פרוייקט סופי מחשוב ענן - BEE  
רז קסל 206017725  
נבו גוטליב 206234353  
איתי בנג'מין 316514207  
דניאל ממן 318155884   
רום הראל 318474582  
בן אבנר מריג'ן 209240274

**1. Design thinking**

**רעיון האתר:**  
  
"מי רוצה להיות מתכנת?"

משחק דפדפן המאפשר לשחקנים להרחיב את ידיעותיהם בשפת התכנות Python. המשחק מתנהל באופן דומה למשחק המוכר "מי רוצה להיות מיליונר", אך במקום שאלות טריוויה וידע כללי, יוצגו למשתתפים שאלות בנושאים שונים הנוגעים לשפת התכנות Python. בנוסף לאופציית המשחק, יכיל האתר מדריך מקוצר לשחקנים מתחילים בו יוכל המשתמש להתרשם מהמשחק ולהבין לעומק כיצד לשחק ולבצע שימוש ושטף באתר. כמו כן, המשתמש יוכל לצפות בתהליך הלמידה שלו אשר יוצג על גבי גרף המתעדכן באופן שוטף כתלות בתוצאות משחקיו. בנוסף יכיל האתר טבלה מפורטת של חמשת השחקנים המובילים נכון לאותה עת.  
לאתר יוגדר פאנל מנהל , בו ניתן יהיה להוסיף שאלות נוספות, לעדכן שאלות קיימות ובמקרה הצורך אף למחוק שאלות לא רלוונטיות כך שתמיד המערכת שלנו תישאר עדכנית.

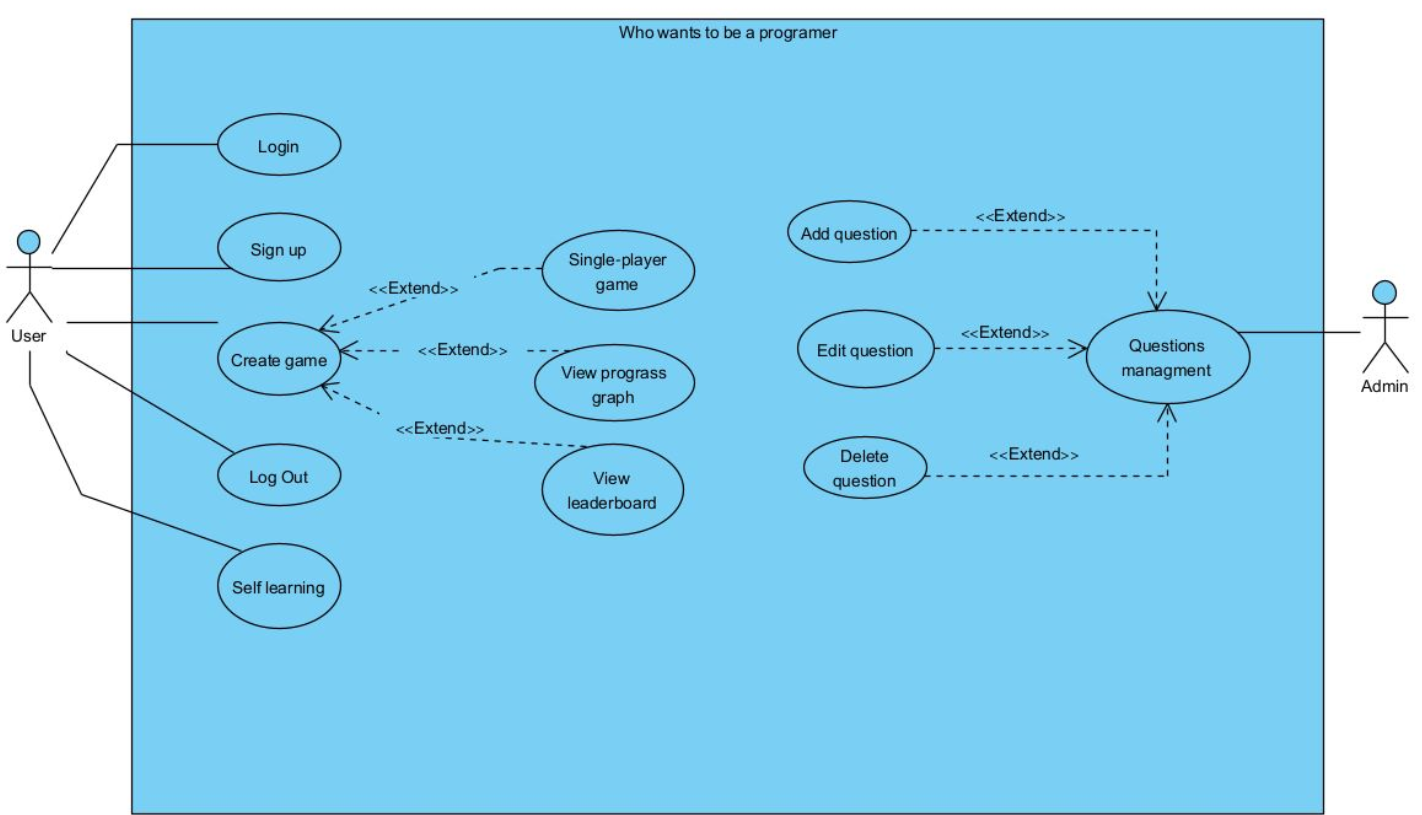
**דרישות מהמערכת:**

**דרישות פונקציונליות:**

* המערכת תאפשר התחברות לאתר.
* המערכת תאפשר משחק Single-player.
* המערכת תאפשר לראות טבלה בה מוצגים חמשת השחקנים בעלי התוצאה הגבוהה ביותר.
* המערכת תאפשר לראות גרף תוצאות אישי המתאר את 15 המשחקים האחרונים של השחקן. לאחר כל משחק.
* המערכת תספק למשתמש מדריך לשימוש ראשוני באתר

**דרישות לא פונקציונליות:**

* Privacy- נתוני המשתמשים לא יהיו חשופים לכלל משתמשי במערכת.
* Response time - מעברי האנימציות באתר יתרחשו באופן מיידי וללא השהיות.
* Accessibility - עיצוב המערכת יותאם גם לבעלי מוגבלויות ויבוסס על עקרונות העיצוב של נורמן.
* Portability – ניתן להשתמש במערכת בסוגים שונים של פלטפורומות/דפדפנים ללא צורך בהתקנות מיוחדות.
* Scalability - המערכת תתוכנן כך שתוכל להתמודד עם כמויות משתנות של משתמשים רשומים לאתר.

**תרשים Use Case של האתר: **

**להלן 2 הטבלאות של האיטרציות שביצענו במהלך הפרויקט:**

**איטרציה 1:**

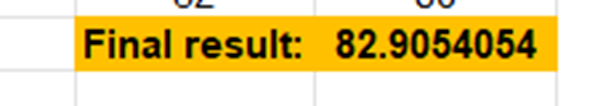
|  |  |  |  |
| --- | --- | --- | --- |
| Executor | Task | Requirement | Iteration |
|  |  |  | Iteration 1 |
| All the team:  Daniel  Raz  Nevo  Rom  Itay  Ben | Learn HTML , JavaScript, CSS, Python, Json, Bootstrap, firebase  And how to integrate the whole materials together. | learning relevant software and languages | Task 1 |
| Backend developer - Nevo + frontend developer - Itai | Open a realtime data base in firebase  Create an acceptable schemes of questions and users.  Create a json file with users, question.  Push json file to firebase. | Create DB | Task 2 |
| All members group:  Daniel  Raz  Nevo  Rom  Itay  Ben | Interview with a potential user  divergent thinking  convergent thinking  F/NF requirements  Use Case  prototype sketch | Design thinking | Task 3 |
| scrum master - Raz + *UI - Daniel* | create html page for each screen  integrate with js logic | Create game screens | Task 4 |
| frontend developer - Itai + Backend developer - Nevo | Building the logic of the game.  integrate the logic with the design. | Build game logic | Task 5 |
| product manager - Rom + *QA - Ben* | Create:  "add function" option and insert it to DB.   "delete function" option and delete a chosen question from DB. | Create admin panel | Task 6 |
| *UI - Daniel +* scrum master - Raz | design with Bootstrap | Create UI view | Task 7 |
| *QA - Ben +* product manager - Rom | **The acceptance test table is below** | Acceptance Testing | Task 8 |

**איטרציה 2:**

|  |  |  |  |
| --- | --- | --- | --- |
| Executor | Task | Requirement | Iteration |
|  |  |  | Iteration 2 |
| Backend developer – Itai  Scrum master - Rom | Implement a scoring system.  Implement a leaderboard.  Add an option to quit during the game.  Improve error handling.  Implement graph logic that shows last 15 results.  Implement leaderboard logic by total score of users.  Improve progress bar functionality. | Enhance game logic | Task 1 |
| UI developer - Raz | Improve question display.  Improve a result screen.    by adding graph and table (leaderboard) to show results  Add some back buttons . | Improve game screens | Task 2 |
| Frontend developer – Ben | Add an update option with an additional button.  Add a remove option with an additional button.  Add some labels to the panel. | Improve admin panel screens | Task 3 |
| Scrum master - Rom | Goes over the feedbacks we got, summarize issues, and fix it | Implement user feedback | Task 4 |
| QA – Daniel | **The acceptance test table is below.   Run all tests again from the last iteration to check integration with all new changes.** | Create more acceptance Test | Task 5 |
| Product manager - Nevo | Create gameplay guide using the template as defined on the course website. | User gameplay guide | Task 6 |
| All members group:  Daniel Raz Nevo Rom Itay Ben | documentation of all code files. | Add documentation to the code | Task 7 |
| QA -Daniel  Scrum master - Rom | A video that showing our game and marketing it.  Show main functionality and promote the website.  Using an AI announcer | Promotional video | Task 8 |
| Product manager – Nevo   UI developer - Raz | Description of application.  An explanation of how it works.  Explain which software and database we used.  Description of all the functions  Create a Scheme of the project to show communication between services using Drawio. | Maintenance guide | Task 9 |
| Backend developer – Itai  Frontend developer – Ben | verify a user's database, which includes a username, password and saving all the sessions the user took part in (including his score in each session he participated in).  Check again the acceptance test from the previous iteration. | Verify database as the requirements | Task 10 |

סעיף 3

**שאלון *SUS*:**

****

נציין כי ציון זה מציב אותנו על פי שיטת הערכה של ***SUS*** ב 10% העליונים.

הציון סיפק אותנו אך כמובן ששאפנו לציון גבוה יותר. בעקבות כך שיפרנו את המערכת על סמך ההערות והמשובים שקיבלנו בכדי שכמה שיותר אנשים יהיו מרוצים והמערכת תהיה יעילה ונגישה לכולם ברמה הכי גבוהה שיש.

נציין שהשאלון עם המידע ותהליך החישוב מצורף בקבצי הזיפ בתרגיל בית 3 חלק א.

**תיק משתמש:**

**User instructions**

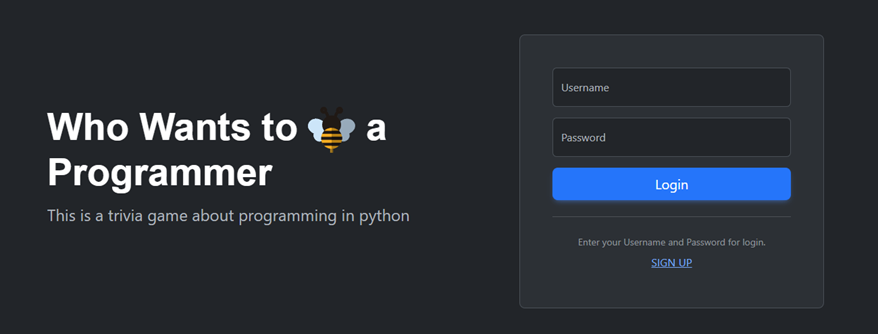
The game "Who wants to be a programmer" is an attractive game designed to improve the user's abilities in the field of Python.  
 The user receives questions related to the field and must answer based on his knowledge base.  
 He will receive a status for his past results and a graph that will show his improvement trend in results over the last 15 games.

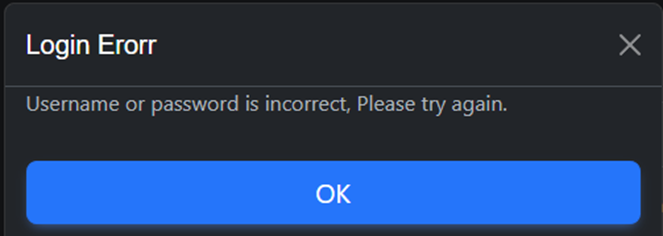
**User Login Screen:**

1. Upon launching the application, you will be presented with a user login screen.

2. Enter username and password access the application.

3. If you enter incorrect username or password, you will receive an error message: "Username or password is incorrect, Please try again."

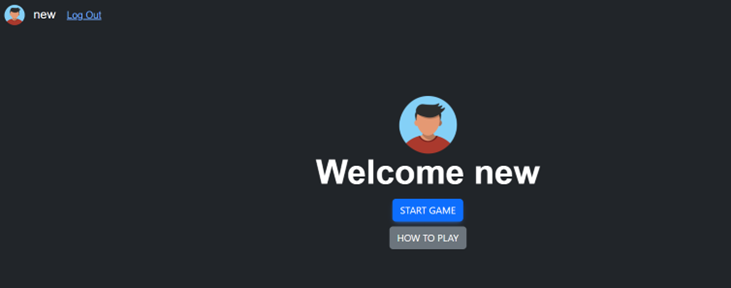
****

****

**Home page:**

1. After successful login, you will be directed to the Home page. Choose between starting a game or how to play (new is the name of the user).

2. You can log out of the user if you want to



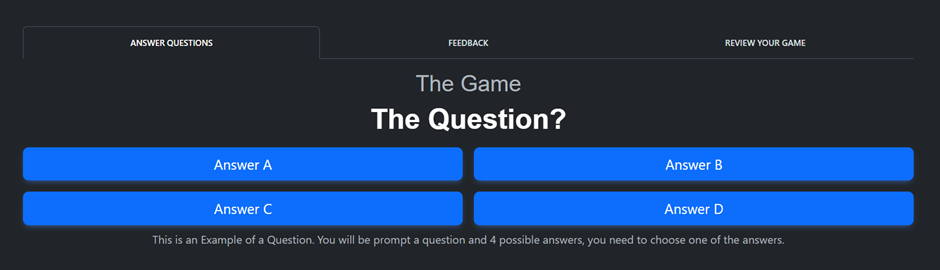
**How to play Screen:**

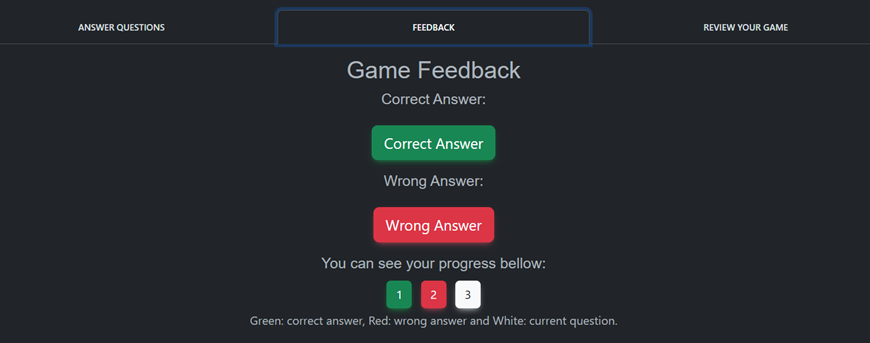
1. From the Home page screen, you can navigate to How to play screen.

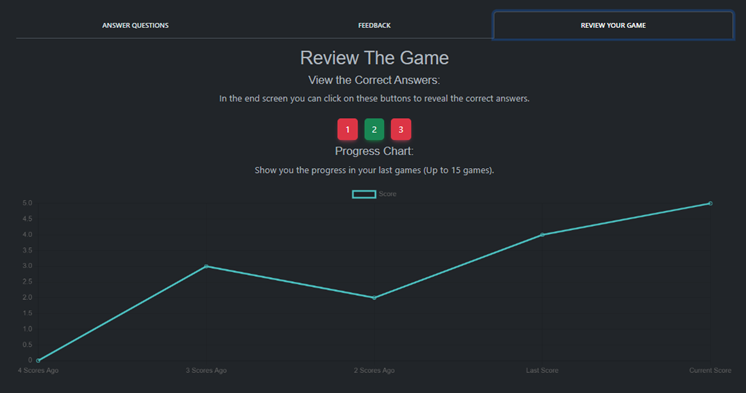
2. You can select "Answer questions" where the structure of the questions will be displayed.

3. You can select "Feedback" where the structure of the answers will be displayed.

4. You can choose "review your game" where the solutions to the questions will be displayed and whether you were right. And in addition a graph will represent your improvement trend in the answers in the last 15 games.

****

****

****

**Game Screen:**

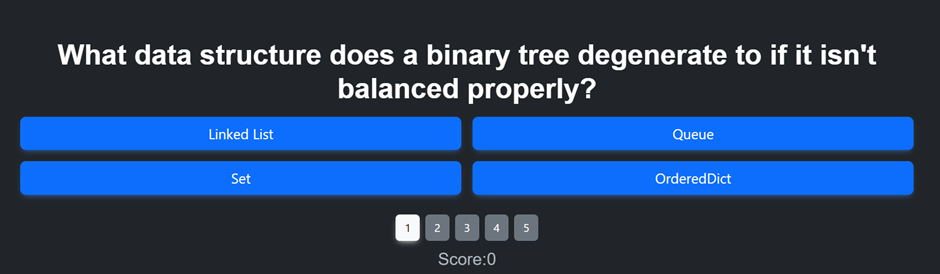
1. Once you have clicked on "Start Game" you will go to the game play.

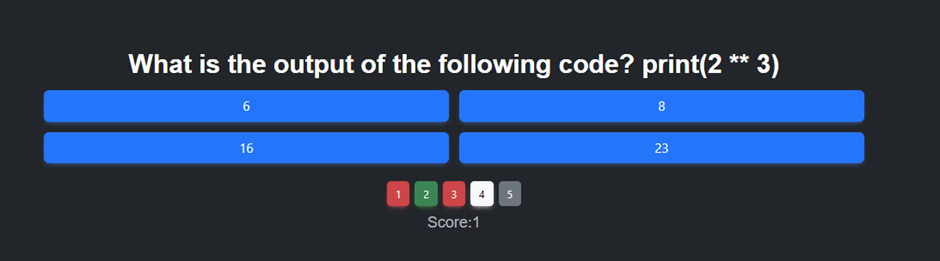
2. Began to answer questions.

3. Choose an answer from the 4 available options.

4. For each option you will receive an immediate answer.

5. You will then be transferred to the "View Results" page**.**

****

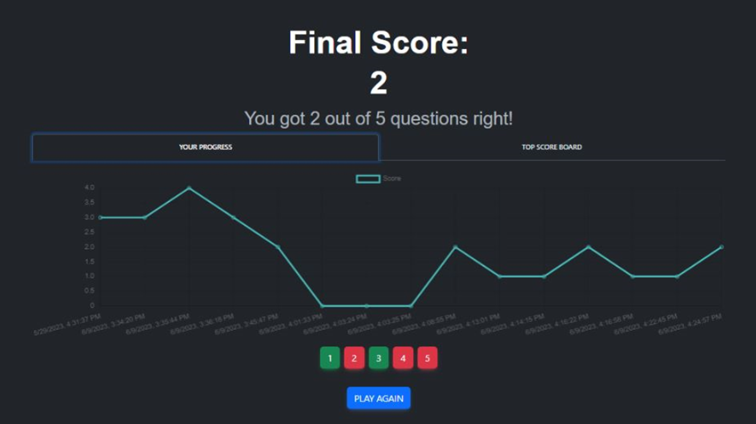
****

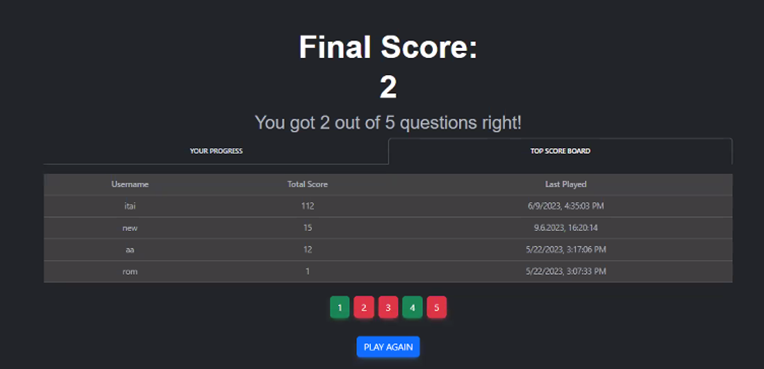
**Results Viewing Screen:**

1. After answering all the questions, you will be automatically transferred to a page that will show the result and the history of your results in the last 15 games or the leaderboard table in all the games.

2. You can click on the question number, see which answer you chose, and which is the correct answer.

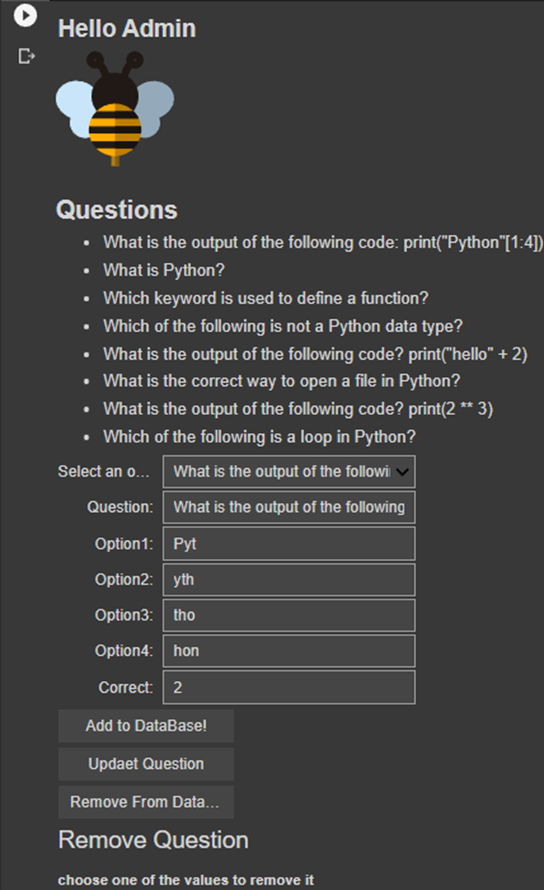
3. You can click "Play again" and go to the "game" screen again and start a new game.

****

****

**Admin panel**

In the admin panel you can add a question to the question pool, delete a question or update it**.** In addition, all the questions that exist in the database are displayed.  
  
 **Results Viewing Screen:**

1. You can select a question to display its information and update it using "Update Question" or alternatively delete it using "Remove from database".

2. Type the subject of the question.

3. Write 4 possible answers.

4. Add a correct answer using "Add to Data Base!".

**Maintenance Guide**

**Maintenance Guide:**

To use and continue developing the system, certain software and hardware requirements need to be met. Please note that the entire program runs on Google Colab. The system is developed using the Python language and JavaScript ,HTML.

**Software and Hardware Requirements:**

**Hardware:**

Computer or laptop or any other device with internet access.

**Software:**

Google Colab (for running the program)

Python (required language for admin panel and backend implementation )

JavaScript library (for backend implementation)

Jquery (for backend implementation)

Bootstrap  ( For styling )

MDBbootstrap  (For styling )

Widgets ( For admin panel interface )

HTML ( Display game )

Firebase library (for connecting to the Firebase )

Jinja ( For set template to insert everything into html )

**Database Environment**

Firebase Realtime Database.

**General Description:**

The website “Who wants to BEE a programmer “ is designed to provide an interactive learning experience for users who are interested in expanding their knowledge about python programming . The site features a trivia game with a wide range of questions related to python programming. It offers various screens, including the home screen, login screen, registration screen, screen,result and statistics screen, manager screen and tutorial screen .

The website caters to individuals of all backgrounds who are eager to learn about python programming. It is user-friendly and offers a gamified learning approach that makes the learning process enjoyable and engaging.

**External files:**

תמונה שמכילה טקסט, צילום מסך

התיאור נוצר באופן אוטומטי**Index.html - >**

* Defines the structure of the

page, contains all

dependencies on external

libraries, both logical libraries

and design libraries that

we used.

* FireBase
* Bootstrap
* MDBbootstrap
* Jquery

**תמונה שמכילה צילום מסך, טקסט

התיאור נוצר באופן אוטומטיStyle.css ->** Style the website .

תמונה שמכילה צילום מסך, טקסט

התיאור נוצר באופן אוטומטי

**Order of running the entire application:**

Click run all in collab .

**Scheme of our system:**

תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי

**Order of running the entire application:**

Click run all in collab.

**Scheme of our system:**

תמונה שמכילה טקסט, צילום מסך, תרשים, גופן

התיאור נוצר באופן אוטומטי

**Index.js**

**Function getFromDataBase(Folder, callback)** – This function gets "Folder", "callback" from a specified folder in the database and provides it to a callback function for further processing or utilization. It takes two parameters: folder, representing the location of the desired data within the database, and callback, a function that will be called with the retrieved data as an argument.

**Function modalShow(title,information**) – The function gets title and information and creates and displays a modal dialog box with a specified title and information. The function removes any existing modal, then dynamically generates HTML for the new modal using the provided title and information. The generated modal is appended to a container in the document for display.

**Function login()** – This function dynamically creates and displays a login form on a web page, allowing users to enter their credentials for authentication or sign up if they don't have an account yet.

**Function loginToggle()** – This function toggles between the login and sign-up modes in a login form by changing the text content of certain elements and animating the display of the "confirm-password" element. This allows users to switch between logging in and signing up with a single button click.

**Function check\_signup()** – This function handles the sign-up process by validating user inputs, checking for username uniqueness, storing the new user in the database, and return appropriate feedback to the user through modals

**Function check\_login()** – This function handles the login process by validating user credentials, logging in the user, displaying appropriate modals for success or error messages, and navigating to the home screen.

**Function homeScreen(username)** – This function gets username and dynamically generates the home screen interface of the application after a user has successfully logged in. It displays the user's avatar, a welcome message, and provides options to start the game or access the help page.

**Function startGame()** - This function prepares the game by resetting variables, shuffling the questions, selecting a subset of questions for the game, and then displaying the first question. It serves as the entry point for initiating the game and setting up the initial game state.

**Function showAnswer()** - This function is responsible for showing the correct answer to a specific question on the result page. It toggles the visibility of the answer container, updates its content with the question and answer, and provides a smooth animation for displaying the information. This allows users to learn from their mistakes and review the correct answers.

**Function showAnswerExample(index)** – This function get index, demonstrates the basic structure and behavior of a function that shows the question and its correct answer on the result page. It uses placeholder values to represent the actual question and answer, and it toggles the visibility of the answer container with a smooth animation.

**Function showQuestions()** – this function dynamically generates and displays the current question, answer options, progress buttons, and score on the game screen. It ensures that the UI is updated with the appropriate information for each question in the quiz.

**Function helpPage()** – this function generates and displays the help page for the game, providing explanations, examples, feedback, and visualizations to assist the user in understanding and navigating the game.

**Function endgame()** - this function handles the necessary actions and generates the HTML content for displaying the end of the game, including the final score, progress bar, correct answers, and options to start a new game. It also updates the user's data in the database and performs any necessary chart-related operations.

**Function startNewGame()** – this function hides the chart container, moves it to the correct location in the HTML structure, and then starts a new game by calling the startGame() function.

**Function showChart()** – this function checks the number of labels in the chart and either displays the chart or provides a message to the user indicating that more games need to be played to see the progress chart.

**Function createLineChart(PlayerData)** – This function gets player data, extracts the scores and dates, and creates a line chart using Chart.js to visualize the player's score progression over time.

**Function createExampleChart()** - This function creates an example line chart using Chart.js to showcase a timeline of scores. The chart displays the scores at different points in time, providing a visual representation of the score progression.

**Function addDataPoint(label,data)** – This function gets label and data and allows you to dynamically add a new data point to an existing line chart. It appends the label and data to the respective arrays in the chart's data object and updates the chart to display the new data point.

**Function shuffleArray(array**) – This function gets an array and shuffles its elements randomly, resulting in a new randomized order of the elements and return the array.

**Function checkAnswer(index,answer)** – This function gets index and answer and validates the user's selected answer in a quiz game. It updates the visual feedback for the chosen answer and keeps track of the user's score. After a brief delay, it proceeds to show the next question.

An admin panel, in Python language, uses widgets to add update and delete questions.

**Admin Panel**

The first part defines the connection to the DB and all the widgets תמונה שמכילה טקסט, צילום מסך

התיאור נוצר באופן אוטומטי

**save\_question(event):**

Gets event , Saves the question data entered in the input fields to the Firebase database. Prints a success message and updates the displayed questions triggered with event click .

**show\_questions( ):**

Retrieves questions from the database and constructs an HTML page with the question list. Updates the dropdown widget and displays the input fields. Return questions\_dict, question\_titles .

**dropdown\_eventhandler(change**)**:**

Gets change event , Handles the change event of the dropdown widget, retrieves the selected question's data from the database, and updates the input fields.

**update\_question(event) :**

Gets event that triggers her , Deletes the selected question from the database and saves the updated question data. Prints a success message.

**remove\_from\_DB(event) :**

Gets event that triggers her , Deletes the selected question from the database. Updates the displayed questions and prints a success message.

4) **האתגרים בהם נתקלנו בעבודה במהלך הסמסטר**

העבודה עם Collab, Python ו-HTML הציבה בפנינו כמה אתגרים ייחודיים. נדרש מאיתנו להסתגל לסביבת עבודה אשר לאף אחד לא היה אליה ידע מקדים. Collab, בהיותה פלטפורמה מבוססת ענן, הציג לנו ממשק וזרימת עבודה שונה בהשוואה לסביבות עבודה שהכרנו. היינו צריכים להכיר את התכונות של Collab, כמו ניהול מחברות, ייבוא ספריות וכו'. זה דרש מאיתנו להשקיע זמן ומאמץ בלמידה ובחקירת הפלטפורמה כדי למקסם את הפוטנציאל שלה.

בהקשר של Python, היינו צריכים לתכנן ולתאם בקפידה את התקנת הספריות, בקרת גרסאות ולתחזק סביבת פרויקט מובנית היטב כדי למזער קונפליקטים ולהבטיח שיתוף פעולה חלק. בנוסף, נדרש מאיתנו לשים לב לבעיות פוטנציאליות כמו שגיאות סוג ואופטימיזציה של קוד כדי להבטיח שהקוד שלנו יבוצע ביעילות ושהמשחק ירוץ ללא השהיות.

בעבודה עם HTML, אתגר משמעותי שעמדנו בפנינו היה יצירת ממשק ידידותי למשתמש. אתגר זה דרש שילוב של עקרונות עיצוב.באמצעות משוב של המשתמשים שקיבלנו לאחר הפעילות הכיתתית התאמנו יותר את השימושיות של המערכת לכלל

אתגר משמעותי נוסף שנתקלנו בו היה הצורך לשלב מספר רעיונות בפרויקט אחד. עם צוות מגוון, כל חבר הביא נקודות מבט וגישות ייחודיות לפתרון בעיות. הגיוון הזה אמנם היה מועיל, אבל הוא גם הציב את האתגר של שילוב רעיונות והבאתם לפתרון יחיד מגובש וטוב. כדי להתגבר על זה, דאגנו שתיהיה תקשורת פתוחה ושקופה, ועודדנו את כולם לחלוק את הרעיונות שלהם ולהקשיב באופן פעיל לאחרים. באמצעות דיונים משותפים ומשוב בונה, הצלחנו לזהות בסיס משותף, לתעדף את הרעיונות המבטיחים ביותר ולשלב אותם בממשק ידידותי למשתמש. תהליך זה לא רק הביא לתוצר סופי טוב יותר אלא גם שיפר את כישורי עבודת הצוות שלנו וחיזק את היכולת שלנו לשתף פעולה ביעילות.

כמו כן, היה קושי לקבוע זמנים ולהיפגש ולעבוד יחד על העבודה כל חברי הצוות, משום שלכל אחד יש גם אילוצים אישיים ולימודיים שונים.

**קישור לתיקיית הגיט של הפרוייקט:** [**https://github.com/RomRL/test**](https://github.com/RomRL/test) **קישור למסד הנתונים:** [**https://console.firebase.google.com/u/0/project/tir6db/overview**](https://console.firebase.google.com/u/0/project/tir6db/overview)