Code Challenge

Coding Challenge: PhoneBuzz

Note: we know your time is valuable and we appreciate you working on this project for us to demonstrate your skills and knowledge. We don't expect you to spend more than 3-4 hours on this challenge, so simply get done whatever you can and if you don't finish, that's fine (many candidates complete only the first couple of phases below).

PhoneBuzz is an implementation of FizzBuzz over the phone. For those unfamiliar with the game of FizzBuzz, it works like this:

- Players generally sit in a circle. The player designated to go first says the number "1", and each player thenceforth counts one number in turn. However, any number divisible by three is replaced by the word *fizz* and any divisible by five by the word *buzz* . Numbers divisible by both become *fizz buzz*.
- A typical round of fizz buzz is as follows: *1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, Fizz Buzz, 16, 17, Fizz, 19, Buzz, Fizz, 22, 23, Fizz, Buzz, 26, Fizz, 28, 29, Fizz Buzz, 31, 32, Fizz, 34, Buzz, Fizz, ...*

The goal of these challenges is to test your ability to write working and correct code, create an end-to-end web application, host it on a server, interact with APIs, process requests, generate responses, and handle various edge cases such as input sanitization and error handling.

As the phases progress, the application will become more complex. To complete each phase, submit a set of URLs to your hosted applications endpoints and a link to the source code (via github, bitbucket or any other hosted source code repository). We might want to run your server local and sub in our Twilio account credentials to test your application, so please also include some way to build/run your app in a README file.

- Phase 1: Simple TwiML PhoneBuzz

Requirements

Your task is to create a web application that will respond with TwiML that asks a user (over the phone) to enter a number, then responds to the input with TwiML that reads back the result of FizzBuzz up to that number. The end result is that we should be able to point a Twilio phone # at your web app and be able to play FizzBuzz via the phone. This does not require a Twilio account, use of the Twilio API, etc. You just need a web application that generates TwiML. When we point a Twilio number at your app and dial the number, the following should occur:

- A voice prompt indicates for the caller to enter a number.
- The correct output of the FizzBuzz game (starting at 1 and ending with the entered number) is read back to the caller.
- Be sure the TwiML portion of your application correctly validates the *X-Twilio-Signature* header, so that only Twilio is allowed to interact with those URLs.

The Rules

- Feel free to use any programming language you wish (though we'd prefer it wasn't Brainf***).
- Feel free to use TwiML generation libraries in your language of choice.
- **If there is a FizzBuzz library for your language (e.g. the fizz-buzz gem), you are not allowed to use it.** You must write the actual fizzing and buzzing yourself. ;-)
- We suggest you sign up for a free Twilio account so that you can point a phone number at this TwiML and test it out.

- Phase 2: Dialing PhoneBuzz

Requirements

Your task is to augment the web application so that now there is an interface which allows a user to initiate an outbound phone call to a user using the Twilio API which then plays PhoneBuzz as in phase 1.

HItting your web application should now load a page by default that contains a form where a user can enter a phone number and a submit button. Once the user enters their phone number and clicks submit, the following should occur:

- Your application dials the entered phone number.
- When the caller answers, a voice prompt indicates for the called user to enter a number.
- The correct output of the FizzBuzz game (starting at 1 and ending with the entered number) is read back to the caller.

The Rules

- Same deal as phase 1; you pick your language.
- Feel free to use Twilio client libraries in your language of choice.
- You probably won't be able to call anyone who isn't whitelisted if your Twilio account is a free account, which is fine.

- Phase 3: Delayed PhoneBuzz

Requirements

This is similar to the simple FizzBuzz challenge, however, this time we want to add a delay to the phone call. The first URL page should now present the user with two inputs, one for a phone number and one for an amount of time. Upon submission, the phone call should be scheduled for the amount of time indicated into the future (e.g., if I enter "1 minute", my phone should ring one minute from now). The rest of the PhoneBuzz program should continue as normal.

The Rules

- It is up to you to decide the format in which you want to accept the time interval, and parse it appropriately.
- You can elect to use scheduling libraries/frameworks for your language of choice, or to write something simple yourself.

- Phase 4: Tracking PhoneBuzz

Requirements

The next step is to add a storage system to your application. You should add to your delayed PhoneBuzz application a history of PhoneBuzz calls made, when they were made, what the delay was, and what number was entered by the user. You should then display this history below the inputs on your home page, and add a button on each line to allow for the replay of any call. When a call is replayed, it should dial the same phone number, but it should not request an input number over the phone; instead, it should immediately start reading back the result of the FizzBuzz game using the input that occurred in the previous call.

The Rules

- You can use any storage system you like.
- Replay calls should be included and tracked as well.