**Maintenance guide**

In this part we will explain to the programmer what the project does, which includes the names of all the main files, main functions, code sections.

**2.1 Introduction**

The project is structured into three central parts:

1. Login screen that allows users to securely access the game. The login functionality incorporates Firebase for user authentication and retrieval of user information from a database.
2. The quiz game itself, which allows users to select the number of questions they want to play. The user's score and progress are tracked throughout the game, and at the end, a summary of their performance is displayed using visual elements like colored spheres and a pie chart.
3. Manager menu, which provides privileged functionalities for administrative tasks. This includes user management, analytics, and content updates. Managers can use this menu to update game content or questions as needed.

**2.2 Platform and technologies**

The project is implemented using Python and Jupyter Notebook, along with various technologies such as HTML, CSS, and JavaScript.

**2.3 Database environment**

The database used in the project is Firebase, which is a cloud-based NoSQL database platform provided by Google.

Collections:

תמונה שמכילה טקסט, כלי, עיצוב, נחושת

התיאור נוצר באופן אוטומטי

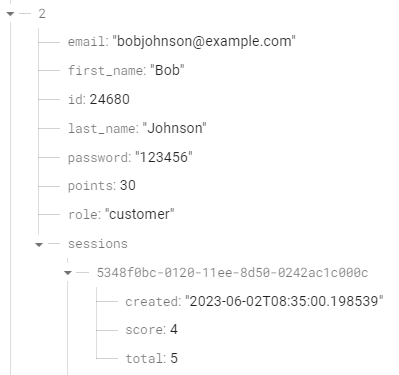
1. תמונה שמכילה טקסט, קבלה, גופן, אלגברה

   התיאור נוצר באופן אוטומטיQuestions:

The ‘Correct’ field represents the index of the correct answer.

1. Users:

A screenshot of a computer

Description automatically generated with medium confidence

For each user, we will meticulously save the following information:

a. Personal details:

b. Role: Manager or Customer (designating whether the user has managerial responsibilities or is a regular customer)

For each customer, we will additionally maintain the following details:

1. Sessions: A list of all the games the player has played:
   1. Game date and time.
   2. Total - Number of questions asked during the game.
   3. Score - Number of correct answers provided by the customer.

**2.4 Code structure**

**2.4.1 Questions part**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Description** | **Input** | **Output** |
| openMenu() | This function is responsible for displaying the main menu. It closes any existing widgets, clears the screen, and calls the main\_menu function to show the menu options. |  |  |
| main\_menu(user) | This function displays the main menu, where the user can select the number of questions to play. It also displays user information and provides a logout button. The function is called after the user logs in. | user (dictionary) - The user object containing user information. |  |
| start\_game(amount) | This function starts the game by specifying the number of questions to be played. It shuffles the questions, closes the menu and buttons for selecting the number of questions, and calls the ask\_question function to display the first question. | amount (integer) - The number of questions to be played. |  |
| ask\_question(question) | This function displays a question and its answer choices. It creates buttons for each answer choice, attaches click event handlers to the buttons, and handles the logic for checking the correctness of the selected answer. It also provides a "Continue" button to move to the next question or view the results. | question (dictionary) - The question object containing the question text, answer choices, and the index of the correct answer. |  |
| create\_pia(correct\_amount, total\_amount) | This function creates a pie chart to represent the correctness of answers. It takes the number of correct answers and the total number of questions and generates a pie chart with labels and colors representing correct and incorrect answers. | 1. correct\_amount (integer) - The number of correct answers. 2. total\_amount (integer) - The total number of questions. | image\_widget (Image widget) - The widget displaying the generated pie chart image. |
| open\_sessionGraph(b) | This function shows a summary of the player's last ten games. Show a motivational phrase is displayed within quotation marks. Create session a horizontal bar chart. The x-axis represents the games from oldest to newest, and the y-axis represents the accuracy percentage.and show total games and current accuracy. | b (button) - The "Your Profile" button that triggered the click event. | It generates the session graph interface |
| show\_session(): | The function calculates and plots a bar graph representing the accuracy of the user's performance in their last few games. It also calculates and displays additional profile information, such as the user's total number of games played and their current accuracy percentage. The function applies CSS styles to the displayed elements. | The function relies on the following global variables: user\_data, user\_name, motivational\_phrases. | Returns a **profile\_box** widget, which contains the generated session graph and the profile information. |
| on\_logout\_clicked(b) | This function handles the click event of the logout button. It closes all the existing widgets, displays a goodbye message, and redirects the user to the login screen after a delay. | The logout button that triggered the click event. |  |
| display\_correctness (correct\_amount) | This function generates colored spheres to represent question correctness. Green spheres indicate correct answers, while red spheres indicate incorrect answers. | correct\_amount (array) - An array of correctness values (1 for correct, 0 for incorrect). | html (HTML widget) - The HTML widget containing the colored spheres. |
| on\_continue\_clicked(b) | The "Continue" button click event by setting the continue\_exit\_value to True, performing actions based on the game state, and displaying the next question or the player's score with visual elements like a pie chart and correctness spheres. | b (button) - The "Continue" button that triggered the click event. |  |

**2.4.2 Info Screen**

|  |  |  |
| --- | --- | --- |
| **Description** | **Input** | **Output** |
| The info screen provides an introduction to the "OpenCloud Game." It encourages the user to play the game and provides some instructions on how to earn points by answering Python-related questions. |  | The output is an HTML page containing the styled content of the info screen. |

**2.4.3 Manager**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Description** | **Input** | **Output** |
| **manager\_Menu(user\_name)** | This function displays the manager menu, which includes options for inserting a new question and updating/deleting existing questions. It also displays a greeting message with the user's name. The function registers event handlers for the menu buttons. | **user\_name** (string): The name of the user (manager). | Displays the manager menu with the user's name and menu options (insert and update/delete buttons). |
| **insert\_button\_clicked(b)** | This function is an event handler for the "Insert Question" button. It creates a form for entering a new question and displays it using Jupyter widgets. The form includes fields for the question text, options, and the correct answer. It also registers the **submit\_form** function as the event handler for the form submission. | **b** (button): The button object that triggered the event. | Displays the form for entering a new question. |
| **update\_button\_clicked(b)** | This function is an event handler for the "Update/Delete Question" button. It clears the output and calls the **update\_page** function to display the update page, where the user can update or delete existing questions. | **b** (button): The button object that triggered the event. | Clears the output and displays the update page. |
| **insert\_question(question)** | This function is responsible for inserting a question into a Firebase database. It retrieves the existing questions from the database, assigns a unique ID to the new question, and inserts the question into the database using the Firebase connection (**FBconn**). It also handles the success and failure messages for the insertion. | **question** (dictionary): A dictionary containing the question details, including the question text, options, and the correct answer. | If the question is successfully inserted into the Firebase database, it displays the message "JSON object inserted into Firebase". If the insertion fails, it displays the message. |
| **submit\_form(button)** | This function handles the form submission when a user clicks the "Submit" button after entering a new question. It retrieves the values entered in the form fields, performs validation to check if any field is left empty, and displays error messages if necessary. If all fields are filled, it stores the form values in an array and calls the **insert\_question** function to insert the question into the database. | **button** (button): The button object representing the form submission. | If there are any validation errors, it displays error messages below the form fields. If the form is submitted successfully, it calls the insert\_question function to insert the question into the Firebase database. |
| **back\_on\_click(b)** | This function is an event handler for the "Back" button. It clears the output and returns to the manager menu by calling the **manager\_Menu** function. | **b** (button): The button object that triggered the event. | Clears the output and returns to the manager menu. |

**2.4.4 Update Manager**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Description** | **Input** | **Output** |
| **update\_page(m\_name)** | Displays a dropdown menu for selecting a question to update. | **m\_name** (string) - The name of the manager. |  |
| **handle\_dropdown\_change(change)** | Handles the value change event of the dropdown menu. | **change** (object) - The change event object triggered by the dropdown menu. |  |
| **show\_question(indexid)** | Displays a form for editing a specific question. | **indexid** (string) - The index identifier of the question to be displayed and edited. |  |
| **update\_question(question, index)** | Updates the data of a question in the Firebase database. | 1. question (dictionary) - The updated question data, including the question text, options, and correct answer. 2. index (integer) - The index of the question to be updated. | If the question is successfully updated in the Firebase database, it will print "Update Question in Firebase." After that, the manager\_Menu function will be called with the name parameter. |
| **delete\_question(index)** | Deletes a question from the Firebase database. | **index** (integer) - The index of the question to be deleted. | Deletes the corresponding question from the Firebase database. After a brief delay, the list of questions is refreshed from the database, the output is cleared, and the manager menu is displayed again. |
| **handle\_delete\_button(button)** | Handles the click event of the delete button. | **button** (object) - The delete button object. |  |
| **handle\_edit\_button\_click(button)** | Handles the click event of the edit button. | **button** (object) - The edit button object. |  |

**2.4.5 Login Screen**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Description** | **Input** | **Output** |
| **Login()** | This function displays a login screen using HTML and collects the username and password from the user. | No explicit input parameters. However, the function expects the username and password to be entered by the user through the HTML form displayed on the screen. | It triggers the **LoginFunc** function or displays an error message based on the input provided by the user. |
| **get\_user (name,password)** | The function get\_user searches for a user's information based on their name and password, returning the user object. | Takes two inputs, name and password, which represent the name and password of a user. | The user object (if the user is found and the credentials match) or False (if the user is not found). |
| **LoginFunc()** | This function validates the username and password provided by the user and performs specific actions based on the validation result. | The function takes two parameters: **username** (string) and **password** (string), which represent the username and password entered by the user. | It either calls the **main\_menu** or **manager\_Menu** function based on the user's role or displays an error message if the username or password is incorrect. |

A full explanation of the code structure is in the Excel file " CodeStructure.xlsx"