**תרגיל בית 3 - מחשוב ענן**

**פרויקט**

מגישים:

שרון מור – 207470329גל פולק – 208638015מיכל גינת – 208111146לידור אנקווה – 205818958טלי אזולאי – 205471204יובל בדיחי – 206850612

תמונה שמכילה קשת בענן, צבעוני

התיאור נוצר באופן אוטומטי

1. שם המערכת – **PythonChaser**מערכת זו היא מערכת וובית אשר מאפשרת לסטודנטים מתחילים ללמוד בצורה יעילה, נוחה ומהירה את נושאי פייתון דרך אתר אינטרנטי. החומר עצמו יהיה מותאם לאנשים אשר יש להם ידע בסיסי בעולם התוכנה אך לא מעבר. המערכת כוללת בתוכה תוכן איכותי ומגוון אשר נלקח מהספרות ומבעלי מקצוע מומחים אשר קשורים לנושאים השונים של עולם הPython. המערכת תאפשר לימוד החומר על ידי סרטונים ומענה על שאלות רלוונטיות. המשתמש יוכל להתחיל לשחק וללמוד ע"י מענה על השאלות. מענה על שאלות אלו יתנהל כשאלון טריוויה רספונסיבי - מענה על תשובה נכונה מתוך 4 תשובות אפשריות. המערכת מכילה מערכת ניקוד וציר התקדמות. בנוסף, מאפשרת שמירת סשנים למתן סטטיסטיקות אשר מוצגות בסוף המשחק ובדף הleaderboard. המערכת מאפשרת השתתפות כמשתמש רשום (הרשמה+התחברות) או כאורח. המערכת בנויה באופן סקלבילי ומאפשרת התרחבות בציר השאלות ובציר השחקנים. המערכת מאפשרת ממשק מנהל (עריכה, מחיקה, הוספת שאלות) בצורה יעילה ונוחה.
2. **דרישות פונקציונליות:**
3. המערכת תאפשר התחברות.
4. המערכת תאפשר הרשמה.
5. המערכת תאפשר שמירת סשנים של משחקים.
6. המערכת תאפשר התחלת מענה על שאלות.
7. המערכת תאפשר הצגת סטטיסטיקה.

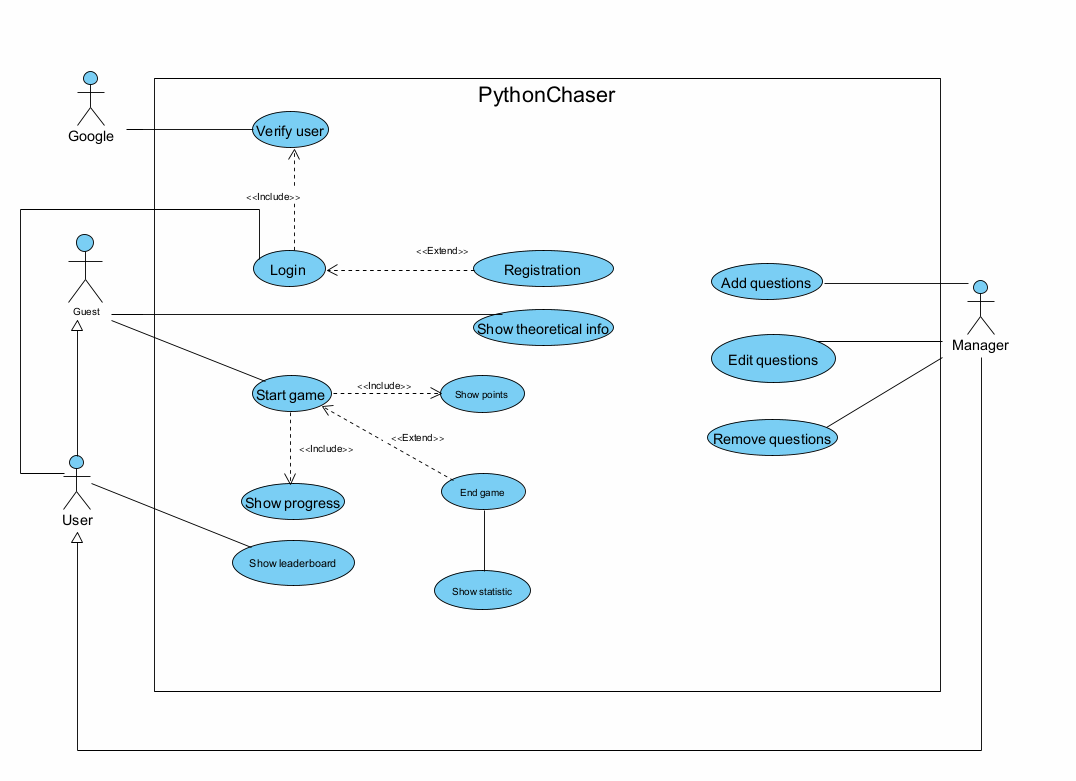
**דרישות לא פונקציונליות:**

1. זמן תגובה של האפליקצייה לא יעלה על שנייה (**Response Time**) – תוך שימוש נכון במשאבי מחשוב ענן(שימוש ב (GPU
2. אבטחת משתמשים ע"י שירותי אבטחה אמינים של firebase((**Security**
3. ממשק דף בית יעיל ונוח להתמצאות המשתמש הכולל: התחלת משחק,לוח מובילים,classes,עזרה,LogOut. ((**Usability**
4. הקוד מפותח בgoogle colab וינוהל בגיט תחת public repository

תחת (**Open Source**)

1. התמודדות וטיפול בשגיאות ינוטרו ויוצגו הודעות למשתמש בהתאם ((**Robustness**

**3.תרשים use-case:**



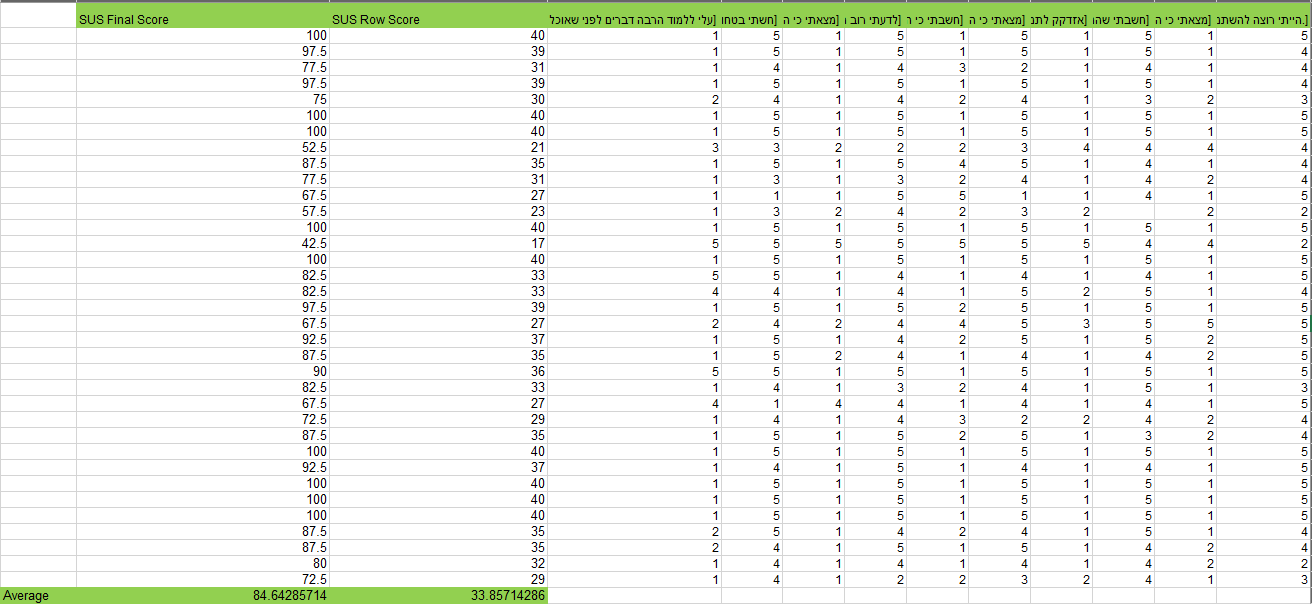
1. **איטרציות עבודה SDP:**

**איטרציה ראשונה:**

|  |  |  |  |
| --- | --- | --- | --- |
| Executor | Task | Requirement | Iteration |
|  |  |  | Iteration 1 |
|  | מסך הצגת פתיח למשחק | המערכת מאפשרת הצגת פתיח למשחק |  |
| UI developer - Yuval | הוספת תיאור משחק |  |  |
| Frontend developer – Sharon | הוספת כפתור מעבר לתחילת המשחק |  |  |
| UI developer - Yuval | עיצוב מסך הצגת פתיח משחק |  |  |
| QA – Tali | בדיקות קבלה (Acceptance Test) |  |  |
|  | מסך הצגת שאלות עם ניסיון מענה | המערכת מאפשרת מענה על שאלות |  |
| Backend developer - Lidor | שליפת השאלות מהdb |  |  |
| Frontend developer – Sharon | הצגת השאלה והתשובות בצורה שלדית |  |  |
| UI developer - Yuval | עיצוב מסך המשחק |  |  |
| Scrum master - Gal | הוספת לוגיקה - בחירה רנדומלית של שאלה והצגתו (ללא חזרות על שאלות) |  |  |
| Frontend developer – Sharon | הצגת התקדמות במשחק |  |  |
| Scrum master - Gal | הוספת לוגיקה – חישוב נקודות במשחק |  |  |
| Frontend developer – Sharon | הצגת נקודות |  |  |
| Frontend developer – Sharon | הוספת אפשרות סיום משחק לפני סיומו בפועל |  |  |
| Scrum master - Gal | שמירת סטטיסטיקות מענה השאלות |  |  |
| QA – Tali | בדיקות קבלה (Acceptance Test) |  |  |
|  | מסך סיום משחק (כולל גרף) | המערכת מאפשרת סיום משחק |  |
| Frontend developer – Sharon | הצגת נתוני המשחק |  |  |
| Scrum master - Gal | הצגת גרף המכיל כמות ניסיונות פר שאלה כתלות בשאלה |  |  |
| Frontend developer – Sharon | הוספת אפשרות חזרה למסך הראשי |  |  |
| UI developer - Yuval | עיצוב מסך סיום משחק |  |  |
| QA – Tali | בדיקות קבלה (Acceptance Test) |  |  |
|  | הכנת וחיבור DB | המערכת מאפשרת אחסון שאלה, תשובות, ומספר תשובה נכונה |  |
| Backend developer - Lidor  +  Scrum master - Gal | הכנת שאלות, תשובות לשאלות, ותשובה נכונה לצורך מאגר שאלות נרחב |  |  |
| Backend developer - Lidor | הוספת שאלות, תשובותיהן, ותשובה נכונה לתוך db |  |  |
| Backend developer - Lidor | יצירת חיבור לdb |  |  |
|  | מסך מנהל - הוספת שאלות ועריכתם | המערכת מאפשרת הוספה ועריכת שאלות מהמאגר השאלות הקיימות |  |
| Frontend developer – Sharon | הצגת האלמנטים הנדרשים להוספת שאלות (שאלה, 4 אופציות לתשובות, מספר התשובה הנכונה) |  |  |
| UI developer - Yuval | עיצוב מסך הוספת שאלות |  |  |
| Backend developer - Lidor | הוספת לוגיקה - הוספת שאלות למסד הנתונים |  |  |
| Backend developer - Lidor | שליפת השאלות מהdb (נעשה כבר במסך הצגת שאלות וניסיון מענה – משחק) |  |  |
| Scrum master - Gal | הצגת השאלות בצורה מסודרת והוספת אפשרות בחירה של שאלה ספציפית ותצוגה של שאלה זו, תשובותיה והתשובה הנכונה (מתוך נתוני השאלה שנשלפו) |  |  |
| Scrum master – Gal | הוספת האלמנטים הנדרשים לעריכה (אפשרות עריכה, תצוגה של הנתונים שיש כעת ואפשרות לשנותם, אפשרות שמירה/חזרה ללא שינוי) |  |  |
| Backend developer - Lidor | הוספת אפשרות מחיקה + עדכון במסד נתונים |  |  |
| Backend developer - Lidor | הוספת לוגיקה –עדכון שינויים של השאלה הנבחרת |  |  |
| UI developer - Yuval | עיצוב מסך עריכה ומחיקת שאלות |  |  |
| QA – Tali | בדיקות קבלה (Acceptance Test) |  |  |
|  | מסך שיעורים – צפייה בחומר הלימוד | המערכת מאפשרת בצפייה בחומר תיאורטי |  |
| Frontend developer – Sharon | הצגת חומר לימוד (סרטון לצפייה) |  |  |
| UI developer - Yuval | עיצוב מסך שיעורים |  |  |

**איטרציה שנייה:**

|  |  |  |  |
| --- | --- | --- | --- |
| Executor | Task | Requirement | Iteration |
|  |  |  | Iteration 2 |
|  |  | המערכת מאפשרת התחברות למשחק באמצעות שם משתמש וסיסמה |  |
| UI developer - Sharon Mor | הוספת האלמנטים הנדרשים לדף כניסה באמצעות שם משתמש וסיסמה ועיצוב דף הכניסה. |  |  |
| Frontend developer – Michal Ginat | בדיקת תקינות שם משתמש וסיסמה והצגת הודעה מתאימה במידה והנתונים לא תקינים. הוספת לוגיקה עבור כניסה לעמוד הראשי במידה והנתונים תקינים. |  |  |
| Backend developer – Gal Polak | יצירת שאילתה למסד נתונים לצורך בדיקה האם שם המשתמש והסיסמה תקינים ונמצאים במערכת.  יצירת שאילתה למסד הנתונים להבאת נתוני המשתמש לצורך הצגתם בעתיד. |  |  |
| QA – Yuval Badihi | בדיקות קבלה עבור כניסה עם שם משתמש וסיסמה. |  |  |
|  |  | המערכת מאפשרת הרשמה למשחק |  |
| UI developer - Sharon Mor | הוספת האלמנטים הנדרשים לדף הרשמה ועיצוב דף זה. |  |  |
| Frontend developer – Michal Ginat | בדיקת תקינות נתוני ההרשמה והצגת הודעה מתאימה במידה והנתונים לא תקינים. במידה והנתונים תקינים הוספת לוגיקה לבקשה עבור שמירת משתמש חדש. |  |  |
| Backend developer – Gal Polak | יצירת שאילתה למסד נתונים לצורך בדיקה האם קיים משתמש עם אותם נתונים.  יצירת שאילתה למסד הנתונים לשמירת משתמש חדש. |  |  |
| QA – Yuval Badihi | בדיקות קבלה עבור הרשמה כמשתמש במערכת. |  |  |
|  |  | המערכת מאפשרת כניסה כאורח |  |
| UI developer - Sharon Mor | הוספת האלמנט (כפתור) המאפשר כניסה כאורח לדף הכניסה. |  |  |
| Frontend developer – Michal Ginat | הוספת לוגיקה המאפשרת מעבר למשחק. |  |  |
| QA – Yuval Badihi | בדיקות קבלה עבור כניסה כאורח. |  |  |
|  |  | המערכת מאפשרת צפייה בלוח השחקנים המובילים במשחק |  |
| UI developer - Sharon Mor | הוספת טבלה שתכיל בתוכה את רשימת השחקנים המובילים ובנוסף עיצוב דף זה. |  |  |
| Frontend developer – Michal Ginat | הוספת לוגיקה להצגת השחקנים המובילים בתוך טבלה. |  |  |
| Backend developer – Gal Polak | יצירת שאילתה למסד נתונים לצורך הבאת הנתונים של 10 השחקנים המובילים של המשחק. |  |  |
| Frontend developer – Michal Ginat | הוספת לוגיקה לשמירת נתוני השחקן עבור כל משחקון בו השתתף (כולל ניקוד שצבר). |  |  |
| Scrum master – Lidor Ankava | יצירת שאילתה לשמירת נתוני משחקון עבור שחקן מסוים. |  |  |
| QA – Yuval Badihi | בדיקות קבלה עבור דף מובילי המשחק. |  |  |
|  |  | המערכת מאפשרת צפייה בגרף נוסף בסיום המשחק (מציג את כמות ניסיונות המענה עבור כל המשחקים עד כה) |  |
| UI developer - Sharon Mor | הוספת גרף pie, ועיצוב הגרף. |  |  |
| Frontend developer – Michal Ginat | הוספת לוגיקה להצגת נתוני השחקן בגרף (מכל המשחקים עד כה) עבור כמות הפעמים שהצליח בפעם הראשונה לענות נכון, בפעם השנייה, בפעם השלישית והרביעית. |  |  |
| QA – Yuval Badihi | בדיקות קבלה עבור הגרף המבוקש. |  |  |
|  |  | המערכת מאפשרת למנהל לעדכן שאלות |  |
| Backend developer – Gal Polak | הוספת שאילתא למסד הנתונים האם המשתמש שנכנס הוא מנהל. |  |  |
| Frontend developer – Michal Ginat | הצגת האפשרות לעריכה ועדכון שאלות עבור מנהל בלבד. |  |  |
| QA – Yuval Badihi | בדיקות קבלה עבור כניסה כמנהל. |  |  |
|  |  | המערכת מאפשרת מעברים חלקים בין הדפים השונים של המערכת |  |
| Scrum master – Lidor Ankava | סידור ויעילות הקוד. |  |  |
| QA – Yuval Badihi | בדיקות קבלה עבור מעברים חלקים בחלקים השונים של המערכת. |  |  |
|  |  | המערכת מאפשרת מעבר ידני בין השאלות השונות במשחק |  |
| UI developer - Sharon Mor | הוספת האלמנטים (כפתורים) הנדרשים עבור מעבר ידני בין השאלות במשחקון. |  |  |
| Frontend developer – Michal Ginat | הוספת לוגיקה עבור מעבר ידני. |  |  |
| QA – Yuval Badihi | בדיקות קבלה עבור מעברים ידניים בין השאלות במשחקון. |  |  |

**5א**.**חישוב ציון SUS:**

**5ב. תיק מתכנת:**

**Maintenance Guide**

This guide provides instructions for maintaining the codebase and ensuring its proper functioning. It covers the necessary steps to follow and the various components involved in the code.

**This code uses several libraries and modules, including:**

* ipywidgets
* firebase
* IPython
* google.colab
* matplotlib
* requests

Make sure to periodically check for updates to these libraries and install the latest versions as needed.

**Database environment**

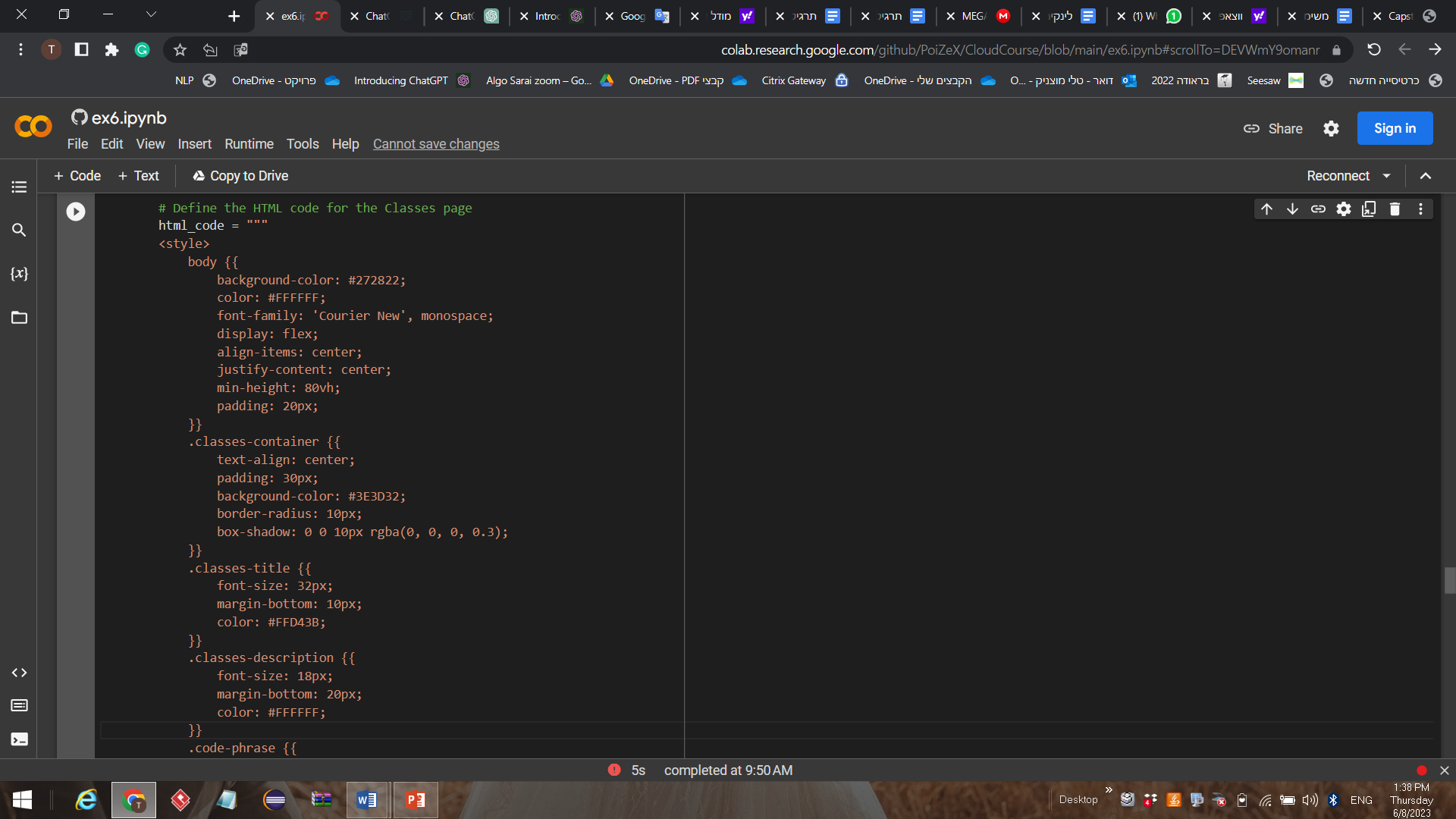
The code integrates with Firebase 4.0.1 to store and retrieve data. The firebase library is used for this purpose. Ensure that the Firebase project is properly set up and the Firebase database rules allow the required operations (read, write, etc.) as per your application's needs.

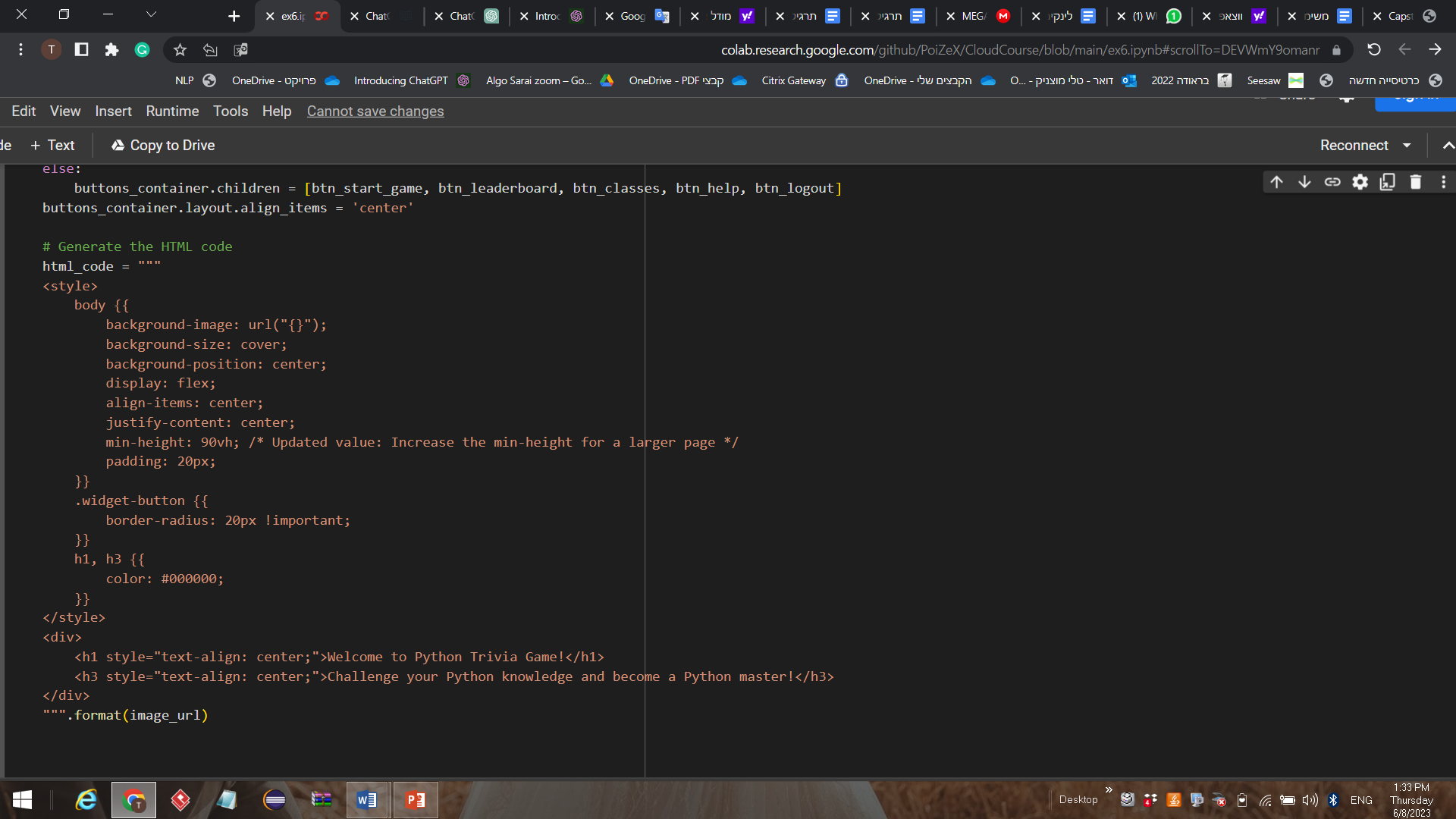
To connect for database, run this line: !pip install firebase

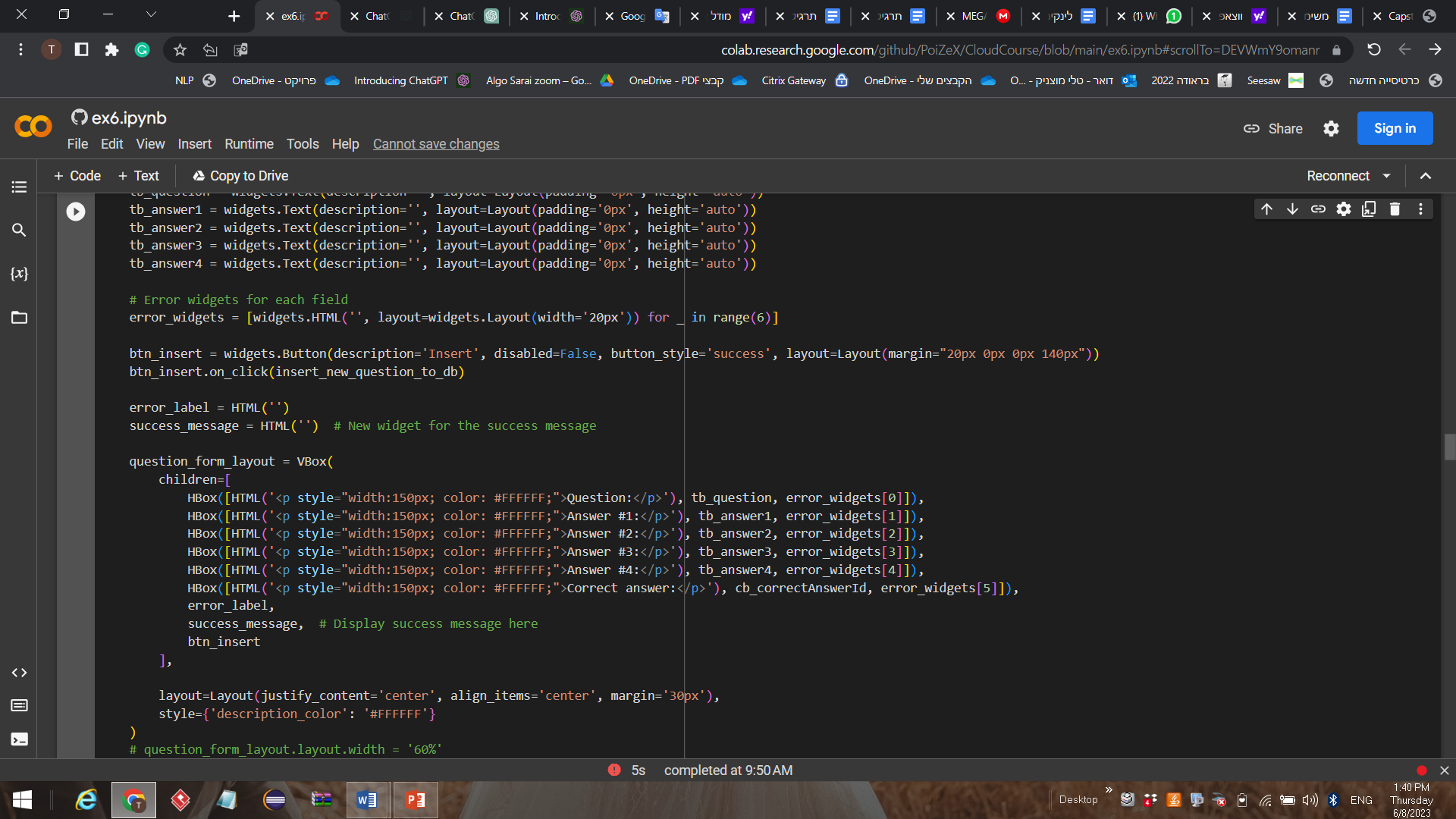
* The code uses Firebase as the database provider. Ensure that the Firebase SDK is properly installed and configured.
* Check the `FIREBASE\_PATH` variable to ensure it contains the correct path to your Firebase database.
* Verify that the `firebase` library is imported and compatible with the code.

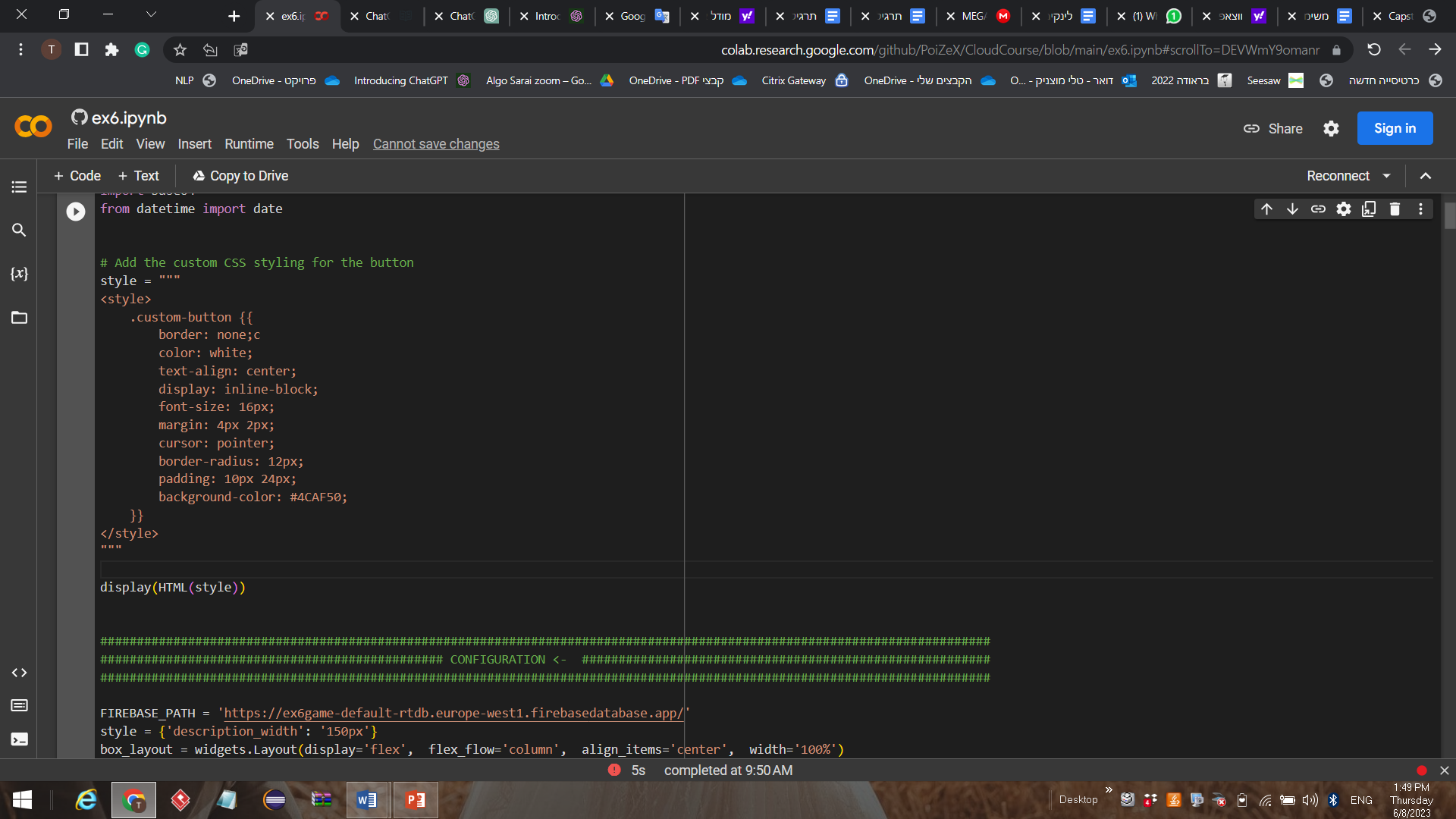
**`firebase.FirebaseApplication`:** This class is used to establish a connection with the Firebase database and perform operations such as getting data (`get`), deleting data (`delete`), and updating data.

**User Interface:** For the UI we use HTML code style for the classes page, Widgets and CSS for button style.





 Present a form for adding new questions into DB



**Main functions:**

**user login**

1. **is\_user\_Exist(username)** function used to check if a user with a specific `username` already exists in the database. This can be useful when someone is trying to register a new account to ensure that the chosen username is unique.
2. **validate\_login(username, password)** function used to validate the login information provided by a user.If a match is found, the function sets the `current\_user` variable, indicating that the user is now logged in .If no match is found, an error message will be displayed to the user.
3. **register\_page(b=None)** function used to display a registration page to the user. This page typically includes text input fields for the desired `username`, `password`, and `confirm password`. The user can enter their chosen credentials, and when they click the registration button, the `sign\_up` function is called to create a new user account with the provided information.

**Design Patterns:**

**Singleton Pattern**: The User class implementing the Singleton pattern. It ensures that only one instance of the class can be created, which is relevant in the context of representing the currently logged-in user.

**For the menu page:**

**show\_menu\_page()** function creates the main menu of the Game, defines button callbacks for each menu option, and utilizes HTML and widgets to present a visually appealing and interactive user interface.

**generate\_leaderboard()** function retrieves the scores from a Firebase database and generates an HTML table to display the leaderboard. The function starts by establishing a connection to the Firebase database using the FirebaseApplication method. To display the results of all users who played the game

**Function for admin edit page**

**AdminEditPage()** function is responsible for displaying the admin edit page. it iterates over each

question in the list of questions retrieved from the database and creates a button for each question.

**insert\_new\_question\_to\_db:** Insert a new question into the database. It retrieves the values entered in the form fields

**show\_add\_questions\_page:** Display a form for adding new questions to the database,Create the form layout using `VBox` and `HBox` containers**.**

**Function to show the results page to the user**

**show\_user\_statistics(combined\_list)** function: Generates and displays a pie chart that represents the distribution of the total number of attempts for each game. This function uses the matplotlib.pyplot library to create the pie chart

Parameters: combined\_list: A list containing the number of attempts for each game.

**show\_results\_page()** function: Displays the results page after the game is over.This function generates HTML content and uses the display() function to show the content.

**Saving points:**

**incrementPointsToUser** **(username, points\_to\_increment)** a function that increments the total points of a user in a Firebase database based on their username, and updates the total points in the database using FBconn.put. It compares the usernames one by one until it finds a match. Once a match is found, it retrieves the user's information

**get\_user\_stats(username)** used to retrieve the statistics of a user from a Firebase database looks for the user's scores in the database and extracts the statistics associated with those scores.

**update\_points(new\_points, display\_points=True):** This function updates the points variable of user based on his answer. It takes new\_points as a parameter, representing the points to be added.

**The question logic:**

**on\_answer\_button\_clicked** function handles the logic for processing user interactions with answer buttons in a game. It updates statistics, evaluates correctness, updates points by checking if it is his first try (receives the max points), and manages the display.

**show\_success\_dialog** the func changes the button style of the continue\_button global variable to 'success', and presents a success dialog by clearing any previous errors, updating the style of the continue button, and enabling it for further interaction.

**display\_questions** function sets up and displays the elements of the questions page in a GUI. It handles the presentation of the question, points label, progress bar, answer buttons, and error output.

**show\_questions\_page** function is responsible for generating and displaying a new question on the questions page.The function calls the get\_random\_question method of the questionsFromDB object to retrieve a random question from the database.

**Main classes:**

1.**WidgetEnum** enumeration defines different types of widgets. 2.**User class** represents a user in the system, with attributes for username, password, role, and total points. It provides methods to check if the user is an admin, retrieve the username, and get the total points. 3.**Score** class represents a game score, with attributes for the date, username, statistics list, and points earned. It provides methods to retrieve the statistics list and game points. 4.**Question** class represents a single question in the quiz game and provides methods for accessing and manipulating question data. 5.**QuestionsDB** class manages the collection of questions, including retrieving random questions, adding questions, and maintaining a local list of questions.

6. **StatsDB** represents a statistics database and provides methods to retrieve statistics based on user ID and date range.

**Conclusions**

Throughout the development phase we faced many challenges. Mainly in:

* **Performance Optimization:** As the game grows and handles a larger number of questions, users, and interactions, optimizing performance becomes crucial. Ensuring smooth gameplay, minimizing loading times, and managing memory usage was challenging and may require optimizing database queries, and implementing caching mechanisms.
* **Game Logic**: Implementing the game's logic, including tracking scores, handling user input, and managing game flow, was little complex. Ensuring the game functions correctly by thinking about interesting graphs for user and combine attractive design interface was challenging **In conclusion, we think we have created an interesting and attractive game. Despite the challenges, we are happy to achieved our goal**

**5ג. תיק משתמש:**

**User documentation**

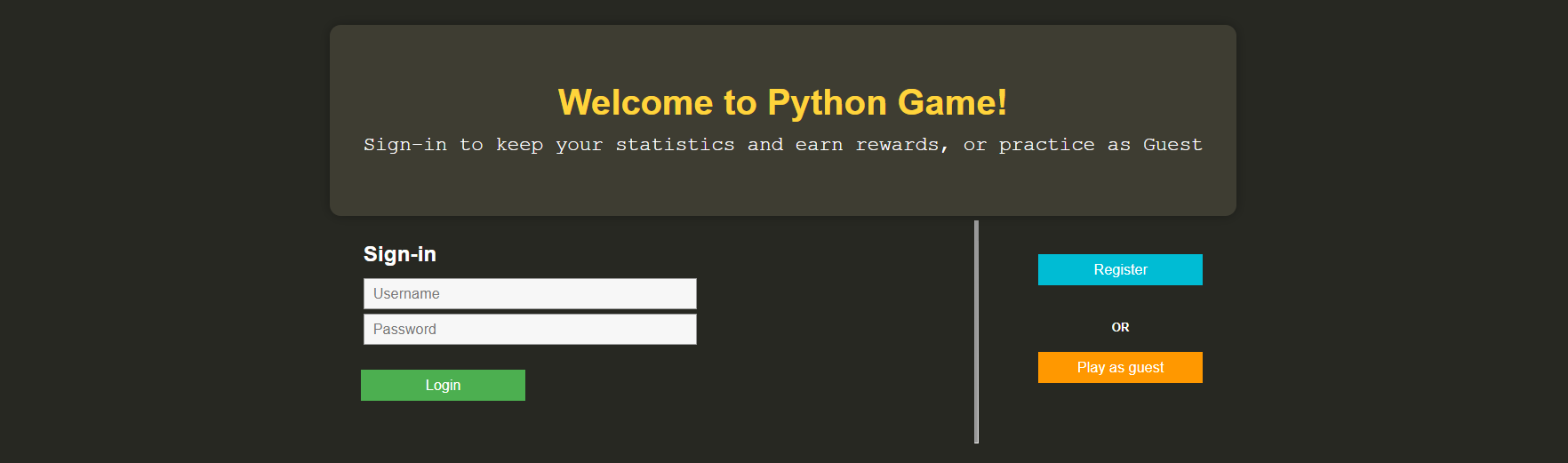
**General description:**

This Python trivia game challenges players' knowledge of Python through a series of questions. Each question has four possible answers, and the objective is to choose the correct one and earn points. Players can earn 3 points for a correct answer on the first attempt, 2 points for a correct answer on the second attempt, 1 point on the third attempt and no points if they exhaust all attempts. Before starting the game, there is an optional feature to access Python classes for knowledge enhancement. Administrators can edit questions to keep the game up to date with the latest Python concepts.

**User interface:**

* **Welcome screen**This is the first screen that the user sees. He can register for the game to maintain his statistics and earn rewards. If the user prefers to play as a guest, there is also an option to skip the registration process. By clicking on "Play as guest" button, the user will be directly taken to the home page of the game without any additional steps.

For those who are already registered, signing into the game is possible by entering the username and password. Once the credentials have been entered, clicking on the "Login" button will lead them to the home page of the game.



Possible mistakes: The user may encounter alerts when entering incorrect usernames or passwords, or if they forget to enter one/two of them.

* **Registration screen**  
  A "Register" button is available in the previous screen, which will direct him to this screen where he can enter a desired username, password, and confirm the password. After entering the required information, he can click on the "Submit" button to complete the registration process. Upon successful registration, he will be redirected to the home page screen. He can also go back to the previous screen by clicking on "Back" button.

תמונה שמכילה צילום מסך, טקסט, עיצוב

התיאור נוצר באופן אוטומטי

Possible mistakes: If the user mistakenly confirms an incorrect password or enters an existing username, they will receive an alert indicating the error.

* **Home screen** (**not administrator users)**

Within the game interface, the user has several options available. They can initiate the game by clicking on the "Start Game" button, view the leaderboard to track their progress and compare scores, explore various classes of Python to enhance their knowledge, access helpful resources by clicking on the "Help" button, and finally, logout from the game when they wish to exit or switch accounts (will direct him to Welcome page).



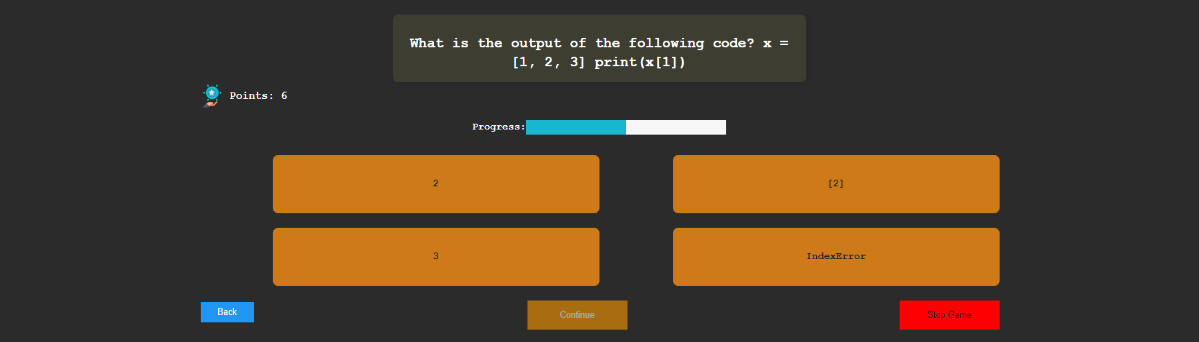
* **Game screen**
  + **Game explanation screen**

When starting the game, the user is presented with an explanation screen and can proceed by clicking "Continue" or go back to the home page.

תמונה שמכילה טקסט, צילום מסך, עיצוב

התיאור נוצר באופן אוטומטי

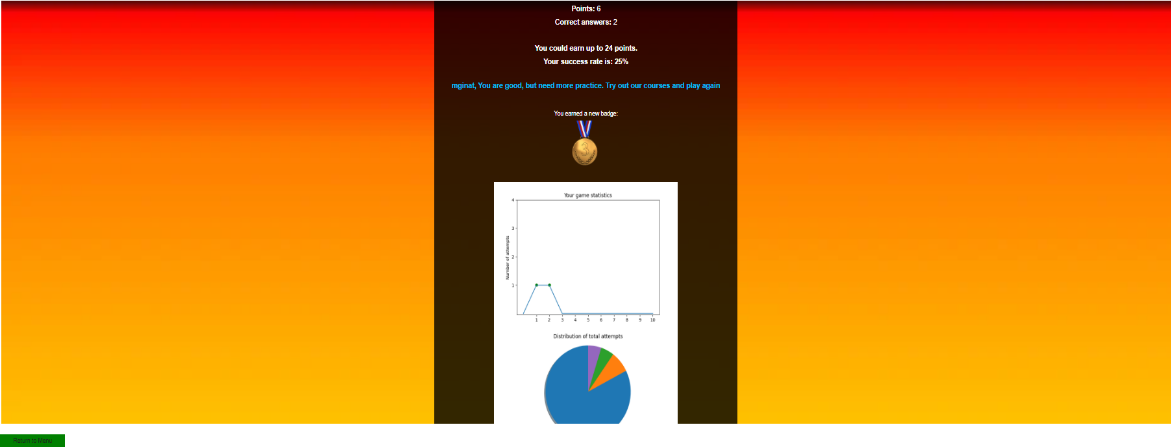
Users select the right answer for each question. Getting it right on the first try earns 3 points, on the second try 2 points, on the third try 1 point. No points for the last available answer. Incorrect answers are red, correct answers are green. Users can see their points, go back or continue (will be enabled only after the user has selected one possible answer), or stop the game. Also displays user's points and a progress bar.



* **End game screen**

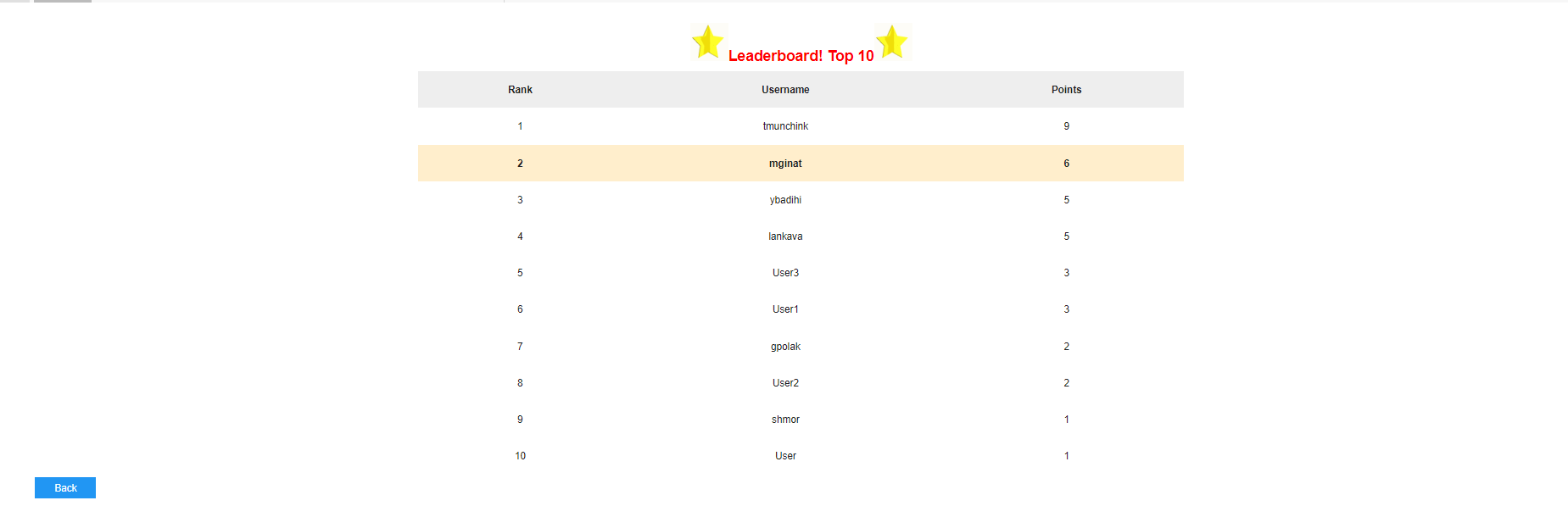
Once the user has finished answering all the questions or after clicking on "Stop Game" button, he will be directed to a summary screen. This screen displays the total points earned, the number of correct answers, the potential points that could have been earned, and the success rate. Additionally, the user is awarded a game budget based on the points achieved.

The summary screen also includes two graphs: one representing the game statistics (number of attempts) and another displaying the distribution of total attempts. To return to the home page, the user can simply click the green button located on the left side of the page.



* **Leaderboard screen**

The user has the option to view the leaderboard, which displays the top ten players. If the user is among the top ten, their name will be highlighted. Otherwise, their position in the leaderboard will be shown. The user can go back to home page screen.

****

* **Classes screen**

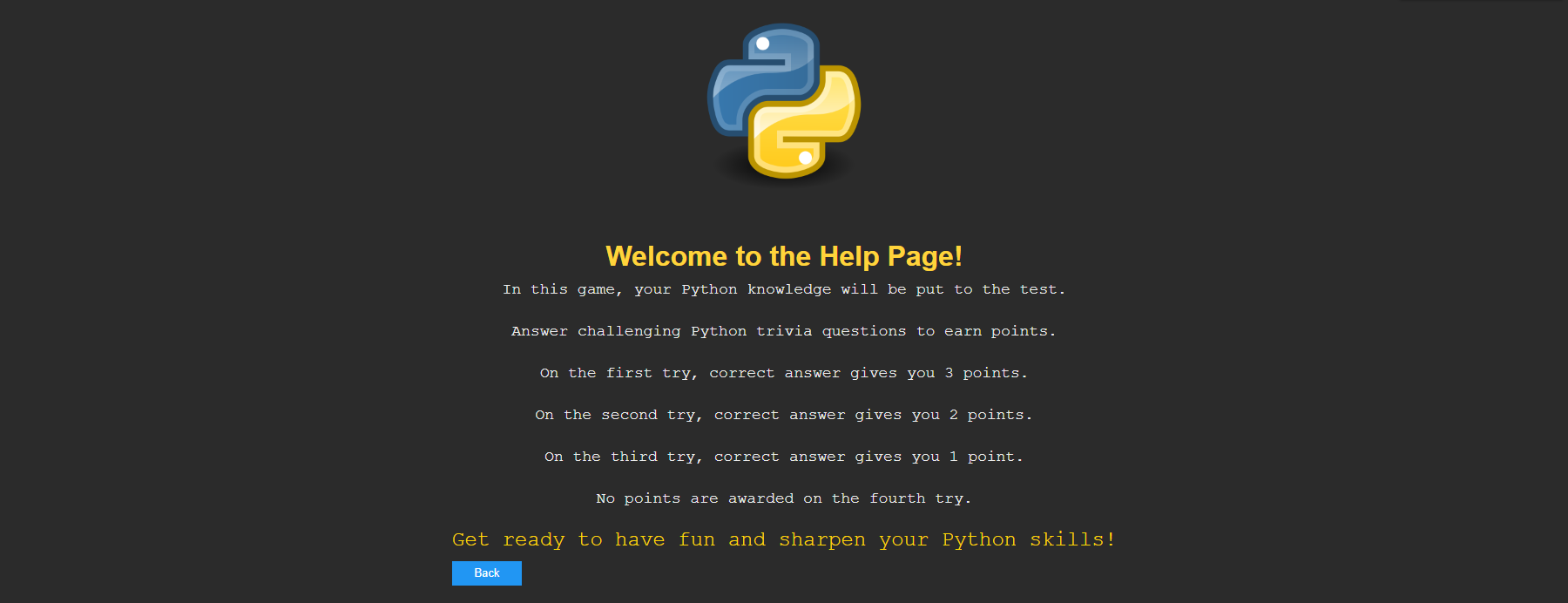
Before starting the game, the user can access informative Python classes. They can watch an educational video by clicking on it to gain knowledge prior to playing. Additionally, the user has the option to easily navigate back to the home page if desired.

תמונה שמכילה טקסט, צילום מסך, עיצוב גרפי, עיצוב

התיאור נוצר באופן אוטומטי

* **Help screen**

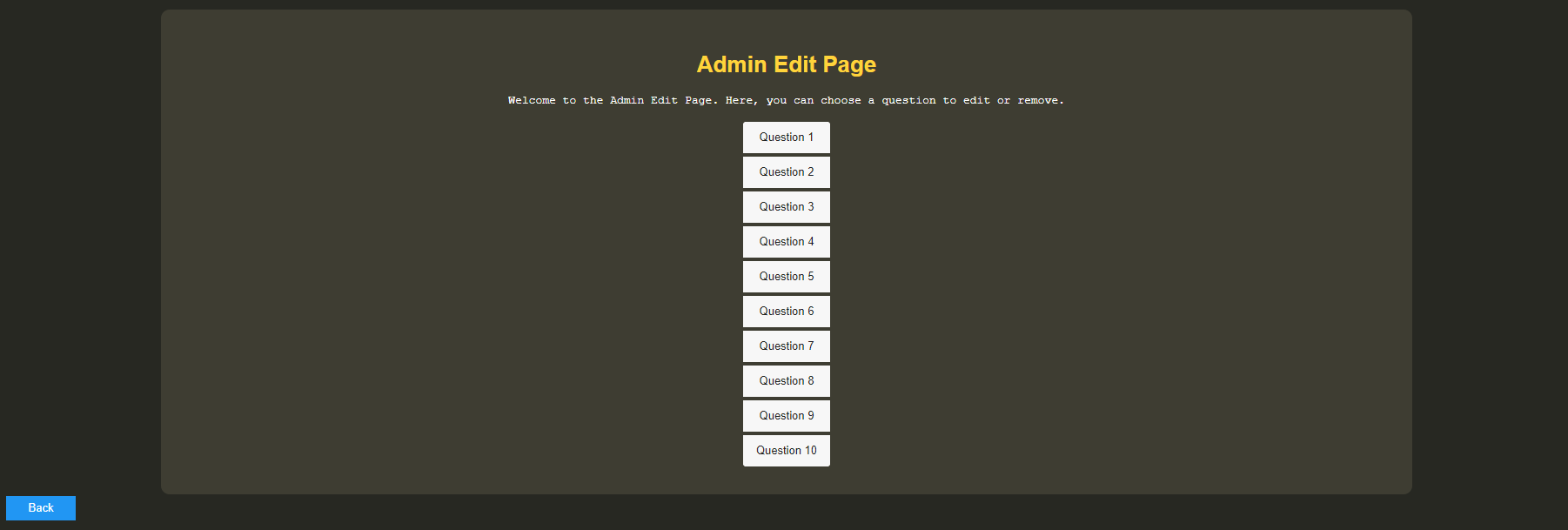
On the home page screen, users have the option to click on the "Help" button before starting the game. This feature provides a description and explanation of the game, offering users valuable information to understand its mechanics and rules. Additionally, the user has the option to easily navigate back to the home page if desired.



* **Home screen** (**administrator users)**

For administrators, their home page is like regular users, but with the added functionality of two additional buttons: "Edit Questions" and "Add Questions". These buttons provide administrators the ability to modify existing questions or add new ones, ensuring the game remains dynamic and up to date with the latest content.



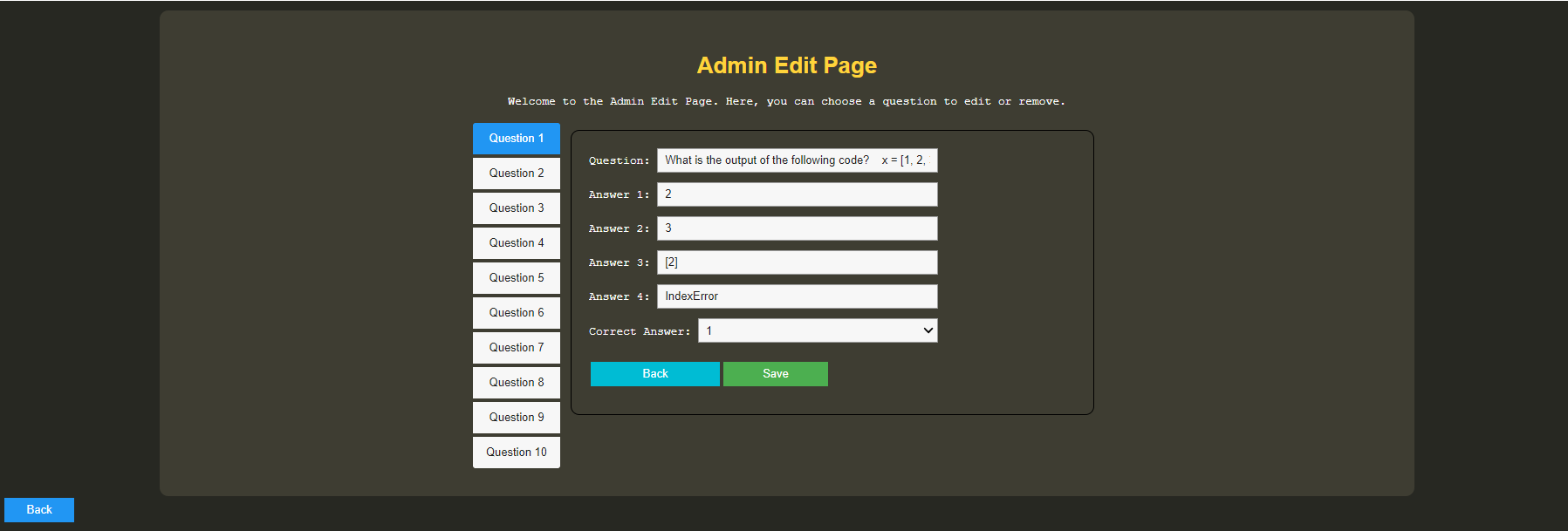
* **Edit questions**As an administrator, he has the option to edit questions by clicking on the "Questions" button. At any time, he can choose to go back to the home page.
* 

Once the desired question is selected, it will be taken the user to a screen where he can edit or remove the question by clicking on the corresponding buttons. The question will be displayed, with the correct answer highlighted in green. He has the flexibility to edit any other parts of the question by clicking on them. At any time, he can choose to go back to the home page.

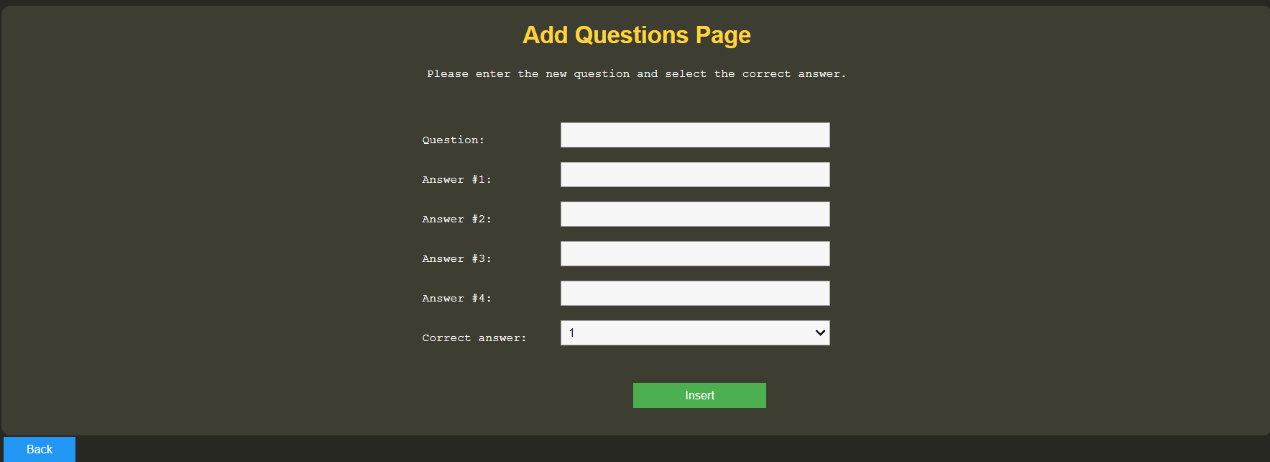
תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

If the user chooses to edit the question, the following screen will be presented to him. He can modify the question text, possible answers, and the correct answer by clicking on the respective combo-boxes. To save the changes, he can simply click on the "Save" button. If he wants to go back to the previous page, he has that option as well. Additionally, he can edit other questions by clicking on them. At any time, he can choose to go back to the home page by clicking the "Back" button at the left.



* **Add questions**  
  As an administrator, he has the privilege to add questions. In this screen, he can enter the question, along with the possible answers and the correct answer for that question. Once he has entered all the necessary information, simply click on the "Insert" button to add the question to the game.



**5ד.** קישור לתיקיית הגיט בה נמצאים כל קבצי הפרויקט:

<https://github.com/PoiZeX/CloudCourse/tree/main/PyProject>

קישור למסד הנתונים:  
<https://console.firebase.google.com/u/0/project/ex6game/database/ex6game-default-rtdb/data/~2F>

**6. אתגרים:**

* סביבה לא מוכרת: את הפרויקט יצרנו בסביבת פיתוח Google Colab, סביבה שלא נתקלנו בה בעבר. היינו צריכים להשקיע זמן רב להבין כיצד עלינו לתפעל את הסביבה כראוי.
* חשיבה מחוץ לקופסה: מאחר וקבוצות רבות קיבלו את אותה המשימה, נתקלנו בבעיה של ייחודיות. לשם כך התכנסנו וביצענו חשיבה מסתעפת ולאחריה חשיבה מתכנסת.גם כאן כאשר חשבנו מחוץ לקופסה, התחשבנו במגבלות השונות העומדות לפנינו כמו סביבת עבודה וזמנים.
* קושי בדיבוג: סביבת העבודה הנוכחית Google colab אינה מאפשרת דיבוג הקוד. על-מנת לאתר שגיאות נדרשנו לבצע הדפסות ולהריץ מספר רב של פעמים. זאת בנוסף לחוסר מספור השורות, אשר הקשה אף יותר על התמצאות.
* עמידה בזמנים: במהלך הפרויקט עמדו לפנינו דד-ליינים רבים של הגשות, דבר שאתגר אותנו יחד עם קורסים נוספים שלקחנו. בנוסף, עקב גודל הקבוצה קביעת מועד פגישה היתה לא פשוטה והקשתה בעמידה בזמנים.
* קושי עיצובי: עיצוב המשחק היה חלק מרכזי בבניית הפרוייקט, עיצוב המשחק תחת ההגבלות של גוגל קולאב היווה אתגר מרכזי שכן הרעיונות עליהן הסכמנו כקבוצה לא תמיד מומשו כנדרש