Life Generator

Alvin Johns

Contents

****

1. Introduction

1.1 Purpose

The purpose of this SRS document is to describe the requirements. specification for a microservice that generates plausible aspects of a persons' life based on input data.

1.2 General Description

Given input data from Craigslist, the Life Generator will formulate. conclusions about aspects of a persons' life (e.g., their job description, their home, their vehicle, items they rent or own, etc.).

1.3 Scope

This document serves only to generate plausible aspects of a person's life. This document does not attempt, in any away, to collect specific information about a person using the input data. When data is used, it will be assigned a subject number to produce speculations about a person's life, not facts.

2. Requirements

2.1 Functional Requirements

- As a developer, I will need to provide an API so that other users can access the microservice.

- As a developer, I will need to have access to external APIs to gather information to use within the microservice.

- As a user, I will need to view information about a subject through a user interface.

- As a client, I will need access to the formatted data.

- As a developer, I will need to create the database to store subject data.

2.2 Non-Functional Requirements

- The microservice API that is created should be agnostic to the language being used to access it. The API should return data that can be interpreted however the end-user chooses.

  Quality Attributes: Accessibility, Interoperability

  Justification: If other services are not able to utilize the microservice, the chances of the microservice growing into a valuable product shrink. Thinking about popular APIs such as those provided by Google, PassportJS, etc., these all provide APIs that welcome their use while not restricting what the end-user chooses to build with those services.

- The user-interface of the microservice will be simple and intuitive.

Documentation will be provided that will cover the purpose and use-cases of the microservice. The documentation will provide details about the APIs and the functions they perform.

Quality Attribute: Memorability

Justification: If end-users are not able to determine what steps they are to take to get data from the microservice, or they do not have a place where they can search for answers, the likelihood that the end-user will use the service is small.

- The microservice should produce results about plausible aspects of a subject's life in under 10 seconds. During this duration, the user should be given feedback about what stage the software is at in the process.

  Quality Attribute: Efficiency, Negotiable

  Justification: If the user is asked to wait more than 10 seconds for the results, a few choices could be made. The user may assume that the service is not working and keep trying - which would produce the same wait-time. The user may choose to keep waiting but begin to assume that the microservice is not efficient or of high-quality. The user may also choose to not use the service altogether because of the relatively long wait-time.

- The database and microservice will active >99% of the time.

  Quality Attributes: Integrity, Reliability

  Justification: If the service is frequently down, the end-users are less likely to return. This is analogous to stores that are always closed when you need them. The chances that you will keep returning to that store to check if they are open goes down when there are more reliable options available.

- The provided formatted data should be in JSON format.

  Quality Attribute: Interoperability.

  Justification: JSON is a format that is language agnostic. It can be interpreted by many forms of programming languages.

****