# Software Requirements

# Software Requirements Specification (SRS)

TeaParty – Server Team

Contents

[Software Requirements 1](#_Toc70375998)

[Software Requirements Specification (SRS) 1](#_Toc70375999)

[1.  Introduction 3](#_Toc70376000)

[1.1    Intended Audience and Purpose 3](#_Toc70376001)

[1.2    How to use the document. 3](#_Toc70376002)

[2.  Concept of Operations 4](#_Toc70376003)

[2.1    System Context 4](#_Toc70376004)

[2.2 System capabilities 4](#_Toc70376005)

[3.  Use Cases 5](#_Toc70376006)

[3.1 Client Update Data 5](#_Toc70376007)

[3.2 Client Queries Data 6](#_Toc70376008)

[3.3 Client Manage Clients Information 6](#_Toc70376009)

[3.4 Client Manage Treatment Record 7](#_Toc70376010)

[3.5 Client Create Diagnosis 8](#_Toc70376011)

[3.6 Client Manage Algorithm 9](#_Toc70376012)

[3.7 Client Send Message 10](#_Toc70376013)

[3.8 Client Sign Up 10](#_Toc70376014)

[3.9 Client Log In 11](#_Toc70376015)

[3.10 Client Compares Diagnostics 12](#_Toc70376016)

[3.11 Client Consult Readings/Videos 13](#_Toc70376017)

[3.12 Client View Command List 13](#_Toc70376018)

[3.13 Client Report Problem 14](#_Toc70376019)

[3.15 Database Send Client Personal Information 15](#_Toc70376020)

[3.15 Database Send Client Medical Record 15](#_Toc70376021)

[3.16 Database Send Receive Messages 16](#_Toc70376022)

[3.16 Database Send Chatbot List of Commands 17](#_Toc70376023)

[4.1 System Inputs and Outputs 18](#_Toc70376024)

[4.1.1 Inputs 18](#_Toc70376025)

[4.1.2 Outputs 18](#_Toc70376026)

[5. Quality Requirements 18](#_Toc70376027)

[6. Expected Subsets 18](#_Toc70376028)

[7. Expected Changes 19](#_Toc70376029)

## 1.  Introduction

### 1.1    Intended Audience and Purpose

This document wants provide information and specifications guiding the operation on the development process of Server module, ensuring that all system requirements are met. So, the following entities should have access to this document, the first entities is the System Architect that need this document to know how to design the architecture of server, the following one is developers that need it to develop all use cases mentioned in this report, and the tester will need this document too, to create a comprehensive testing regimen that will guarantee requirements are met.

### 1.2    How to use the document.

1. Introduction

2. Concept Operations – Describe the main goal of the project and explication about the solution

2.1 System Context – System requirements that application will need to run.

2.2 System Capabilities – Descriptions of all capabilities available from the server to the other modules

3. Use Cases – Detailed Specification of all modules use cases.

Table Contents

Specifications

4. Behavioral Requirements - How the server module will interact with other modules.

4.1 Input and output requirements

4.1.1 Input- Describes any restrictions that will be placed on allowed input.

4.1.2 Output - Describes the range of outputs that can be generated.

4.2 Detailed Output Behavior – Output descriptions

5. Quality Requirements - Requirements not pertaining to the function of the application will be listed here

6. Expected Subsets - Expected levels of functionality at checkpoints during development

7. Fundamental Assumptions – Some specifics about input, output, or behavior upon which other requirements are founded will be listed here

8. Expected Changes - Future features and directions the project is expected to take

## 2.  Concept of Operations

The goal of the team is creating and development of server module to the scoliosis self-examination system. Basically, in one system like that, is necessary one model responsible for the communication and data transfer between the other modules for the good functioning of the system. So, our team proposes to develop a server with the use cases present in this document, that will provide communication between client’s modules and server modules like was explain in use cases specifications.

### 2.1    System Context

**System Requirements:**

(What are our system requirements?)

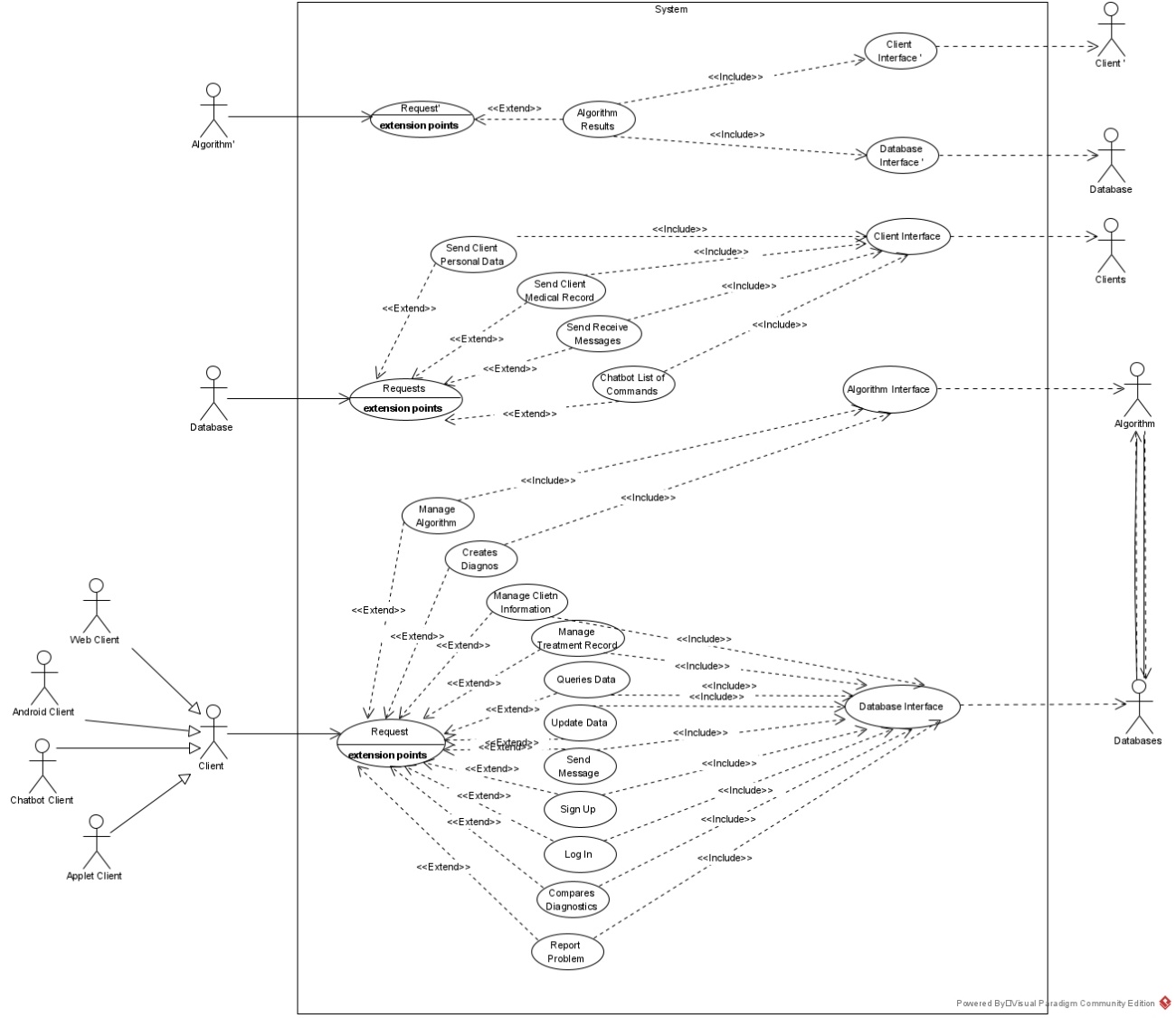
### 2.2 System capabilities

Our server capabilities are simple, every time that he had a request from any module, he will try to deliver the response to the target module and bring back to the origin module deliver the response. In this project we have seven different modules, the Website, Android Application, Chatbot, Server, Algorithm, and Database, the first four modules are Client modules and the last three are Server modules.

For the Clients modules, the server provides them possibility to make many requests to the algorithm or to database, like requests to search clients data’s, in case of the target module is database or to make a new diagnosis in case the target module is the algorithm.

For the Servers Modules, basically, the Server provides to them the possibility to respond to the client’s modules requests and interact between them, because the algorithm needs to send your diagnosis in the database to be saved in the client record and the database need to send the X-Ray images to the algorithm to training and testing the model.

## 3.  Use Cases



### 3.1 Client Update Data

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Client Update Data | | |
| Version | V1 | Created (date): | *19/04/2021* |
| Author | Pedro Nunes | | |
| Source | Client (Web) (Android) (Applet) | | |
| Goals | Update data in database about client (doctors or patients), like patients’ medical record, patient’s personal data, doctor’s personal data and doctor’s schedule. Patient can change yours doctor. | | |
| Summary | How Client has session logged in, he can raise a request to update data. After request, the server contacts database to apply the modifications, after this the modifications are applied. Clients can be doctors, patients, and administrators: | | |
| Actors | Client (Web) (Android) | | |
| Trigger | Client does request to update data | | |
| Precondition | Server is running, client is logged in, and already has data registered in database. | | |
| Frequency | Rarely | | |
| Postconditions | Data Updated | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  | Send a request to server |  |
|  |  | Apply the modifications (Move to alternate flow 1 when this flow failed) |
| **Alternative Flow** | *Actor* | *System* |
|  |  | Tell client that the request is illegal. End. |

### 3.2 Client Queries Data

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Client Queries Data | | |
| Version | V1 | Created (date): | *12/04/2021* |
| Author | Pedro Nunes | | |
| Source | Client (Web) (Android) (Applet) (Chatbot) | | |
| Goals | Client queries data in database, like patient record (diagnosis and personal data), and doctor´s schedule and doctor´s personal data. Clints (patient) (doctors) also can see his messages | | |
| Summary | How Client has session logged in, he can raise a request to query data. After request to query what he wants, the system contacts the database to get the data that Client has requested. If user is a Doctor, this one can queries your own data, like personal information, his schedule, and data from your patients, if the users are a Patient, he can query your own data and some data from your doctor. The admin can query data from everyone. | | |
| Actors | Client (Web) (Android) (Chatbot) | | |
| Trigger | Web Client does request to queries data | | |
| Precondition | Server is running, client is logged in, and have permissions to queries that data. | | |
| Frequency | Frequently | | |
| Postconditions | Client queries are answered. | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  | Raise an App request |  |
|  |  | Answer queries. (Move to alternate flow 2 when failed.) |
| **Alternative Flow** | *Actor* | *System* |
|  |  | Tell client that the request is illegal. End. |

### 3.3 Client Manage Clients Information

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Client Manage User Information | | |
| Version | V1 | Created (date): | *23/04/2021* |
| Author | Pedro Nunes | | |
| Source | Client (Web) (Applet) | | |
| Goals | Manage all user information. Add new accounts in system and delete accounts. | | |
| Summary | How client has session logged in, he can do a request to manage user information. After request, the server contacts database to apply the modifications, after this the modifications are applied. Web clients can be administrators.  He has permissions to add new accounts in system, delete accounts, edit accounts, and search account in database. Only Administrator have permissions to this use case. | | |
| Actors | Client (Web) | | |
| Trigger | Client does request to manage client data | | |
| Precondition | Server is running, Web Client is logged in. | | |
| Frequency | Frequently | | |
| Postconditions | Modifications are applied | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  | Send a request to server |  |
|  |  | Apply the modifications (Move to alternate flow 1 when this flow failed) |
| **Alternative Flow** | *Actor* | *System* |
|  |  | Tell client that the request is illegal. End. |

### 3.4 Client Manage Treatment Record

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Client Manage Treatment Record | | |
| Version | V1 | Created (date): | *23/04/2021* |
| Author | Pedro Nunes | | |
| Source | Client (Web) (Applet) | | |
| Goals | Manage all treatment records. Add new records in system, delete records and search records | | |
| Summary | How client has session logged in, he can do a request to manage treatment record. After request, the server contacts database to apply the modifications, after this the modifications are applied. Client can be administrator.  He has permissions to add new records in system, delete accounts, and search records in database. | | |
| Actors | Client (Web) | | |
| Trigger | Client does request to manage treatment record | | |
| Precondition | Server is running, Web Client is logged in. | | |
| Frequency | Frequently | | |
| Postconditions | Data Updated | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  | Send a request to server |  |
|  |  | Search for corresponding record in system (Move to alternate flow 1 when this flow failed) |
|  |  | Apply the modifications (Move to alternate flow 2 when this flow failed) |
| **Alternative Flow** | *Actor* | *System* |
|  |  | Tell client that the record was not found. End. |
|  |  | Tell client that the request is illegal. End. |

### 3.5 Client Create Diagnosis

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Web Client Create Diagnosis | | |
| Version | V1 | Created (date): | *12/04/2021* |
| Author | Pedro Nunes | | |
| Source | Client (Web) (Android) (Chatbot) | | |
| Goals | Client creates a patient diagnosis. | | |
| Summary | How client has session logged in, he can do request to create a diagnosis. After request, the server contacts algorithm to create a new diagnosis, then algorithm send to database the new diagnosis to be saved in patient record. The diagnosis can be made by patients, the doctor can make the diagnosis by adding the patient's x-ray to the algorithm. | | |
| Actors | Web Client (Web) (Android) (Chatbot) | | |
| Trigger | Client does a request to create diagnosis. | | |
| Precondition | Server is running and web client is logged in. | | |
| Frequency | Frequently | | |
| Postconditions | Diagnosis of one client have been made | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  | Raise an API request |  |
|  |  | Checks if the client's x-ray is valid. (Move to alternative flow 1 when this flow failed) |
|  |  | Algorithm send the new diagnosis to database. (Move to alternative flow 2 when this flow failed) |
| **Alternative Flow** | *Actor* | *System* |
|  |  | Tell client that his X-Ray is invalid. End. |
|  |  | Tell client that he can´t save the diagnosis in patient medical record. End. |

### 3.6 Client Manage Algorithm

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Manage Algorithm | | |
| Version | V1 | Created (date): | *12/04/2021* |
| Author | Pedro Nunes | | |
| Source | Client (Web) | | |
| Goals | Clients can manage, make changes in the algorithm, or even change algorithm for another algorithm. | | |
| Summary | Client (Administrator) do request to manage the algorithm. After request, the server contacts algorithm interface to apply changes in algorithm, or even change the algorithm for another. | | |
| Actors | Client (Web) | | |
| Trigger | Client does an API request. | | |
| Precondition | Server is running and client is logged in. | | |
| Frequency | Rarely | | |
| Postconditions | Changes were made in the algorithm | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  | Raise an API request |  |
|  |  | Check request is valid or not. (Move to alternate flow 1 when failed.) |
| **Alternative Flow** | *Actor* | *System* |
|  |  | Tell client that the request is illegal. End. |

### 3.7 Client Send Message

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Send Message | | |
| Version | V1 | Created (date): | *12/04/2021* |
| Author | Pedro Nunes | | |
| Source | Client (Web) (Android) (Chatbot) | | |
| Goals | Clients send a message to another client. The client can be a patients, doctors, and administrators, they can send messages to each other. | | |
| Summary | How client has session logged in, he can do a request to send a message to another user, after this the server get in contact with database to get the other user´s contact, then after receiving the contact, message is sent to other client. | | |
| Actors | Client (Web) (Android) (Chatbot) | | |
| Trigger | Client does a request to send a message. | | |
| Precondition | Server is running and client is logged in. | | |
| Frequency | Frequently | | |
| Postconditions | Message is sent to destiny | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  | Raise an API request |  |
|  |  | Check request is valid or not. (Move to alternate flow 1 when failed.) |
|  |  | Check if found client contact in database. (Move to alternate flow 2 when failed) |
| **Alternative Flow** | *Actor* | *System* |
|  |  | Tell client that the request is illegal. End. |
|  |  | Tell Web Client than can´t find the client contact. End |

### 3.8 Client Sign Up

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Client Sign Up | | |
| Version | V1 | Created (date): | *22/04/2021* |
| Author | Pedro Nunes | | |
| Source | Client (Web) (Android) (Applet) | | |
| Goals | Client create new account in system, he can be a patient, doctor, or administrator. | | |
| Summary | Client does a request to sign up, after this the server get in contact with database to create new account in system. | | |
| Actors | Client (Web) (Android) (Applet) | | |
| Trigger | Client does a request. | | |
| Precondition | Server is running and web client isn´t logged in. | | |
| Frequency | Frequently | | |
| Postconditions | Web client had access to his new account | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  | Raise an API request |  |
|  |  | Check request is valid or not. (Move to alternate flow 1 when failed.) |
|  |  | Check data of new account. (Move to alternate flow 2 when failed) |
| **Alternative Flow** | *Actor* | *System* |
|  |  | Tell client that the request is illegal. End. |
|  |  | Tell Web Client than can´t create new account. End |

### 3.9 Client Log In

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Client Log In | | |
| Version | V1 | Created (date): | *22/04/2021* |
| Author | Pedro Nunes | | |
| Source | Client (Web) (Android) (Applet) | | |
| Goals | Login of the client in system, to have access to the system, its functionalities, and data. | | |
| Summary | Client does a request to Log In, after this the server get in contact with database to check user credentials to log in your account. | | |
| Actors | Client (Web) (Android) (Applet) | | |
| Trigger | Client does a request. | | |
| Precondition | Server is running and web client isn´t logged in. | | |
| Frequency | Frequently | | |
| Postconditions | Clients enter in your system account | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  | Raise an API request |  |
|  |  | Check request is valid or not. (Move to alternate flow 1 when failed.) |
|  |  | Check Web user credentials. (Move to alternate flow 2 when failed) |
| **Alternative Flow** | *Actor* | *System* |
|  |  | Tell client that the request is illegal. End. |
|  |  | Tell Web Client than his credentials wasn´t found in system. End |

### 3.10 Client Compares Diagnostics

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Compares Diagnostics | | |
| Version | V1 | Created (date): | *19/04/2021* |
| Author | Pedro Nunes | | |
| Source | Client (Android) | | |
| Goals | Clients receive the comparison between two different diagnosis in a graph | | |
| Summary | How client has session logged in, he can do a request to get a comparison between two diagnosis. Then the server contacts the databases to get the diagnosis, to do the comparison. | | |
| Actors | Client (Android) | | |
| Trigger | Client does a request to compare diagnostics. | | |
| Precondition | Server is running and client is logged in. | | |
| Frequency | Rarely | | |
| Postconditions | App Client get comparison between two diagnosis | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  | Raise an API request |  |
|  |  | Check request is valid or not. (Move to alternate flow 1 when failed.) |
|  |  | Check if found the diagnosis in database. (Move to alternate flow 2 when failed) |
| **Alternative Flow** | *Actor* | *System* |
|  |  | Tell client that the request is illegal. End. |
|  |  | Tell Client than can´t find the diagnosis in database. End |

### 3.11 Client Consult Readings/Videos

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Client Consult Readings/Videos | | |
| Version | V1 | Created (date): | *19/04/2021* |
| Author | Pedro Nunes | | |
| Source | Client (Android) | | |
| Goals | Clients consult corresponding readings and videos about scoliosis. | | |
| Summary | How client has session logged in, he can does a request to get access to readings and videos about scoliosis, server contact the databases to get corresponding information. Readings and videos, can be accessed by patients and doctors. | | |
| Actors | Client (Android) | | |
| Trigger | Client does a request. | | |
| Precondition | Server is running and web client is logged in. | | |
| Frequency | Rarely | | |
| Postconditions | The corresponding videos and readings are presented to client. | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  | Raise an API request | Check request is valid or not. (Move to alternate flow 1 when failed.) |
|  |  | Check if Client had medical record, to correspond information to client. (Move to alternate flow 2 when failed) |
| **Alternative Flow** | *Actor* | *System* |
|  |  | Tell client that the request is illegal. End. |
|  |  | Tell client that he doesn´t have medical record, so he doesn´t have access to information about scoliosis. End |

### 3.12 Client View Command List

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Client View the Command List | | |
| Version | V1 | Created (date): | *24/04/2021* |
| Author | Pedro Nunes | | |
| Source | Client (Chatbot) | | |
| Goals | Clients see list of commands on chatbot. | | |
| Summary | Chatbot Client raise request to get command list of chatbot. After request, the server contacts database to get command list. | | |
| Actors | Chatbot Client | | |
| Trigger | Chatbot Client raises a request to get command list of chatbot. | | |
| Precondition | Server is running and chatbot client send a message for chatbot. | | |
| Frequency | Frequently | | |
| Postconditions | Diagnosis of one client have been made | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### 3.13 Client Report Problem

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Chatbot client Report Problem | | |
| Version | V1 | Created (date): | *24/04/2021* |
| Author | Pedro Nunes | | |
| Source | Client (Chatbot) | | |
| Goals | Chatbot Client report something wrong with chatbot, or with other chatbot users. | | |
| Summary | Client does a request to report something to an administrator. After request server sent report record to an administrator.  If chatbot client was a patient or a doctor, he can report something about the system or about other chatbot user. | | |
| Actors | Chatbot Client (Chatbot) | | |
| Trigger | Client does a request to make a report. | | |
| Precondition | Server is running and client send a message for chatbot. | | |
| Frequency | Frequently | | |
| Postconditions | Report is created. | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
| 1. | Raise a report request to server. |  |
| 2. |  | Send the report to database, where his save in user record. |
| **Alternative Flow** | *Actor* | *System* |
| 1. |  | Tell Web Client that his report is invalid. End |

### 3.15 Database Send Client Personal Information

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Database Send Client Personal Data | | |
| Version | V1 | Created (date): | *26/04/2021* |
| Author | Pedro Nunes | | |
| Source | Database | | |
| Goals | Database send to clients the corresponding personal data that required in his requests. | | |
| Summary | Database does a request to send corresponding personal data to clients. After request to send data be confirmed, the server contacts the database to sending the required data. The required data in this case is client personal data, in case the client be a patient, can be a data about his medical record, next consultation and his doctor information’s. In case that client be a doctor, this one can get your own data and from his patients, your schedule and his contacts. | | |
| Actors | Database | | |
| Trigger | Database raises a Request to send clients data | | |
| Precondition | Server and database are running | | |
| Frequency | Frequently | | |
| Postconditions | Clients receive the requested data. | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  | Raise a request to send client’s data |  |
|  |  | Check request is valid or not. (Move to alternate flow 1 when failed.) |
|  |  | Send data to client. (Move to alternate flow 2 when failed.) |
|  |  |  |
| **Alternative Flow** | *Actor* | *System* |
|  |  | Tell client that the request is illegal. End. |
|  |  | Tell client that the request data are unavailable. End |

### 3.15 Database Send Client Medical Record

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Database Send Client Medical Record | | |
| Version | V1 | Created (date): | *26/04/2021* |
| Author | Pedro Nunes | | |
| Source | Database | | |
| Goals | Database send to clients the corresponding medical record data that required in his requests. | | |
| Summary | Database does a request to send corresponding medical record to clients. After request to send data be confirmed, the server contacts the database to sending the required data. The required data in this case is medical record from patients. Medical records can be required by patient of the diagnosis, doctor of patient and by administrator. | | |
| Actors | Database | | |
| Trigger | Database raises a Request to send clients data | | |
| Precondition | Server and database are running | | |
| Frequency | Frequently | | |
| Postconditions | Clients receive the requested data. | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  | Raise a request to get medical record |  |
|  |  | Check request is valid or not. (Move to alternate flow 1 when failed.) |
|  |  | Send data to client. (Move to alternate flow 2 when failed.) |
| **Alternative Flow** | *Actor* | *System* |
|  |  | Tell client that the request is illegal. End. |
|  |  | Tell client that the request data are unavailable. End |

### 3.16 Database Send Receive Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Database Send the Receive Messages | | |
| Version | V1 | Created (date): | *26/04/2021* |
| Author | Pedro Nunes | | |
| Source | Database | | |
| Goals | Database send to clients the corresponding list of received messages of client. | | |
| Summary | Database does a request to send corresponding list of received messages of client. After request to send data be confirmed, the server contacts the database to sending the required list of messages. | | |
| Actors | Database | | |
| Trigger | Database raises a Request to send clients data | | |
| Precondition | Server and database are running | | |
| Frequency | Frequently | | |
| Postconditions | Clients receive the requested data. | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
|  | Raise a request to receive list of his messages |  |
|  |  | Check request is valid or not. (Move to alternate flow 1 when failed.) |
|  |  | Send data to client. (Move to alternate flow 2 when failed.) |
|  |  |  |
| **Alternative Flow** | *Actor* | *System* |
| 1. |  | Tell client that the request is illegal. End. |
| 2. |  | Tell client that the request data are unavailable. End |

### 3.16 Database Send Chatbot List of Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Database Send List of Commands | | |
| Version | V1 | Created (date): | *26/04/2021* |
| Author | Pedro Nunes | | |
| Source | Database | | |
| Goals | Database send to clients the chatbot list of commands. | | |
| Summary | Database does a request to send the chatbot list of commands. After request to send data be confirmed, the server contacts the database to sending the required list of commands. | | |
| Actors | Database | | |
| Trigger | Database raises a Request to send clients data | | |
| Precondition | Server and database are running | | |
| Frequency | Frequently | | |
| Postconditions | Clients receive the requested data. | | |
| Diagram |  | | |

|  |  |  |
| --- | --- | --- |
| **Basic Flow** | *Actor* | *System* |
| 1. | Raise a request to receive list of his messages |  |
| 2. |  | Check request is valid or not. (Move to alternate flow 1 when failed.) |
| 3. |  | Send data to client. (Move to alternate flow 2 when failed.) |
|  |  |  |
| **Alternative Flow** | *Actor* | *System* |
| 1. |  | Tell client that the request is illegal. End. |
| 2. |  | Tell client that the request data are unavailable. End |

### 3.16 Algorithm send his Results

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Diagnosis Results | | |
| Version | V1 | Created (date): | *26/04/2021* |
| Author | Pedro Nunes | | |
| Source | Algorithm | | |
| Goals | Algorithm send to clients and server the results of diagnosis. | | |
| Summary | Algorithm does a request to send the results of diagnosis. After request to send data be confirmed, the server contacts the algorithm to sending the results to clients and database.  Algorithm results are the key points of the bone, or the points of the two bones form the cobb, and the cobb value. | | |
| Actors | Algorithm | | |
| Trigger | Algorithm raises a request to send diagnosis results | | |
| Precondition | Server and algorithm are running | | |
| Frequency | Frequently | | |
| Postconditions | Clients and database receive the results. | | |
| Diagram |  | | |

4. Behavioral Requeriments

### 4.1 System Inputs and Outputs

### Inputs

Inputs to the server come from the user’s and server’s modules and all inputs vary conform who makes the request. So, the inputs in server are data that’s in request form other modules. The server is thinking to be able to respond to all clients and server modules requests.

### Outputs

The outputs are sent to the server by certain modules, and from the server they are sent to users or to modules such as an algorithm or database. Some outputs of this system are patient´s and doctor personal data, patient’s medical record and other data relation with de clients and with the system.

### Quality Requirements

The application must be competitive with similar applications in regard to performance, reliability, consistency, and scalability, and for that the system needs a server capable to manage all requests from all modules without losing data.

Performance: Responsiveness to user input

\* Standard actions that manipulate address books or their contents should not exceed 500ms execution time.

\* Searching on address books is a possible exception to the above standard, as search performance will depend on the size of the address book.

\* Application should be modularized such that adding/extending features and functions only require changes to a single component and the interface with that component, if applicable.

### Expected Subsets

Our server expect that the system have more six other modules, and all together work so that there is no loss of information during requests and responses that exist in the program.

### Expected Changes

Features to Add:

Any type of new function on the part of customers, algorithm, or database like.

* Function to search a contact writing his name;
* Additional data for customers;
* Needing possibility of new tables in databases for storing data;
* New use cases needed to make the connections between the other modules and the server