

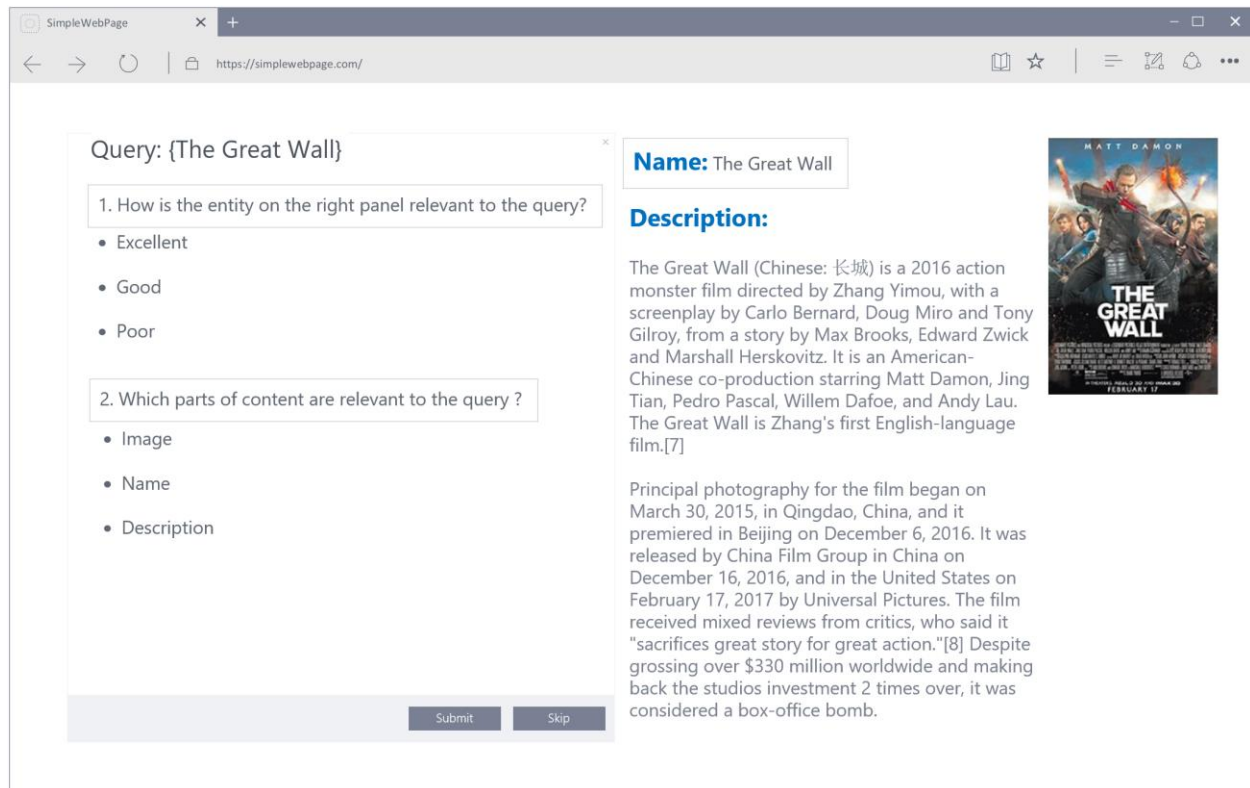
A Simple Web application

Introduction

This is a simple web application and we expect the candidates to complete the implementation and test it in web browser within 1 hour using HTML and JavaScript without any framework support, such as React or Vue.

This application consists of two panels. From the left panel the user will see at most two questions based on the user's answers; from the right panel the user will see the content of an entity. A user is expected to read through the content and answer the questions from the left panel based on her or his knowledge. A user can choose to submit or skip a query based on her or his confidence on the correctness of the answers.

A possible layout of the web page is shown as below:



The screenshot shows a web browser window with the URL <https://simplewebpage.com/>. The application is titled "SimpleWebPage".

Query: {The Great Wall}

1. How is the entity on the right panel relevant to the query?

- ☐ Excellent
- ☐ Good
- ☐ Poor

2. Which parts of content are relevant to the query ?


- ☐ Image
- ☐ Name
- ☐ Description

Name: The Great Wall

Description:

The Great Wall (Chinese: 长城) is a 2016 action monster film directed by Zhang Yimou, with a screenplay by Carlo Bernard, Doug Miro and Tony Gilroy, from a story by Max Brooks, Edward Zwick and Marshall Herskovitz. It is an American-Chinese co-production starring Matt Damon, Jing Tian, Pedro Pascal, Willem Dafoe, and Andy Lau. The Great Wall is Zhang's first English-language film.[7]

Principal photography for the film began on March 30, 2015, in Qingdao, China, and it premiered in Beijing on December 6, 2016. It was released by China Film Group in China on December 16, 2016, and in the United States on February 17, 2017 by Universal Pictures. The film received mixed reviews from critics, who said it "sacrifices great story for great action." [8] Despite grossing over \$330 million worldwide and making back the studios investment 2 times over, it was considered a box-office bomb.



Feature requirements

As shown above, the web page contains two questions that each one has three radio buttons, and two normal buttons, which represents the interactive part of the application. Other than these, the page

consists of three text boxes: Query, Name and Description; and one image which should be represented by a fake URI. Paragraphs below depict the feature requests of this application.

1. The right panel, i.e., the entity content, **is always shown by default** on loading of the webpage.
2. The first question is mandatory and **is shown by default** on loading of the webpage. There are three answers the user can and must choose one of them, based on the user's understanding how relevant the entity is to the Query. And each answer has score associated with it:

Excellent – 100

Good – 50

Poor – 0

It's required the application to store the score per user's choice to a JavaScript variable, let's name it **q1_value**. Please note this is a single choice question and the user must make a choice.

3. **If and only if** user select **Excellent** in the first question, the second question would be displayed. As the first question, each answer in second question has a score associated as below:

Image – 50

Name – 50

Description – 25

Likewise, the application must store the score in a variable, let's say **q2_value**. Please note this is a multiple choices question and the user must make at least one choice. The value of **q2_value** should be the summation of all chosen options.

4. Based on user's responses, the candidate should define another variable, let's say **sum**, whose value is the addition result of **q1_value** and **q2_value**. Please note that if user select the options other than **Excellent** in question 1, the value of sum should be equal to **q1_value**, because question 2 is not displayed for such cases.
5. After we have collected the sum value, we should call the provided API: **SendToServer**, to send the sum value to backend service. You should treat this API as given and do not speculate its implementation. Ideally, this API call should happen in the event function of the button **Submit**.
6. If user clicks **Skip** button, the web page should reload itself and reset all user's inputs.

Validation checkpoints

To control the answer qualities, there are following validation check logic should be enforced in the appropriate event functions:

1. If user clicks **Submit** button, you should check if the user has made valid selection by examining if all displayed questions have been answered.
2. Make sure user cannot select more than one option for question 1, and can select more than one for question 2.
3. Validate the *sum* value is within valid range if user clicks **Submit**. For example, it must be a non-negative value and cannot be great than 225.

Functional Tests

The candidate should test the application in a real web browser such as Edge or Chrome, to ensure it works properly as required above smoothly without any crash or noticeable latency.