Programming Assignment – Apple Siri Team

Ryan Lacerna

This solution is implemented using .NET core and C#.

Time spent 3.5 hours

**How to run:**

Install .NET core version 2.1.503 SDK:

<https://dotnet.microsoft.com/download/thank-you/dotnet-sdk-2.1.503-macos-x64-installer>

Open your terminal:

1. Navigate into the /Rhymes-master folder
2. dotnet restore
3. cd ConsoleApp1/
4. dotnet run
5. The Console will then prompt you to enter a word, it will then return the matches or no matches result.

**Project Structure:**

Classes**:**

* **Startup.cs** – Contains the main method to run the program, and the registration of our dependencies in a service container
* **Data.cs –** Contains the collection of words we need to compare with the user inputs
* **WordAnalyzer.cs –** Contains all the functions to process the words and find the rhyming matches

We have two interfaces:

* IData
* IWordAnalyzer

These allow us to implement dependency injection in our program to help loosely-couple our classes. This pattern also allowed us to easily test our functions independently.

Tests:

Inside the /**Siri.Tests** foldercontains our unit tests. This created using NUnit and Moq frameworks. The tests coverage currently is very light, and not all scenarios and functions are covered due to the assignments’ time constraint.

**Assumptions:**

* This program assumes that there is only one user tat uses the program, Therefore, most of our functions as synchronous. If it were API service, where millions of people are using, the functions needs to be asynchronous.
* The program assumes that the user only inputs one word at each time.
* Currently we only check for the similarity of letters at the ending of each word. But other complexity such as rhyme sounding words such as “Men” with “Again”, “Rule” with “School”, “Snow” with “Go”; should be addressed also.

**Aspects:**

* **In terms of accuracy** – Currently, we are only checking for nulls and if the user input word already exists in the collection**.** Ideally we should sanitize the user inputs for upper-case/lower-case, special characters, two-word inputs, spellings and others that may affect the accuracy and security.
* **In terms of scalability –** Although we are currently processing all collection of words in parallel; if we need to processes millions of word entries, it’s best to divide the collection of words into multiple sub-collections and process each sub-collection in parallel. Ideally in a distributed environment like the cloud.
* **In terms of reliability –** More unit tests should be added to cover all program functionalities and various scenarios to provide better confidence of reliability of the program. Currently, we only provided a few unit tests. Ideally, more should be added. Exception handling should be properly implemented and logging.