Competency Evaluation

Software Developer Apprentice

Developed by AlbanyCanCode Inc. Funding Provided by State University of New York and SUNY Schenectady

Rationale

Overview

This document is broken down by the major Work Processes outlined in Appendix A of the Software Developer Apprenticeship Standards. It provides both explanation/justification for various decisions in how to assess apprentices, as well as providing a core written assessment tool. This assessment can be further adapted to different platforms that allow for automated testing of coding sections and online administering of the evaluation. Care has been taken to make the assessment programming language agnostic to account for the many different job placements that apprentices may have.

Workplace Basics

The Workplace Basics standards A1 - A5 would be best evaluated by the employer hosting the apprentice. While not an assessment in and of itself, testimonial from employers would likely be the most accurate indication of employees workplace skills. These standards could also be assessed by standard workforce training certifications that are available, or they could be assessed via an essay prompt.

Defining Projects/Designing Software

Apprentices are asked to respond to several prompts and from them identify key requirements, user actions, and user stories based on the project descriptions. There is no mandatory structure to their response - however, their response should be easy to understand, regardless of their chosen approach.

From the same prompts used for Defining Projects, apprentices are asked to translate those user requirements into technical requirements. These technical requirements should focus on necessary hardware considerations, choosing and justifying appropriate data structures, and planning to mitigate potential security issues.

Writing Software Code

Because different apprentices have likely learned different programming languages, the following questions have been designed to be as language-agnostic as possible. This also means that the apprentice *must declare the language they would like to answer in* before going through this section.

Earlier questions are intended to test fundamental syntax knowledge in the apprentices chosen language, and address Standards D1-D2, D10, D12

Later questions will ask apprentices to write small code samples. These exercises have been chosen to allow apprentices to demonstrate the array of skills expected in Standards D2-D9, D11 in a more practical way that attempting to assess each individually. For example, the Fibonacci Exercise will likely assess:

- Arithmetic Operations
- Variables and Assignment
- Value Comparisons
- Logical Truth
- Loops
- Conditionals
- Custom Functions

Similarly, the Sorting Algorithm exercise will evaluate many of the above topics, as well as determining the apprentices familiarity with other computer science principles such as computational complexity even if they are unfamiliar with the academic definitions.

Questions are designed to be assessed in the order provided in the Assessment section.

In regards to implementing the technical portion of the assessment, apprentices should be allowed time to work on the code in a reasonable development environment. A reasonable development environment should include:

- Common tools, such as code completion and syntax/error highlight
- Ability to test code, modify, and test again
- Access to standard libraries for the language of choice, such as Math libraries

A resource such as https://repl.it/languages would be an example of a reasonable development environment capable of being used for a large range of common programming languages.

Building GUI/Applications

Due to the broad range of possible placements for apprentices, and how much the process of building graphic interfaces varies between different languages, Standard E1 is difficult to assess in this format. It very much depends on the specific software work an apprentice has done (for

example, a developer working in SQL databases may never work in developing a graphic interface in those projects).

Standard E2 is valuable across all project types, and across all levels of skill. Assessment focused on determining how the apprentice approaches testing their applications, and if they are aware of and concerned about the issues that can occur when their application is deployed on different platforms.

Testing, Deployment, and Maintenance

Due to the broad nature of Standards F1-F3, it is difficult to use questions to assess them. That said, Standard F3 can be combined with Standard A5 to assess the apprentices ability to simplify and explain technical concepts. By providing students with a wide range of technical concepts to explain in layman's terms, we can both keep apprentice to a particular level of complexity and avoid having the question misalign entirely with their apprenticeship.

Assessment

Defining Projects/Designing Software

Writing Prompts

Read through the following descriptions of different systems. Based on these descriptions, develop and describe:

a. A list of customer requirements for the system/product.
b. A list of potential actions users may take.
c. A list of technical or system requirements, such as how data will be structured.
d. Any potential hardware requirements or limitations.
e. Potential vulnerabilities in the system you describe.

Your lists do not need to be comprehensive given time constraints, but a recommendation would be 10 items. They should be self-consistent, and proposed technical/system requirements should be realistic. Hardware and vulnerability concerns should be justified in some way.

Tic-Tac-Toe

> A client has asked you to make a Tic-tac-toe game for them. At this time, they haven't specified where they would like the game to be able to run - in the browser or as a desktop application. A game of tic-tac-toe consists of the following:
>
> **1.** The game is played on a grid that's 3 squares by 3 squares.
>
> **2.** You are **X**, the opponent is **O**. Players take turns putting their marks in empty squares.
>
> **3.** The first player to get 3 of her marks in a row (up, down, across, or diagonally) is the winner.

4. When all 9 squares are full, the game is over. If no player has 3 marks in a row, the game ends in a tie.

Automated Teller Machine

Base your requirements on the following user experience:

"It's payday, and Joseph needs to deposit his paycheck with his bank. He ended up leaving work late today, and can't make it to the bank before it closes. He needs to pay his upcoming bills, so he chooses to use the ATM instead.

He identifies himself to the ATM using his bank card, and then inserts the check into the machine. He confirms the amount of the check is $1000. He then specifies he wants to transfer $100 from checking to saving, and would also like to withdraw $50 in cash from checking. He indicates that we would like to get the current balances, and would like printed record of this transaction. He takes his $50 cash and transaction record which includes the current balances on his accounts."

Programming Knowledge & Skills

1. List and define 3 primitive data types that are consistent across languages.
2. What are some limitations of the different number data types?
3. In <LANGUAGE OF CHOICE>, perform the following conversions/type casts:
    a. Integer to String
    b. String to Integer
    c. Integer to Float
    d. Float to String

4. **Fibonacci Numbers**

The Fibonacci numbers are the numbers in the following integer sequence:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……..

Starting with the numbers 0 and 1, the next number in the sequence is the sum of the preceding two. For example:

```
0 + 1 = 1 #Third number
1 + 1 = 2 #Fourth number
2 + 1 = 3 #Fifth number
etc...
```

In <LANGUAGE OF CHOICE>, write a program/function that does the following:

a. Prints out the n-th Fibonacci number. For example, the 10th Fibonacci number is 55.
b. Make sure to consider and handle possible errors.
c. Discuss the memory use of the program you wrote. This does not need to be in great detail, but you should address the limitations of your program.

5. **Custom zip() Function**

The zip() function in several programming languages takes two or more lists, arrays, or other iterable objects, and combines them into a new list. With two lists, for example:

```
list1 = ["a", "b"]
list2 = [1, 2]
newList = zip(list1,list2)

print(newList)
>>> [ ["a", 1], ["b", 2] ]
```

a. In <LANGUAGE OF CHOICE> write a custom zip() function which takes only two lists as arguments, and returns the zipped result as shown above.
b. Make sure to consider edge cases, such as the lists having different lengths.

6. A variety of algorithms exist for sorting lists of data. In <LANGUAGE OF CHOICE>, implement a sorting function called ***sort()*** which takes a list of integers and arranges them in ascending order (smallest to largest). An example using Python's sorted() function is below:

```
unsorted_list = [45, 3, 1, 7, 2, 67]
sorted_list = sorted(unsorted_list)

print(sorted_list)
>>> [1, 2, 3, 7, 45, 67]
```

   Assume that the list of numbers to be sorted is a random list of integers from 0 to 1000, with each integer being equally likely. You do not know the length of the list.

   Discuss the performance of the algorithm you wrote, in terms of memory use and performance time.

7. In <LANGUAGE OF CHOICE>:

   a. Write out a small script importing additional libraries, modules or functions. You do not need to refer to actual libraries/modules/etc.

   b. What are some potential concerns when importing external libraries?

Testing and Quality Assurance

1. From your experiences, describe how your workplace handles testing and quality assurance of projects.
   a. Do they follow a particular methodology?
   b. What has your role in the process been?
   c. Why is code testing and quality assurance important?
2. What possible issues are there when supporting an application on different platforms? How can these issues be addressed or handled in an example scenario? This could be different desktop operating systems (if developing desktop apps), different mobile operating systems (if developing mobile apps), or different web browsers (if developing web apps).

Communicating Technical Ideas

1. Select a technical concept from the following list[1]:
    a. RESTful APIs
    b. Relational Databases
    c. Encryption
    d. Machine Learning/AI

    As best you can, describe that concept using minimal technical terminology. For example, describe the concept as if you were discussing this with a client who has no education in programming or computer science. You can assume a basic level of computer knowledge with a common operating system.

2. Describe the software development lifecycle as it applies to the work you have done, such as if your work follows a waterfall or agile methodology, or if it is a hybrid system. Your answer should address potential benefits and shortcomings of the methodology.

[1] This list can be modified to allow for a more diverse pool of apprentice background and job experiences.