Software Requirements Specification

for

Vocational test

Version 1.0 approved

Prepared by Grupo 1

Analisis y diseño de sistemas

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

In this SRS, we will be presenting our product, a system related to the making and taking of vocational tests. This will describe requirements for the entire system.

What is a vocational test?

A vocational test presents a list of different questions that an user answers in the way they feel, and using the answers to said questions, the test then tells them which career path is the most suited for them. These tests are very commonly taken by would-be tertiary education students that haven't yet made up their mind about what they enjoy most.

1.2 Intended Audience and Reading Suggestions

This SRS may be read by all types of people related to this product. We will describe the many functions and characteristics the completed software system should have. The introduction, as its name implies, just has the objective of introducing a reader to the product. The overall description starts getting a bit more specific in what the product does and requires. The external interface requirements lists requirements that are more related to how the product interacts with the user. System requirements will go deeper into exactly what functions should our system have.

1.3 Product Scope

The product whose software requirements we're specifiying in this document is a system that enables our client to create and modify tests, and share them with users so that they make take them.

Our software is a system that enables our client to create and modify a vocational test, and a user to take it and get a relevant result. We have already described what a vocational test is (section 1.1). Our software should provide all the functionalities needed for our client to create an accurate vocational test, and for different users to fill them out and get a result that's been calculated specifically for them.

2. Overall Description

2.1 Product Perspective

Our product is a new and self-contained one. We plan to make this system from scratch.

2.2 Product Functions

The functions our product must perform are, broadly, letting its owner create, modify and delete tests, and share them with users that might complete them. When the answers to the test are submitted, an appropiate result relevant to the user will be returned. Refer to appendix B to check out a class diagram.

In our class diagram, the class question is every question of the vocational test, it has a description, which is in the form of a statement that the user either identifies or doesn't identify with.

The class choice, is the particular answer that the user chose for a particular question, in our software the answer is given in the form of a numeric value that represents how much the user identified with the statement in the question.

We then have an outcome, which connects choices and careers together. This is so we can determine how every particular choice relates to every career when it comes to calculating the final result. Our outcomes have a weight attribute, which is used to make our choice matter more or less for a certain career.

The last class is survey. A survey is the combination of a certain user, and the career they got as a result. This class functions as a history of every test taken. Do note that in our system, registering a survey at the end of the test is optional for the user.

2.3 User Classes and Characteristics

Primarily, there should be three user types:

1- The administrator, administrators should be able to modify the test selection as needed. Administrators only have privileges regarding tests and their contents.

2- The regular user, users should be able only to take the test and receive the appropiate results. They have the least privileges, and might not even need to have an account.

3- The manager/s of the system, managers are administrators that can also appoint other administrators or remove them, as well as perform rollbacks. Managers are the most important user class

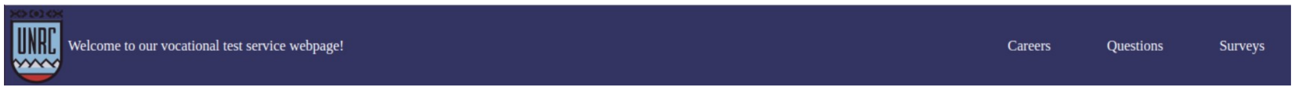
However, due to time constraints, we might not have the time to implement this.

3. External Interface Requirements

3.1 User Interfaces

We are using a combination of HTML and CSS languages to create our user interfaces.

Elements:



We use the above header as a way to quickly navigate our page. The welcome message acts as a return to homepage button. Careers redirects to a list of all available careers from where careers can be added or deleted, or one can see information about a career. Questions redirects to a list of all questions of the vocational test, which provides add/delete functionalities, and if you access the URL of a certain question, you can create, delete or modify outcomes for a certain question from this page. Surveys contains a list of all surveys that have been taken, with the username of the user and the career returned.

3.2 Software Interfaces

We are running this project on linux, with all our required applications running as docker containers. We are using ruby version 2.6.4 for code-related purposes, puma 4.3.3 for webserver, sinatra 2.0.7 for our webapp, and the latest versions of postgres for our database, sequel for interacting with our database from our page, and rails for similar purposes to sequel.

Docker:

Docker loads many containers with applications on its engine, without needing to use a separate OS for each container. This makes it more efficient, as turning operation systems on and off, and running them, is costly. If we need persistent data, docker uses volumes to load it.

Ruby:

Ruby is a high level language that supports multiple paradigms. Ruby has duck typing, which means we often don't need to specify a type, if an operation can be performed on an object then it is performed. Everything is an expression in ruby.

Sinatra:

Sinatra is a language we use to make HTTP interactions with our webapp way simpler. We can specify what happens when we apply a post or get method to a specific URL. We can use parameters to specify our URL. Sinatra permits us to use views, erb files that are placed in the views folder are interpreted when our code is running.

Sequel: We use sequel for ORM, which lets our webapp interact with the database in a more convenient way by mapping objects in memory to our persistent database, and also identifying which queries we need to use. We use models to represent the many types of object types we're storing in our database.

Puma: Puma is a ruby webserver that we need to use in order for Sinatra to work correctly.

Postgres: Postgres is a database management system.

Rails: Rails, or ruby on rails, is a webapp framework that includes many functionalities useful for interacting with our database through our app.

3.3 Communications Interfaces

We are using the HTTP protocol for our communication needs. HTTP is a protocol that defines the way to transfer information (in the form of text usually) across the net, between a client and a server. HTTP sends out a request, which our client makes to the server, and the server replies with a response. The methods we are using from the request side are mainly get (I want the server to send something to me) and post (I want to send information to the server for processing).
From the response side, we mostly return either 201 (created) when we create something, or 500 (internal server error) when we encountered an unexpected error in said creation.

4. System Features

4.1 User stories

Following are some user stories that describe the requirements of our system:

Title: Correct Result

As a user, I want my answers to the test to determine the results that most fit me, so that I can know what my answers say about myself.

Priority: High

Title: Result Display

As a user, I want to see what result fits me most, along with how relevant the other results where, so I can compare and think about which career I like most.

Priority: High

Title: Easy-to-use Interface

As a user, I want the test's graphic interface to be intuitive, so that I can easily answer it.

Priority: Medium

Title: Question management

As an administrator, I want to be able to create new questions for my vocational test, or delete them as needed, so that I can control the contents of my test.

Priority: High

Title: Career management

As an administrator, I want to be able to create or delete a new career to be considered in my vocational test, so that I can decide which careers my test contemplates.

Priority: High

Title: Outcome Managing

As an administrator, I want to be able to create or delete outcomes (the way in which it is decided whether a question's answer influences a certain career) as well as modify their weighting, so that I can provide a more accurate result for my test.

Priority: High

Title: Relevant information

As an administrator, I want to be able to see information about my different questions, careers, and outcomes, so that I can decide if they need to be modified or deleted.

Priority: High

Title: Survey storing

As an administrator, I want to be able to see a history of who took the test and which result they got, so that I may have information that could prove useful for statistics.

Priority: Low

Title: Answer Question

As a user, I want to be able to answer every question of my test with the value of my choosing, so I can get an answer tailored to my preferences.

Priority: High

Title: Authentification System

As an administrator, I want my system to have a reliable authentification system, so that only people of my choosing can modify, create or delete tests, test questions or results.

Priority: High

Title: Efficient Result

As an administrator, I want my tests to be reasonably efficient when processing the answers, so that users don't have to wait long for results.

Priority: Low

Title: Administrator appointment

As a manager, I want to be able to choose who is an administrator, so that I can change my administrator team as needed.

Priority: Low

5. Other Nonfunctional Requirements

5.1 Security Requirements

If time allows it, the product should have an authentication system so that only administrators can modify the test selection. If possible, the manager/s should have higher security, given that their accounts have the most privileges.

5.2 Software Quality Attributes

Our software should be able to efficiently determine how every response to a question will affect the result of the vocational test. We need a way to "weigh" how much every career relates to a certain question. Our database will also have testing capabilities, to see if our models are working as intended.

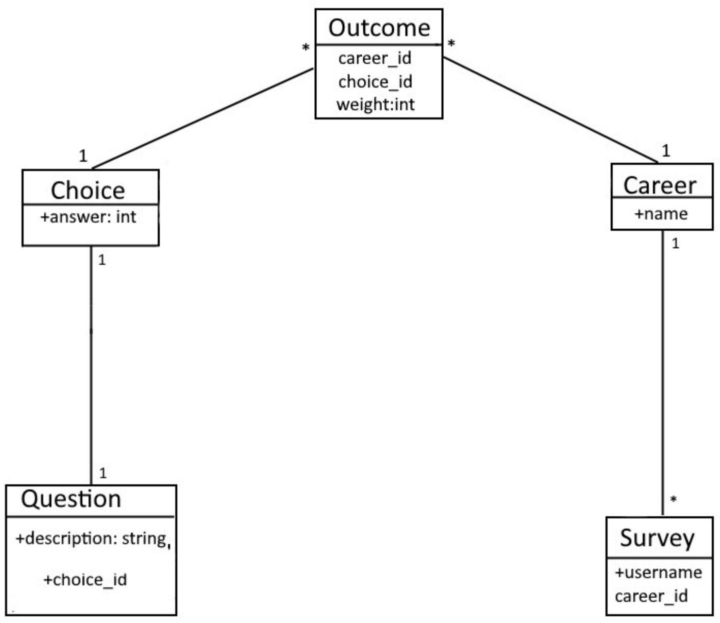
6. Other Requirements

Appendix A: Glossary

-

Appendix B: Analysis Models

Class diagram associated to our system:



Appendix C: To Be Determined List

-