Technical Test – Web Developer

This test has two parts and its main goal is to evaluate the knowledge and skills in web development and problem solving of the applicant to the position.

In the first part of this test, you will find two problem statements that you will need to solve in order to show your problem solving skills. In the second part you will find a short description for a Web application that needs to be built. You can use the language, framework of your preference to implement every solution.

Part I: Problem solving

NOTE: The most important thing about this first part of the test is that you as a developer prove your skills to build effective algorithmic solutions to specific problems.

A. An anagram is a type of wordplay, the result of rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once; for example *Doctor Who* can be rearranged into *Torchwood*. Any word or phrase that exactly reproduces the letters in another order is an anagram.

   Write a small application in the language of your preference, which reads a file containing an input dictionary and determines which words within that file are actual anagrams.

   The file contains a list of words, for example:

   Laid
   Better
   Dial
   Times
   Brain
   Items

   You have to read this file and check which set of words are anagrams. Based on the previous list, the output should be:

   Times, Items

Dial, Laid

The file will be provided along together with this document.

BONUS: Submit tests along the results, validating methods and functions used in the code.

B. An ananagram is a word or phrase that has no anagrams, this is, no other word or phrase can be generated as a result of rearranging the letters of it, using all the original letters exactly once; for example *Sun* can not be rearranged into another word.

   Write a small application in the language of your preference, which reads a file containing an input dictionary and determines which words within that file are ananagrams (or have no anagrams).

   The file contains a list of words, for example:

   Laid
   Sun
   Times
   Items

   You have to read this file and check which set of words are ananagrams. Based on the previous list, the output should be:

   Sun
   Laid

   The file will be provided along together with this document.

   BONUS: Submit tests along the results, validating methods and functions used in the code.

Part II: Web Application

NOTE: The most important thing about this second part of the test is that the application you build and the language / framework you choose, help you show your development skills, including:

1. The right use of OOP (Object oriented programming)

2. MVC pattern use, to properly split apart business logic from user interface
3. Observer design pattern to check the prize (in case you opt to use it)

Build an application using Web technologies to assign prizes to specific visitors of a page after they subscribe

A small company wants to build a small web application for one of its stores, where visitors are able to enter their email and subscribe to get information and updates about the store's products.

The web application's root page, will display the title of the store and a small paragraph with information. Also, it should have an input field for the email and a subscribe button. Visitors will land to this page to subscribe and get prizes.

The process and rules for subscribers are:

1. Visitors should type their email in the landing page and hit 'Subscribe' button. If that subscriber wins a prize, then the page should notify immediately.
2. Subscribers can enter their email once a day to win prizes. If an email is entered two times the same day, the app should notify that this email was already registered that date.

On the other hand the web application should also have an admin site. Users with valid credentials will be the only ones allowed to enter this site (You will need to provide a default admin user to access the admin site). For the admin site, you should also be aware that admins:

1. Should be able to manage the prizes. Create, edit, and delete prizes.
2. Register an inventory of prizes, number of existences for each prize.
3. Need to setup the conditions to win the prizes. Each condition must be associated to an existing prize, which will be automatically assigned to the subscriber and will decrement its stock in the inventory. Conditions should be easy to setup. Some examples are:
    a. Wins prize specific subscribers. Subscriber #50, #100, #2000, etc
    b. Wins prize subscribers multiples of 500.
    c. Wins prize subscribers multiples of 50 after 1000 subscribers.
    d. In the case that two conditions overlap, the app should assign one prize for the subscriber, and the other to the immediately next subscriber, e.g. the conditions above (b) and (c) will overlap for a subscriber #1500, #2000, etc. In this case wins the subscriber #1500 for rule (b), and #1501 for rule (c) (rules ordered by creation date).
    e. Be aware that more than two rules can overlap at some point. For example, for subscriber #2000 rules (a), (b) and (c) overlap. In this case subscriber #2000

wins the prize for rule (a), subscriber #2001 wins the prize for rule (b) and subscriber #2002 wins the prize for rule (c)