**Maintenance Guide -** Cloud Computing City

**DB structure:**

**questions:**  
Id – contain the question's id  
question – contain the question contact.  
Option1, Option2, Option3, Option4 – contain the four answers.  
Correct – contain the correct option.  
Level – contains the difficulty level of the question.

**users:**  
Admin – contain 'yes' or 'no'. 'yes' for user with admin permission.  
Username – contain the user's username.  
Password – contain the user's password.  
CurrentQuestionsInARow – contain the maximum question that player answered correctly in a row.  
**Games**: save data on the sessions.  
(Note: for every user there is one game that contain session: -1 and sessionScore: -1 as default)  
 session – contain the session number.  
 sessionScore – contain the score that player got in the session.  
Score – contain the user's score.  
QuestionsThatBeenAnsweredCorrect – contain all the questions id that the user answered correct.  
userPlace – contain the user place compere the other users. (Note: this field added to the user data when he clicks on the Leader Table button)

**videos:**  
imgVideo – contain the image that will be display for that video in the Cinema page.  
title – contain the title that will be display for that video in the Cinema page.  
urlVideo – contain the URL video.

**DB variables:**

**res** - contain all user data from the db.  
**resQuestion** - contain all questions data from the db.  
**resVideo** - contain all videos data from the db.  
**easyQuestion** - contain all questions data with level easy from the db.  
**easyAndMediumQuestion** - contain all questions data with level easy and medium from the db.

**Global variables:**

**userDetails** – save the details on the user that current logged in  
**countCorrectAnswer** – count the number of correct answer in session  
**countWorngAnswer** – count the number of worng answer in session  
**questionNumberForGame** – the number of questions that user choose to play in session  
**countQuestionThatUserAnswerInGame** – count the number of questions that user answered in session  
**firstVideoOnPage** – save the index of the first video in the Cinema page  
**questionId** – contain the current id question that is displayed  
**qestionLevel** – contain the current level question that is displayed  
**additionalScore** – save the score that user earn in session  
**CurrentQuestionsInARow** – count the current number of questions that user answered correct in a row  
**maxQuestionsInARow** – contain the maximum questions that user answered correct in a seesion  
**start\_time** – contain the time that user start play the session

**Score variables:**

scoreForEasyQuestion = 50  
scoreForMediumQuestion = 100  
scoreForHardQuestion = 150  
scoreFor3CorrectAnswerInARow = 30

**Functions:**

**def signUp():**  
input - none  
output - none  
do - display the Signup page.

**def addQuestionBtnAdminPage():**  
input - none  
output - none  
do - display the Add Question page for admin.

**def editQuestionBtnAdminPage():**  
input - none  
output - none  
do - display the Add/Delete Question page for admin.

**def addVideoBtnAdminPage ():**  
input - none  
output - none  
do - display the Add Video page for admin.

**def adminPage():**  
input - none  
output - none  
do - display the Admin home page.

**def logout():**  
input - none  
output - none  
do - display the Login page.

**def homePage():**  
input - none  
output - none  
do - display the Home page and call loadUserScoreAndBadge function.

**def walcomePage():**  
input - none  
output - none  
do - display the Welcome page.

**def loadUserScoreAndBadge():**  
input - none   
output - none  
do - load the user information and badge (if has).

**def cinemaPage():**  
input - none  
output - none  
do - display the Cinema page and call loadVideoForPage function.

**def updateUserDataAfterPlaying():**  
input - none  
output - none  
do - update user data after playing (update res array, userDetails and db).

**def resultPage():**  
input - none  
output - none  
do - calculate the time that user played and display it, call loadChartPie and loadBarChart function and display the result game.

**def loadChartPie():**  
input - none  
output - none  
do - create the chart pie that contain the number of questions that user answered correct and wrong.

**def loadBarChart():**  
input - none  
output - none  
do - create the bar chart that contain the user scores in the last previous sessions (up to 3).

**def register(userName, password, kwarg1=None):**input - userName: the username that user enter on the login page, password: the password that user enter on the login page  
output - none  
do - check the input, if some fields are empty or if username already in use, display a matched error message. If all inputs are filled correct it's entered the new user to the db, show a pop up and display the Login page.

**def clearMsgInPage():**  
input - none  
output - none  
do - clear an error message.

**def newUserExist(res, userName, kwarg1=None):**  
input - res: array that contain all users' details, username: the value that user enters in the username input in the Login page.  
output - True or False  
do - return true if the username already exists in the db.

**def connectCheck(userName, password, kwarg1=None):**  
input - userName: the value that user enters in the username input in the Login page, password: the value that user enters in the password input in the Login page.  
output - none  
do - if the user exists it's checked the user permission (admin/player) and display the page accordingly. In case the user enters invalid details, it's displayed an error message.

**def startGame():**  
input - none  
output - none  
do - call setQuestionNumberForGame function.

**def setQuestionNumberForGame(startGame, questionNumber, kwarg1=None):**  
input - questionNumber: the number of question that user choose to play in that session, startGame: equal to '1' if user press on the Start button, otherwise equal to '0'.  
output - none  
do - set the number of questions that user gone play in that session. Call load\_question function.

**def user\_exist\_check(userName, password):**  
input - userName: the user's username value, password: the user's password value.  
output - True or False  
do - return true if the username and the password are correct and initializes userDetails accordingly, otherwise return False.

**def admin\_check(userName):**input - userName: the user's username value,  
output - True or False  
do - return true if the user has admin permission, otherwise return False.

**def edit\_question(question\_id, level, question\_text, option1, option2, option3, option4, correct):**  
input - question\_id, level, question\_text, option1, option2, option3, option4, correct: the value that admin enters in the fields input in the AddQuestion page accordingly.  
output - none  
do - update the question value according to the input fields in the db and initializes resQuestion. Display a pop-up with success message.

**def remove\_question(question\_id):**  
input - question\_id: the question's id.  
output - none  
do - remove the question with the id provide in the input, initializes resQuestion and display a pop-up with success message.

**def addQuestion(question, correct, option1, option2, option3, option4, level, kwarg1=None):**  
input - question, correct, option1, option2, option3, option4, level: the value that admin enters in the fields input in the EditQuestion page accordingly.  
output - none  
do - check if there are no empty fields, if there is, display an error message. Otherwise, upload the new question to the db and add that data to resQuestion as well. Clear all input fields and display a pop-up with success message.

**def addVideoBtn(imgVideo, titleVideo, urlVideo, kwarg1=None):**  
input - imgVideo, titleVideo, urlVideo: the value that admin enters in the fields input in the AddVideo page accordingly.  
output - none  
do - check if there are no empty fields, if there is, display an error message. Otherwise, upload the new video to the db and add that data to resVideo as well. Clear all input fields and display a pop-up with success message.

**def display\_popup\_for\_user(msg):**input - msg: the message to display to the user.  
output - none   
do - display a pop-up with a message.

**def initSearchDropdownQuestions():**  
input - none  
output - none  
do - get the questions from the db initializes the dropdown.

**def searchQuestion(id, kwarg1=None):**  
input - id: the question's id that as been choose in the dropdown.  
output - none.  
do - for the given id it's display the question data.

**def getQuestionsFromDB():**input - none  
output - question data  
do - get a random question according to user score (score < 700 - easy question, score < 1500 - easy/medium question, above 1500 - easy/medium/hard)

**def load\_question():**  
input - none  
output - none  
do - display the Game page, call checkEndGame function, call getQuestionsFromDB and display the question and four answer.

**def checkEndGame():**  
input - none  
output - none  
do - check if this is the last question for the session if it is than display the next question button in the pop-up. Also update the countQuestionThatUserAnswerInGame and display it.

**def updataCountQuestionThatUserAnswerInGame():**  
input - none  
output - none  
do - raise the value by one in countQuestionThatUserAnswerInGame after user move to the next question.

**def updateCountAnswer(flag, kwarg1=None):**  
input - flag: contain '1' – if user answered correct or '0' – if user answered wrong.  
output - none  
do - if flag equal to '1' it raises the value countCorrectAnswer by one, otherwise it raises the value countWorngAnswer by one. Also, it counts the maximum questions that user answered in a row and update the additionalScore if needed.

**def updateBounsScore():**  
input - none  
output - none  
do - if user answered correct and press the end game it is update the bonus score, if needed.

**def updateCorrectAnswerAndScore():**  
input - none  
output - none  
do - enter the id question to the QuestionsThatBeenAnsweredCorrect array (contain all the id question that user answered correctly) and update the user score according to the question's level.

**def initFirstVideoOnPage(btnId, kwarg1=None):**  
input - btnId: contain 'nextPageVideo' or 'previousPageVideo'.  
output - none  
do - raise/decrease the valuein firstVideoOnPage by 6. Call loadVideoForPage function.

**def loadVideoForPage(startVideoOnPage):**input - startVideoOnPage: contain the video index that should be first in the page.  
output - none  
do - display 6 video on the Cinema page.

**def loadLeaderTable(leaderTableHTML = leaderTableHTML):**  
input - leaderTableHTML: Leader Table page.  
output - none  
do - call createRankUsers and display to the table the user rank and the top 5 users.

**def createRankUsers():**input - none  
output - sorted list by score with all users (not included admins)  
do - sorted all user according to their score, add 'userPlace' field to the user to indicate his rank.