**Implementation Model Document**

**for**

**+Health**

## Version 1.0 Approved

**Prepared by:**

Cindy Torres

Ricardo Luna

Giovanni Guzmán

**11/29/19**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 11/29/2019 | 1.0 | Implementation Model Document 1.0 | Ricardo Luna Cindy Torres Giovanni Guzmán |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. Introduction 4
   1. Purpose 4
   2. Scope 4
   3. Definitions, Acronyms and Abbreviations 4
   4. References 5
   5. Overview 5
2. Overall Description 5

2.1 Assumptions and Dependencies 5

1. Unit Testing 6

3.1 Create an Account 7

3.2 View Patient’s Medical History 10

3.3 Create Patient’s Diagnosis 14

3.4 Create Patient’s Treatment 16

3.5 Upload Medical Studies 18

1. Unit Testing 20
2. Source code and functional prototype 22
3. **Introduction**

An implementation model is a collection of components, and the implementation subsystems that contain them. Components include both deliverable components, such as executables, and components from which the deliverables are produced, such as source code files.

Implementation is the central activity in software development. Requirements and architecture are done before construction effectively. System testing (in the strict sense of independent testing) is done after construction to verify that construction has been done correctly. Construction is at the center of the software development process.

* 1. **Purpose**

The purpose of the Implementation Model is to describe how design model’s elements are implemented in terms of components. It also describes how components are organized according to structuration and modularization approaches available in the environment.

* 1. **Scope**

The purpose of +Health is to ease patients’ medical information management and to create a convenient and easy-to-use application for doctors, trying to have all their patients’ medical information in a handy way. Above all, it is expected for the application to provide a comfortable user experience.

* 1. **Definitions, Acronyms and Abbreviations**
* Patient’s Medical Information: Medical general information of the patient such as height, weight, chronic diseases and clinical information.
* Patient’s Medical History: Data related to a person’s medical history, including symptoms, diagnoses, procedures, and outcomes. It also includes patient histories, lab results, x-rays, and so on.
* Patient’s Code. Unique identifier that the system assigns to a patient.  
  1. **References**

Software Requirements Specification. (2019). 2nd ed. +Health.

Analysis Model Document. (2019). 1st ed. +Health.

Design Model Document. (2019). 1st ed. +Health.

* 1. **Overview**

The remaining sections of this document show you a direct link to the source code, the unit tests, a code check-list and some screenshots of the application.

1. **Overall Description**

What is +Health? +Health is an application that pretends to gather all the patients’ medical information such as chronic diseases, medical analysis, prescriptions, treatments, diagnoses and operations.

+Health purpose is to eliminate the lack of information of a patient on their medical history by having a centralized system that: 1) is able to access medical information at any given moment. 2) give access to a third party when needed. It will give a much wider and concrete insight on a patient (speaking in medical terms).

The patient will be able to access any of their medical diagnoses and treatments, mitigating misunderstandings between the parties involved. This will also improve the verification of a diagnoses and treatments from a third party, since all the information will be presented directly through the system.

Having a centralized system with all the medical history of a patient will keep misinterpretations to a minimum.

**2.1 Assumptions and Dependencies**

This system does not have any kind of dependencies.

1. **Unit Testing**

Unit Testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

* 1. **Create an Account**

Test Scenario ID: 1 Test Priority: High

Test Case Description: Test Case for new\_account\_information () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Enter the information of the new account. | New account information (first name, last name, email, password, birthdate, phone, sex, school, graduation date, speciality and hospital). | It is created a new doctor with all the given attributes. | It is created a new doctor with all the given attributes. | Passed. |

Test Scenario ID: 2 Test Priority: High

Test Case Description: Test Case for validate\_information () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Validate that the information of a new doctor is not null. | New doctor object with all their attributes (first name, last name, email, password, birthdate, phone, sex, school, graduation date, speciality and hospital). | All the data of the doctor is not null. | All the data of the doctor is not null. | Passed. |

Test Scenario ID: 3 Test Priority: High

Test Case Description: Test Case for create\_new\_doctor () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Insert a created doctor into the database. | New doctor object with all its attributes validated (first name, last name, email, password, birthdate, phone, sex, school, graduation date, speciality and hospital). | The doctor is inserted into the database. | The doctor is inserted into the database. | Passed. |

Test Scenario ID: 4 Test Priority: High

Test Case Description: Test Case for upload\_medical\_degree () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Enter the medical certification degree file. | New medical certification degree file. | It is created a new doctor with the medical certification degree. | It is created a new doctor with the medical certification degree. | Passed. |
| 2 | Enter a filename that is not supported. | New medical certification degree with a filename that is not supported. | Filename is not supported. | Filename is not supported. | Passed. |

Test Scenario ID: 5 Test Priority: High

Test Case Description: Test Case for validate\_degree\_file () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Validate that the filename of the new medical certification degree is not null. | Not null medical certification degree filename. | The medical certification degree filename is not null. | The medical certification degree filename is not null. | Passed. |

Test Scenario ID: 6 Test Priority: High

Test Case Description: Test Case for create\_degree () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Update the doctor information adding the new medical certification degree | New medical certification degree object with all its attributes validated. | The doctor information is updated in the database. | The doctor information is updated in the database. | Passed. |

* 1. **View Patient’s Medical History**

Test Scenario ID: 1 Test Priority: High

Test Case Description: Test Case for enter\_patients\_code () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Enter the id of the patient to search | Patient information (patient id) | It is created a new object with all the patient information. | It is created a new object with all the patient information. | Passed. |

Test Scenario ID: 2 Test Priority: High

Test Case Description: Test Case for validate\_code () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Validate that the information of the new patient is not null. | New patient object with its patient id. | All the data of the patient is not null. | All the data of the patient is not null. | Passed. |

Test Scenario ID: 3 Test Priority: High

Test Case Description: Test Case for get\_patient () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Search the correct patient given their patient\_id. | New patient object with its patient id validated. | Select the correct patient and its id. | Select the correct patient and its id. | Passed. |

Test Scenario ID: 4 Test Priority: High

Test Case Description: Test Case for request\_general\_information () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Request the patient information. | Patient confirmation and information (patient\_id). | Update the permissions of the doctor in the database and request the information. | Update the permissions of the doctor in the database and request the information. | Passed. |
| 2 | Request the patient information. | Patient denegation and information (patient\_id) | Doctor permissions update denied. | Doctor permissions update denied. | Passed. |

Test Scenario ID: 5 Test Priority: High

Test Case Description: Test Case for retrieve\_general\_information () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Retrieve general information. | Patient information (patient\_id). | Retrieve the general information of the patient. | Retrieve the general information of the patient. | Passed. |

Test Scenario ID: 6 Test Priority: High

Test Case Description: Test Case for request\_medical\_history () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Request the medical history of the patient. | Patient information (patient\_id). | It is created a new object with the patient information. | It is created a new object with the patient information. | Passed. |

Test Scenario ID: 7 Test Priority: High

Test Case Description: Test Case for retrieve\_medical\_history () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Validate that the patient information of the new object is not null. | Patient information (patient\_id). | All the data of the patient is not null. | All the data of the patient is not null. | Passed. |

Test Scenario ID: 8 Test Priority: High

Test Case Description: Test Case for get\_patient () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Get the patient and all their information. | Patient information (patient\_id). | Get the patient information. | Get the patient information. | Passed. |
| 2 | Get all the treatments of a patient. | Patient information (patient\_id). | Get the patient treatments. | Get the patient treatments. | Passed. |
| 3 | Get all the diagnosis of a patient. | Patient information (patient\_id). | Get the patient diagnosis’. | Get the patient diagnosis’. | Passed. |
| 4 | Get all the medical studies of a patient. | Patient information (patient\_id). | Get the patient medical studies. | Get the patient medical studies. | Passed. |

* 1. **Create Patient’s Diagnosis**

Test Scenario ID: 1 Test Priority: High

Test Case Description: Test Case for add\_diagnosis () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Enter the information of the new diagnosis. | New diagnosis information (date, description, type, patient id and doctor id). | It is created a new diagnosis with all the given attributes. | It is created a new diagnosis with all the given attributes. | Passed. |

Test Scenario ID: 2 Test Priority: High

Test Case Description: Test Case for validate\_diagnosis\_information () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Validate that the information of a new diagnosis is not null. | New diagnosis object with all their attributes (date, description, type, patient id and doctor id). | All the data of the diagnosis is not null. | All the data of the diagnosis is not null. | Passed. |

Test Scenario ID: 3 Test Priority: High

Test Case Description: Test Case for create\_new\_diagnosis () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Insert a created diagnosis into the database. | New diagnosis object with all its attributes validated (date, description, type, patient id and doctor id). | The diagnosis is inserted into the database. | The diagnosis is inserted into the database. | Passed. |

* 1. **Create Patient’s Treatment**

Test Scenario ID: 1 Test Priority: High

Test Case Description: Test Case for add\_prescription () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Enter the information of the new prescription. | New prescription information (description, patient id, doctor id, type, date created, dose, administration and frequency). | It is created a new prescription with all the given attributes. | It is created a new prescription with all the given attributes. | Passed. |

Test Scenario ID: 2 Test Priority: High

Test Case Description: Test Case for validate\_prescription () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Validate that the information of a new diagnosis is not null. | New prescription object with all their attributes (description, patient id, doctor id, type, date created, dose, administration and frequency). | All the data of the prescription is not null. | All the data of the prescription is not null. | Passed. |

Test Scenario ID: 3 Test Priority: High

Test Case Description: Test Case for create\_prescription () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Insert a created diagnosis into the database. | New prescription object with all its attributes validated (description, patient id, doctor id, type, date created, dose, administration and frequency). | The prescription is inserted into the database. | The prescription is inserted into the database. | Passed. |

* 1. **Upload Medical Studies**

Test Scenario ID: 1 Test Priority: High

Test Case Description: Test Case for add\_medical\_study () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Enter the information of the new medical study. | New medical study information (name, patient id, date, type, description). | It is created a new medical study with all the given attributes. | It is created a new medical study with all the given attributes. | Passed. |

Test Scenario ID: 2 Test Priority: High

Test Case Description: Test Case for validate\_medical\_study () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Validate that the information of a new medical study is not null. | New medical study with all their attributes (name, patient id, date, type, description). | All the data of the medical study is not null. | All the data of the medical study is not null. | Passed. |

Test Scenario ID: 3 Test Priority: High

Test Case Description: Test Case for create\_medical\_study () method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Action** | **Inputs** | **Expected Output** | **Actual**  **Output** | **Test Result** |
| 1 | Insert a created medical study into the database. | New medical study object with all its attributes validated (name, patient id, date, type, description). | The medical study is inserted into the database. | The medical study is inserted into the database. | Passed. |

1. **Code Check-list**

Does the code work? Does it perform its intended function, the logic is correct etc.

Is all the code easily understood?

Does it conform to your agreed coding conventions? These will usually cover location of braces, variable and function names, line length, indentations, formatting, and comments.

Is there any redundant or duplicate code?

Is the code as modular as possible?

Is there any commented out code?

Do loops have a set length and correct termination conditions?

Do the names used in the program convey intent?

Are there any obvious optimizations that will improve performance?

Can any of the code be replaced with library or built-in functions?

Can any logging or debugging code be removed?

Are output values checked and encoded?

Are invalid parameter values handled?

Do comments exist and describe the intent of the code?

Is any unusual behavior or edge-case handling described?

Is there any incomplete code?

Is the code testable?

Do tests exist, and are they comprehensive?

Do unit tests actually test that the code is performing the intended functionality?

Could any test code be replaced with the use of an existing API?

1. **Source Code and Functional Prototype**

Here is the direct link to our repository on GitHub:

https://github.com/richardmoonw/PlusHealth.git









