Guess Who?

Design and implement a web application for playing a one-player version of the famous “Guess Who?” board game.

In the game, the player must guess the identity of a specific “item” amongst a wide set of items, by making a set of successive guesses. The gameplay defines a catalog of items, and items are characterized by a set of properties, which may assume different values. Each item in the catalog has a *unique* combination of values for its properties. All properties must be defined (i.e., have a value) for all items. The player can also add custom items by choosing their characteristics.

A catalog of items may cover any domain, such as famous people, cartoon characters, dinosaurs, constellations, pokemons, sports players, professors, animals, flowers, etc. You are free to use your fantasy, and the application should implement **only one** domain of your choice. The set of properties and their values must be customized depending on the type of items. The number of properties and values should be defined taking into account the characteristics of the catalog.
For example, if the selected items were wild animals, the properties could be “nutrition” (with values: carnivore, herbivore, …), “legs” (with values 0, 2 or 4), “flying” (values yes or no), “color” (with a wide range of values), etc.

The gameplay consists of a series of matches of various difficulty (easy = 12 displayed items, medium = 24 items, hard = 36 items). At the beginning of each match, the full list of possible items is displayed (by showing a picture for each item in a bi-dimensional graphical grid, pictures can be loaded as you prefer), and a “secret item” is randomly chosen by the server. The following two conditions apply:

1. To avoid cheating, the identity of the secret item must **never** be transferred to the client.
2. The set of values associated with the properties of each item should not be displayed in the grid, as the player should be able to understand them from the item’s picture.

Each match consists of a set of guesses. In each guess, the player will select one property from a list, and select one possible value of that property out of the possible value options. Upon confirming this selection, the application will tell the player whether the guess was correct or not (i.e., whether the secret item’s property value matches the player’s guess). The application will then ‘disable’ all the items on-screen that are incompatible with the guess.
For example, if the player guesses “flying=yes”, and the secret item is a flying animal, the application will confirm the guess, and will disable all non-flying items. If the secret item is not a flying animal, the application will communicate that the guess is wrong, and disable all flying animals.
Note: for any possible sequence of guesses, the secret item will always be in the set of still-enabled items.

To terminate the match, the player selects one of the still-enabled items (it can be done only once, at any time during the match, and it will terminate the match; it will be the only option if there is only one remaining item). If it corresponds to the secret item, the player wins, otherwise he/she loses.

At the end of the match, a score is assigned:

- 0 points if the player loses;

- an integer non-negative score if the player wins, computed as (n_items - n_guesses * k), where n_items is the initial number of items in the board and k depends on the difficulty (easy: k=1, medium: k=2, hard: k=3). If the difference is negative, a score of 0 is assigned.

In other words, each "wrong guess" makes you lose points from the initial total (i.e. n_items which depends on the difficulty). The final "select" does not count as a guess, and terminates the match.

From the main page of the application, any anonymous user can select a difficulty level, start the game, play the full match, and receive a score (that will not be stored nor remembered).

From the main page of the application, logged-in users can select a difficulty level, start the game, play the full match, and the result of their match is remembered and added to the user's personal history.

A logged-in user may access a page with the history of his/her played matches, with a list of **only completed** matches, showing for each of them: date, difficulty, secret item, score. Also the user's total score (the sum of all scores obtained by that logged-in user) must be displayed on the same page.

The organization of these specifications in different screens (and possibly on different routes) is left to the student.

Important note

- As mentioned before, the identity of the secret item must **never** be sent to the browser (except at the end of the game), therefore the APIs must be designed taking into account this requirement.