="goBack()">Go Back</button>
        <h1>Make sure that order is valid: no repeats and in order</h1>
        <div id="userListDiv">
                <table id="tbl" border="1">
                <thead>
                <th>Order</th>
                <th>Question</th>
                <th>Choice A</th>
                <th>Choice B</th>
                <th>Choice C</th>
                <th>Choice D</th>
                <th>Time Limit</th>
                <th>Correct Answer</th>
                <th>Explaination</th>
                <th></th>
                </thead>
                </table>
        </div>

        <dialog id="dlg">
                <form id="fm">
                        <input type="text" id="act" name="act"
readonly="readonly"/> <br>
                        <input type="text" id= "questionOrder"
name="questionOrder" placeholder="Order"></input> <br>
                        <input type="text" id="description" name="description"
placeholder="description"></input><br>
                        <input type="text" id="choiceA" name="choiceA"
placeholder="choice A"></input><br>
                        <input type="text" id="choiceB" name="choiceB"
placeholder="Choice B"></input><br>
```

```
                    <input type="text" id="choiceC" name="choiceC"
placeholder="Choice C"></input><br>
                    <input type="text" id="choiceD" name="choiceD"
placeholder="Choice D"></input><br>
                    <input type="text" id="time" name="time"
placeholder="time"></input><br>
                    <input type="text" id="correct" name="correctanswer"
placeholder="correct Answer"></input><br>
                    <input type="text" id="explain" name="explaination"
placeholder="explaination"></input><br>
                    <input type="text" id="questionID" name="id"
placeholder="ID" readonly="readonly"></input><br>
                    <button onclick="saveQuestion()">save</button>
             </form>
      </dialog>
</body>
</html>
```

Game.html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Game Code</title>
<style type="text/css">
 html,
body {
      height: 100%;
      color: white;
}

body {
      margin: 0;
      background: #1A1A1D;
      font-family: sans-serif;
      font-weight: 100;
}

.container {
      position: absolute;
      top: 50%;
      left: 50%;
      transform: translate(-50%, -50%);
}

table {
      width: 1100px;
      border-collapse: collapse;
      overflow: hidden;
      box-shadow: 0 0 20px rgba(0,0,0,0.1);
}

th {
      padding: 15px;
```

```
        background-color: #C3073F;
        color: #fff;
}

td{
        padding: 15px;
        background-color: #6F2232;
        color: #fff;
}

th {
        text-align: left;
}

h1{
        text-align: center;
}
</style>
<script src="jquery-3.5.1.js"></script>
<script type="text/javascript">
        function loadGameInfo(){
                $.get("Gamedo", function(result, status){
                        console.log(result.data)

        $('#tbl').append("<tr><td>"+result.data.gameName+"</td><td>"+result.data
.gameID+"</td></tr>")
                });
        }
        function checkQuestion(){
                setInterval("getCurrentQuestion()",2000)

        }

                function getCurrentQuestion(){
                        $.get("gameholder", function(result, status){
                                console.log(result)
                                console.log(result.questionID)
                                $("#current_num").text(result.description)
                                if(result.questionID ==1){
                                        window.location.assign("host.html")
                                }
                        });
                }

</script>
</head>
<body onload="loadGameInfo(); checkQuestion();">
<br><br><br>
<table id="tbl" border="1" style="border:1px solid black;margin-
left:auto;margin-right:auto;">
        <thead>
        <th>Game</th>
        <th>Code</th>
        </thead>
        </table>
```

```
        <br><br><br>
        <h1>WELCOME TO TRIVIA NIGHT!</h1>
</body>
</html>
```

GameCode.html:

```
<!DOCTYPE html>
<html>
    <link rel="stylesheet" type="text/css" href="style.css">
    <title>Enter Game Code</title>
    <body>
       <form action="Gamedo" method="post">
         <div class="center">
             <input type="text" name="code" id= "verticalspacelogin"
placeholder="Enter Game Code" class="color1 stack verticalspace2"/>
                   <input type="text" name="team" id= "team" class="color1
stack verticalspace2" placeholder="Enter Team Name"/>
             <button id="loginButton">Enter</button>
         </div>
       </form>
    </body>
    <script src="jquery-3.5.1.js"></script>
    <script>
    </script>
</html>
```

host.html:

```
<!DOCTYPE html>
<html>
<style type="text/css">
 html,
body {
       height: 100%;
       color: white;
}
body {
       margin: 0;
       background: #1A1A1D;
       font-family: sans-serif;
       font-weight: 100;
       text-align: center;
}

h1{
       font-size:50px;
}

span{
       font-size: 40px;
}
</style>
<script src="jquery-3.5.1.js"></script>
<script type="text/javascript">
```

```
	function checkQuestion(){
		setInterval("getCurrentQuestion()",1000)

	}

	function getCurrentQuestion(){
		$.get("gameholder", function(result, status){
			console.log(result)
			console.log(result.questionID)
			if (result.questionID==0){
				window.location.replace("results.html");
			}
			$("#current_num").text(result.question)
		});
	}
</script>
<head>
<meta charset="ISO-8859-1">
<title>Host</title>
</head>
<body onload="checkQuestion();">
	<br><br><br><br><br><br><br><br>
	<h1>Question:</h1>
	<div>
		<br><br>
		<span  id="current_num"></span>
	</div>
</body>
</html>
```

index.html:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Trivia Night</title>
        <link rel="stylesheet" type="text/css" href="style.css">
        <body>
            <h1 id="title"><b>TRIVIA NIGHT</b></h1>
            <br>
            <button type="button" id="button1" class="button color2 stack
titlepagebutton verticalspace">Join Game</button>
            <button type="button" id="button2" class="button color1 stack
titlepagebutton verticalspace">Admin</button>
            <button type="button" id="button3" class="button color3 stack
titlepagebutton">Login</button>
        </body>
        <script src="jquery-3.5.1.js"></script>
        <script>
            $('#button1').click(function()
            {
                window.location.replace('GameCode.html');
            });
```

```
            $('#button2').click(function()
            {
             window.location.replace('AdminLogin.html');
            });
            $('#button3').click(function()
            {
                window.location.replace('LoginPage.html');
            });
        </script>
    </head>
</html>
```

Leaderboard.html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Leaderboard</title>
<style type="text/css">
 html,
body {
	height: 100%;
}

body {
	margin: 0;
	background: #1A1A1D;
	font-family: sans-serif;
	font-weight: 100;
	color: white;
	text-align: center;
}

.container {
	position: absolute;
	top: 50%;
	left: 50%;
	transform: translate(-50%, -50%);
}

table {
	width: 1100px;
	border-collapse: collapse;
	overflow: hidden;
	box-shadow: 0 0 20px rgba(0,0,0,0.1);
}

th {
	padding: 15px;
	background-color: #C3073F;
	color: #fff;
}

td{
```

```
        padding: 15px;
        background-color: #6F2232;
        color: #fff;
}

th {
        text-align: left;
}

thead {
        th {
                background-color: #55608f;
        }
}

 button{
        background-color: #950740;
        border: none;
        color: white;
        text-align: center;
        text-decoration: none;
        font-size: 16px;
        border: 1px solid black;
        padding: 6px 9px;
}

button:hover{
        transition-duration: 0.2s;
        background-color: #C3073F;
        color: white;
}

dialog{
        background-color: #C3073F;
}

input {
        color: white;
        background-color:#950740;
}
 </style>
<script src="jquery-3.5.1.js"></script>
<script type="text/javascript">
        function loadTeamRanking(){
                $.get("loadTeam", function(result, status){
                        var q=1
                        console.log(result.data)
                        var  team = result.data[0]
                        var score = result.data[1]
                        for(i=team.length-1;i>=0;i--)
                        {
                                console.log(result.data[i])

        $('#tbl').append("<tr><td>"+q+"</td><td>"+team[i]+"</td><td>"+score[i]+"
</td></tr>");
```

```
                        q++
                    }
                    $.post("Validation")
                });
        }
</script>
</head>
<body onload="loadTeamRanking()">
<br><br><br>
        <h1>Final Rankings:</h1>
        <br><br><br><br><br><br><br><br><br><br>
        <div id="userListDiv">
                <table id="tbl" border="1" align="center">
                <thead>
                <th>Ranking</th>
                <th>Team</th>
                <th>Points</th>
                </thead>
                </table>
        </div>
</body>
</html>
```

LoginPage.html:

```
<!DOCTYPE html>
    <html>
        <link rel="stylesheet" type="text/css" href="style.css">
        <title>Login</title>
        <body>
             <form action="Logindo" method="post">
            <div class="center">
                    <input type="text" name="user" id= "verticalspacelogin"
class="color1 stack verticalspacecenter verticalspace"  placeholder="input
your username"/>
                    <input type="password" name="password" id= "password"
class="color2 stack verticalspace" placeholder="input your password"/>
                    <button id="loginButton">Login</button>
            </div>
             </form>
        </body>
    </html>
```

pastyearsquestions.html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>HUB</title>
<style type="text/css">
 html,
body {
       height: 100%;
```

```
}

body {
	margin: 0;
	background: #1A1A1D;
	font-family: sans-serif;
	font-weight: 100;
}

.container {
	position: absolute;
	top: 50%;
	left: 50%;
	transform: translate(-50%, -50%);
}

table {
	width: 1100px;
	border-collapse: collapse;
	overflow: hidden;
	box-shadow: 0 0 20px rgba(0,0,0,0.1);
}

th {
	padding: 15px;
	background-color: #C3073F;
	color: #fff;
}

td{
	padding: 15px;
	background-color: #6F2232;
	color: #fff;
}

th {
	text-align: left;
}

thead {
	th {
		background-color: #55608f;
	}
}

 button{
	background-color: #950740;
	border: none;
	color: white;
	text-align: center;
	text-decoration: none;
	font-size: 16px;
	border: 1px solid black;
	padding: 6px 9px;
}
```

```
button:hover{
	transition-duration: 0.2s;
	background-color: #C3073F;
	color: white;
}

dialog{
	background-color: #C3073F;
}

input {
	color: white;
	background-color:#950740;
}
 </style>
<script src="jquery-3.5.1.js"></script>
<script type="text/javascript">
	function loadQuestionList() {
		$.get("Questiondo",{
			"act":"past"}, function(result, status){
			var q
			console.log(result.data)
			for(i=0; i<result.data.length; i++)
			{
				console.log(result.data[i].question)

	$('#tbl').append("<tr><td>"+result.data[i].questionID+"</td><td>"+result
.data[i].question+"</td><td>"+result.data[i].answerchoiceA+"</td><td>"+result.
data[i].answerchoiceB+"</td><td>"+result.data[i].answerchoiceC+"</td><td>"+res
ult.data[i].answerchoiceD+"</td><td>"+result.data[i].timelimit+"</td><td>"+res
ult.data[i].correctAnswer+"</td><td>"+result.data[i].explaination+"</td></tr>"
);
			}
		});
	}


	function goBack() {
		window.location.assign("quiz.html")
	}

</script>
</head>
<body onload="loadQuestionList()">
	<button onclick="goBack()">Go Back</button>
	<div id="userListDiv">
		<table id="tbl" border="1" align="center">
		<thead>
		<th>Year</th>
		<th>Question</th>
		<th>Choice A</th>
		<th>Choice B</th>
		<th>Choice C</th>
		<th>Choice D</th>
```

```
            <th>Time Limit</th>
            <th>Correct Answer</th>
            <th>Explanation</th>s
            </thead>
            </table>
      </div>
</body>
</html>
</body>
</html>
```

Question.html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>HUB</title>
 <style type="text/css">
 html,
body {
      height: 100%;
}

body {
      margin: 0;
      background: #1A1A1D;
      font-family: sans-serif;
      font-weight: 100;
}

.container {
      position: absolute;
      top: 50%;
      left: 50%;
      transform: translate(-50%, -50%);
}

table {
      width: 1100px;
      border-collapse: collapse;
      overflow: hidden;
      box-shadow: 0 0 20px rgba(0,0,0,0.1);
}

th {
      padding: 15px;
      background-color: #C3073F;
      color: #fff;
}

td{
      padding: 15px;
      background-color: #6F2232;
      color: #fff;
```

```
}

th {
	text-align: left;
}

thead {
	th {
		background-color: #55608f;
	}
}

 button{
	background-color: #950740;
	border: none;
	color: white;
	text-align: center;
	text-decoration: none;
	font-size: 16px;
	border: 1px solid black;
	padding: 6px 9px;
}

button:hover{
	transition-duration: 0.2s;
	background-color: #C3073F;
	color: white;
}

dialog{
	background-color: #C3073F;
}

input {
	color: white;
	background-color:#950740;
}
 </style>
<script src="jquery-3.5.1.js"></script>
<script type="text/javascript">
	function loadQuestionList() {
		$.get("Questiondo", function(result, status){
			var q
			console.log(result.data)
			for(i=0; i<result.data.length; i++)
			{
				console.log(result.data[i].question)

	$('#tbl').append("<tr><td>"+result.data[i].questionOrder+"</td><td>"+res
ult.data[i].question+"</td><td>"+result.data[i].answerchoiceA+"</td><td>"+resu
lt.data[i].answerchoiceB+"</td><td>"+result.data[i].answerchoiceC+"</td><td>"+
result.data[i].answerchoiceD+"</td><td>"+result.data[i].timelimit+"</td><td>"+
result.data[i].correctAnswer+"</td><td>"+result.data[i].explaination+
                                        "</td><td><button
onclick=\"showEditDlg('edit','"+result.data[i].questionOrder+"','"+result.data
```

```
[i].question+"','"+result.data[i].answerchoiceA+"','"+result.data[i].answercho
iceB+"','"+result.data[i].answerchoiceC+"','"+result.data[i].answerchoiceD+"',
'"+result.data[i].timelimit+"','"+result.data[i].correctAnswer+"','"+result.da
ta[i].explaination+"','"+result.data[i].questionID+"')\">Edit</button>"+"<butt
on
onclick=\"deleteQuestion('"+result.data[i].questionID+"')\">X</button>"+"</td>
</tr>");
                        }
                });
        }

        function showDlg(act) {
                document.getElementById("act").value = act;
                document.getElementById("dlg").open = true;
        }

        function showEditDlg(act, id, question, choiceA, choiceB, choiceC,
choiceD, time, correctAns,explaination,qID) {
                document.getElementById("act").value=act;
                document.getElementById("questionOrder").value = id;
                document.getElementById("description").value=question;
                document.getElementById("choiceA").value=choiceA;
                document.getElementById("choiceB").value=choiceB;
                document.getElementById("choiceC").value=choiceC;
                document.getElementById("choiceD").value=choiceD;
                document.getElementById("time").value=time;
                document.getElementById("correct").value=correctAns;
                document.getElementById("explain").value=explaination;
                document.getElementById("questionID").value=qID;
                document.getElementById("dlg").open = true;
        }

        function saveQuestion() {
                $.post("Questiondo",
                        $("#fm").serialize(),
                        function(data,status) {
                                document.getElementById("dlg").open = false;
                                loadQuestionList();
                        }
                );
        }

        function deleteQuestion(QuestionID) {
                if(confirm("Are you sure you want to remove this question?")) {
                        $.post("Questiondo",
                                        {"questionOrder":QuestionID,
                                "act":"delete"},
                        );
                }
                location.reload();
        }

        function goBack() {
                window.location.assign("Welcome.html")
        }
```

```
</script>
</head>
<body onload="loadQuestionList()">
	<br><br><br>
	<button onclick="goBack()">Go Back</button>
	<button onclick="showDlg('add')">Add question</button>
	<br><br><br>
	<div id="userListDiv">
		<table id="tbl" border="1" align="center">
		<thead>
		<th>Order</th>
		<th>Question</th>
		<th>Choice A</th>
		<th>Choice B</th>
		<th>Choice C</th>
		<th>Choice D</th>
		<th>Time Limit</th>
		<th>Correct Answer</th>
		<th>Explaination</th>
		<th></th>
		</thead>
		</table>
	</div>

	<dialog id="dlg">
		<form id="fm">
			<input type="text" id="act" name="act"
readonly="readonly"/> <br>
			<input type="text" id= "questionOrder"
name="questionOrder" placeholder="Order"></input> <br>
			<input type="text" id="description" name="description"
placeholder="description"></input><br>
			<input type="text" id="choiceA" name="choiceA"
placeholder="choice A"></input><br>
			<input type="text" id="choiceB" name="choiceB"
placeholder="Choice B"></input><br>
			<input type="text" id="choiceC" name="choiceC"
placeholder="Choice C"></input><br>
			<input type="text" id="choiceD" name="choiceD"
placeholder="Choice D"></input><br>
			<input type="text" id="time" name="time"
placeholder="time"></input><br>
			<input type="text" id="correct" name="correctanswer"
placeholder="correct Answer"></input><br>
			<input type="text" id="explain" name="explaination"
placeholder="explaination"></input><br>
			<input type="text" id="questionID" name="id"
placeholder="ID" readonly="readonly"></input><br>
			<button onclick="saveQuestion()">save</button>
		</form>
	</dialog>
</body>
</html>
```

quiz.html:

```
<!DOCTYPE html>
<html>
    <title>Welcome</title>
    <script src="jquery-3.5.1.js"></script>
      <script type="text/javascript">
      function hostGame() {
            window.location.replace('pastyearsquestions.html')
      }
      </script>
    <link rel="stylesheet" type="text/css" href="style.css">
    <body>
        <div class="center">
            <button type="button" id="verticalspacewelcome" class="color2
welcomepagebutton horizontalspace">This years quiz</button>
            <button type="button" onclick="hostGame()" id="hostgame"
class="color3 welcomepagebutton">View past years questions</button>
        </div>
    </body>
    <script>
      $('#verticalspacewelcome').click(function()
       {
            window.location.replace('Welcome.html');
       });
    </script>
</html>
```

result.html:

```
<!DOCTYPE html>
<html>
<style type="text/css">
 html,
body {
      height: 100%;
      color: white;
}
body {
      margin: 0;
      background: #1A1A1D;
      font-family: sans-serif;
      font-weight: 100;
      text-align: center;
}

h1{
      font-size:50px;
}
</style>
<head>
```

```
<meta charset="ISO-8859-1">
<title>Results</title>
</head>
<body>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<h1>Thats it folks! Thank you all for attending years quiz! The winners
are...</h1>
</body>
</html>
```

style.css:

```
button {
  transition-duration: 0.4s;
  text-align: center;
}

button
{
    text-align: center;
    border: #1a1a1d;
}

button:hover {
  background-color: orchid;
}

.color1
{
    background-color: #6f2232
}

.color2
{
    background-color: #950740;
}

.color3
{
    background-color: #c3073f;
}

.stack
{
    display: block;
    margin: auto;

}

.sameline
{
    display: inline;
}

body {
```

```
    background-color: #1a1a1d;
}

#title {
    text-align: center;
    font-weight: bold;
    font-size: 6rem;
    color: white;
}

.titlepagebutton
{
    font-size: 2rem;
    width:27rem;
    height:9rem;
}

.verticalspace {
    margin-bottom:55px
}

.verticalspace2 {
    margin-bottom: 45px;
}

#verticalspacelogin {
    margin-top: 350px;
}

.horizontalspace
{
    margin-right:80px;
}

input {
    color: white;
    border:10px #1a1a1d;
    height: 30px;
    width: 200px;

}

#loginContainer
{
    height: 400px;
    position: relative;
    text-align: center;
    border:3px solid green;
}

#loginButton
{
    height: 30px;
    width: 80px;
}
```

```
.center {
   text-align: center;
  }

#verticalspacewelcome {
    margin-top: 400px;
}

.welcomepagebutton
{
    height: 70px;
    width: 250px;
}
```

Welcome.html:

```
<!DOCTYPE html>
<html>
    <title>Welcome</title>
    <script src="jquery-3.5.1.js"></script>
      <script type="text/javascript">
      function hostGame() {
            if(confirm("Do you want to start a game?")) {
                  $.get("Validation",function(data, status){
                        console.log(data);
                        window.location.replace(data);
                  });
            }
      }
      </script>
    <link rel="stylesheet" type="text/css" href="style.css">
    <body>
        <div class="center">
            <button type="button" id="verticalspacewelcome" class="color2
welcomepagebutton horizontalspace">Edit Questions</button>
            <button type="button" onclick="hostGame()" id="hostgame"
class="color3 welcomepagebutton">Host Game</button>
        </div>
    </body>
    <script>
      $('#verticalspacewelcome').click(function()
        {
            window.location.replace('Question.html');
        });
    </script>
</html>
```