APPLE INDUSTRIES

Code Sample

Overview

To assess a candidate's experience working in a professional software engineering setting we ask that a code sample be provided for review prior to a final interview (and hopefully onboarding a new team member!). The challenges contained herein are admittedly contrived novelties but should nonetheless be approached with the same degree of execution you feel represents production-ready software. You are free to complete the challenges using any of the listed languages (see "Approved Languages" below). You are expected to provide both the source code as well as instructions for executing the sample (or, for QA Engineers, testing the provided sample); you should assume the reviewer of your sample has a basic understanding of the language used.

Approved Languages

Position Title*	Languages/Frameworks
Desktop Application Engineer	C#
Backend Engineer	Java, C#
Frontend Engineer	Javascript (or Typescript) / Node.js, Vue, React, or Similar
QA Engineer	Any of the above

**Position-specific requirements are listed at the end of this document.*

Sample Components

The sample provided is comprised of three (3) primary components outlined in the following sections. Do your best to approach each scenario as if each was a realistic project requirement. Be creative, be yourself, do your best to show us how you see good software development.

Challenge 1: Mirror, Mirror on the Screen!

Apple Industries is working with an exciting new reflective display screen in our latest line of photo booths. While the screen itself is cool, its reflective qualities mean we need to build our displays backwards, including the text! To accommodate this, we'll need some software that can take a sentence and reverse the order of the words.

Challenge 2: I'm feeling lucky!

At Apple Industries it's important to remember that we sell fun: you and your friends climbing in one of our mega booths for some silly group scenes, hanging out in the movie theater lobby and being part of the movie magic, making your own memorabilia at a sports stadium or theme park. We also like winning prizes! Our booths have three types of photo packages: prints (4x6 photo) for $5, panoramas (6x12 prints) for $7, and strips (two 2x6 photo strips) for $5. Once per hour at most, we want one customer that orders each package type to have a chance to receive the other two packages of prints for free. To do that, we'll need some software that knows what package the customer ordered, determines if that customer's order is eligible to win, and then if they in fact won the free prints.

Challenge 3: Only two things in life are guaranteed…

Once per calendar month Agent Smith of the Internal Revenue Service (IRS) wants to know how much tax we owe them. For now, since our headquarters is in Long Island, New York, the IRS only requires us to pay the local sales tax rate of 8.625% no matter where the customer paid to use our photo booth. Luckily, we don't need to pay taxes on the packages we gave away for free. To make this easy, our accounting department will need some software that can add up all the order revenue collected in a photo booth by calendar month and determine how much tax we owe the IRS.

***Desktop Application Engineers* – please present your solution as a Windows desktop application, using either WPF or WinUI3. This desktop application will be used by our accounting department to accomplish the stated task.**

***Frontend Engineers* – please present your solution as a browser-based-application, using your framework of choice. This application will be used by our accounting department to accomplish the stated task.**

QA Engineers* – a sample application that is supposed to accomplish the above challenges has been provided in the format of a Docker image. Your task is to test this application according to your best interpretation of the requirements given in the challenges above. Instructions to run the provided sample application, as well as info for the API specifications, are included in the README.