Maintenance Guide

Head in the clouds Game is a Python application designed for students to test their knowledge in the field of Cloud Computing. It's written using Jupyter notebooks on Google Colab, with a combination of IPython widgets for the user interface.

Database environment

The system links with Google Firebase via the firebase module in the Main file. For maintenance and updates to the Firebase database, the user requires Firebase Admin SDK access.

Special Requirements

1. **Google Account**
   a Google account is needed to log in to Google Colab. Once logged in, you can load the code into a new Python notebook.
2. **Firebase Connection**
   The code connects to a Firebase database. Google Colab does not store environmental variables between sessions. The code requires secure access to Firebase, uploading these credentials is needed each time a new Colab session is started.
3. **Internet Connection**
   Because Google Colab runs on the cloud, an active internet connection is required to run the cells in the notebook.
4. **Browser**
   Google Colab works best on the latest versions of Chrome, Firefox, and Safari. Ensure your browser is up to date for the best performance.

Main files and methods

1. Welcome Screen
   - **createWelcomeScreen()**: This function creates the initial welcome screen for the game.
2. Manager Screen
   - **createManagerScreen():** This function creates the Manager screen, providing options to Add, Remove, or Update quizzes.
   - **handleAddQuizButton(b):** This function handles the event when the "Add Quiz" button is clicked.
   - **handleUpdateQuizButton(b):** This function handles the event when the "Update Quiz" button is clicked.
   - **handleRemoveQuizButton(b):** This function handles the event when the "Remove Quiz" button is clicked.
3. Player Screen
   - **createPlayerScreen():** This function creates a Player screen where players can view their level and start playing the game.
   - **handle_start_play_button_click(b):** This function handles the event when the "Start Play" button is clicked.
4. Topic Selection Screen

- **createSelectionScreen():** This function creates a screen for players to select a topic for the quiz.
- **handle_topic_selection(button):** This function handles the event when a topic is selected.

5. Exit Buttons
   - **exit_selection_handler(b):** This function handles the event when the exit button is clicked on the selection screen.
   - **exit_player_handler(b):** This function handles the event when the exit button is clicked on the player screen.

Design Patterns

1. **Modular Design:**

The application is built using a modular approach, where each screen is a separate function. This allows easy maintenance, as changes to one screen do not affect others.

2. **Event-Driven Programming**

The game leverages an event-driven programming model. Actions in the game (like button clicks) trigger events which are then handled by specific functions.

3. **Functional Programming**

The application extensively uses Python's functional programming features. Functions are stateless, and output depends only on the input.

Maintenance Instructions

1. **Adding more quizzes**
   To add more quizzes, add them through the UI using the manager account or add quizzes through the database on Firebase.
2. **Changing the game layout**
   To change the game layout or styling, modify the HTML/CSS code in the functions.
3. **Updating the logic**
   To update game logic, identify the function responsible for that part of the game and make changes there.

Additional Notes

1. **Refactoring**
   The application has been built with functions for modularity. As the complexity grows, consider refactoring to classes for better encapsulation and readability.