Time: up to 3 hours

\* Using Google is allowed

In the following exam, you will be required to solve various problems utilizing all sorts of technologies. Please make sure to spend your time wisely on each question (answer bonus sections only if you have finished questions fully), in case you are stuck it is recommended to move on to the next question.

**Notes**:

1. Please remember that even if you don't have any prior experience with the following concepts try to solve them in the best possible manner.

2. Please remember that it is vital for us to see what you have accomplished and the approach you have taken towards finding a solution.

**Requirements & Guidelines**

1. Choice of tools, frameworks, libraries, caching mechanisms, and architecture, etc. are completely free.

2. Attention should be given to clean coding best practices.

**About The Exam**

You are about to develop an “Dog talks” application.

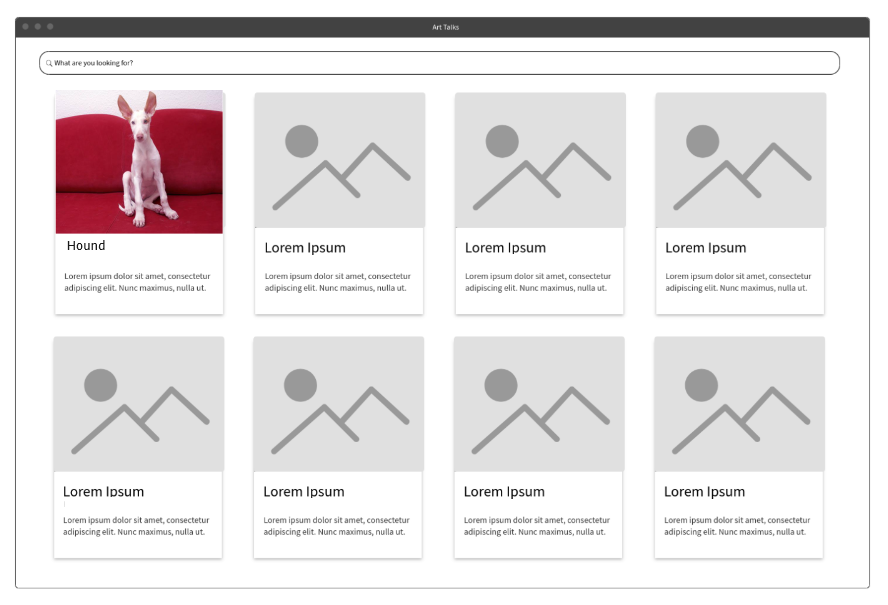
The “Dog Talks” is designed to display a Dog breeds image gallery and a discussion page about it.

**EXAM:**

Please solve the following tasks:

1. Dog breeds gallery homepage
   1. **Get the breeds images** from <https://dog.ceo/dog-api/documentation/>   
      please note, you should **create a list of hardcoded breeds to fetch**, or look at bonus question number 1.
   2. **Each breed should have its own card**. The picture cards should contain the **picture itself**, the Dog’s **breed name**, and a “lorem ipsum” **description**.
   3. **Search functionality – filter** breed pictures that contain the search term in their breed’s name.

**UI reference**



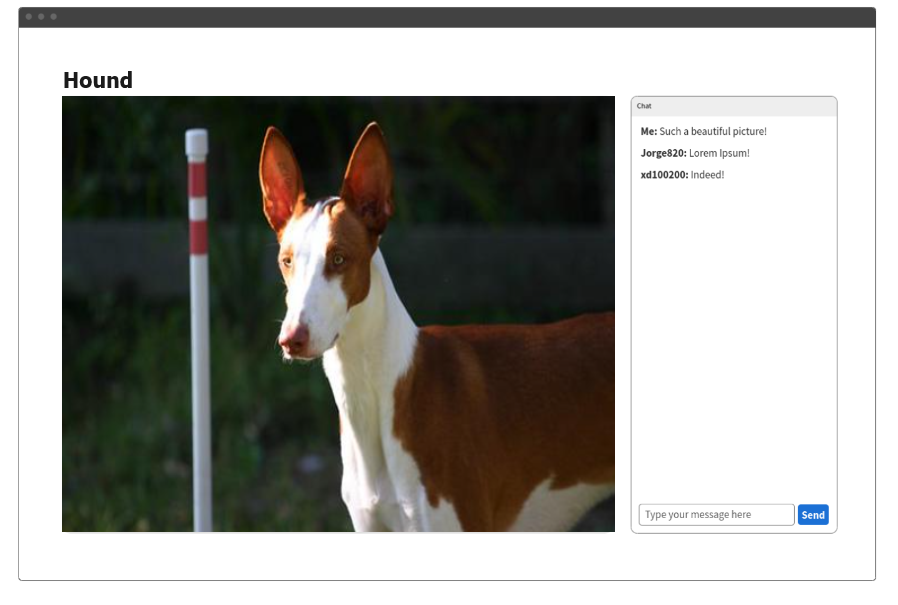
**2. Dog breed discussion page**

The page features are:

* 1. **Show the chosen dog breeds picture** (use the API to fetch one random one) **together** with its related data such as its **name** and lorem **description**.
  2. **Implement a chat interface** on the **right side** of the page that allows the user to **type a** **message and submit it**.

NOTE: this chat interface is not meant to be usable if you are not doing the bonus question. **It should be available to a user currently on the page**, they should be able to type a message, and **all the messages that the user specifically types should appear.**

**UI Reference:**



**3. Bonus**

* 1. Get the breed list from the api. Then use it to fetch and create a list of card for all the breeds.
  2. Implement Web Sockets, and in any new chat incoming message, send a Message with the Web Socket and update the UI accordingly. Note: this requires a backend.