Software Requirements Specification

for

Clinic Reception SRS

Prepared by:

Lana Sleman

Carolin Hreqes

Raneem Drdr

Nagham Al Hasan

18 – Jun – 2020

Table of Contents

To do

# Introduction

## Purpose

The purpose of this document is to present a detailed description of the Clinic Reception System. It will explain the purpose and features of the system, the User interfaces of the system, what the system will do, explaining this through the diagrams. This document is intended for both the stakeholders and the developers of the system.

## Project Scope

This System will be a Reception System in a Clinic consisting of several different departments and several doctors in the department. This system will facilitate patient reservations, as well as facilitate the work of doctors in departments.

As the system will provide a simple interface for the doctor to view and add to the patient's medical file. Also, through this system, the patient can book an appointment remotely.

## Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Patient File |  |
| NID |  |
| Database |  |
| Precedent illness |  |
| Medication History |  |
| Social History |  |
| Family History |  |
| Genetic Diseases |  |
| Appointment |  |
| Complaint Onset |  |
| Complaint Last Occur |  |
| Reduce Factors |  |
| Complaint Frequency |  |
| Trigger Factors |  |
| Software Requirements Specification | A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document. |
| Stakeholder | Any person with an interest in the project who is not a developer. |
| User | Patient or Doctor. |

## Overview of Document

To do

# Overall Description

## System Environment

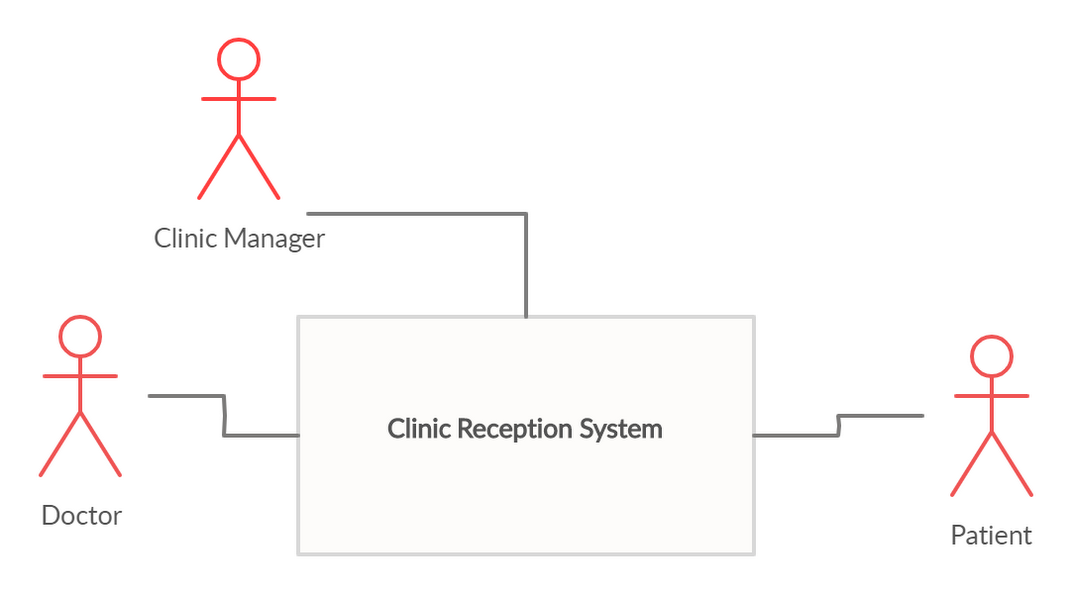


Figure ‎2‑1 System Environment

The Clinic Reception has three active actors. The Clinic Manager and Doctors, who can access the system in the clinic, and Patient, who can access the system through the internet.

## Functional Requirements Specification

This Section outlines the use cases for each active factor. The Diagram below shows the Use Cases Diagram of the System. We will go in details next.

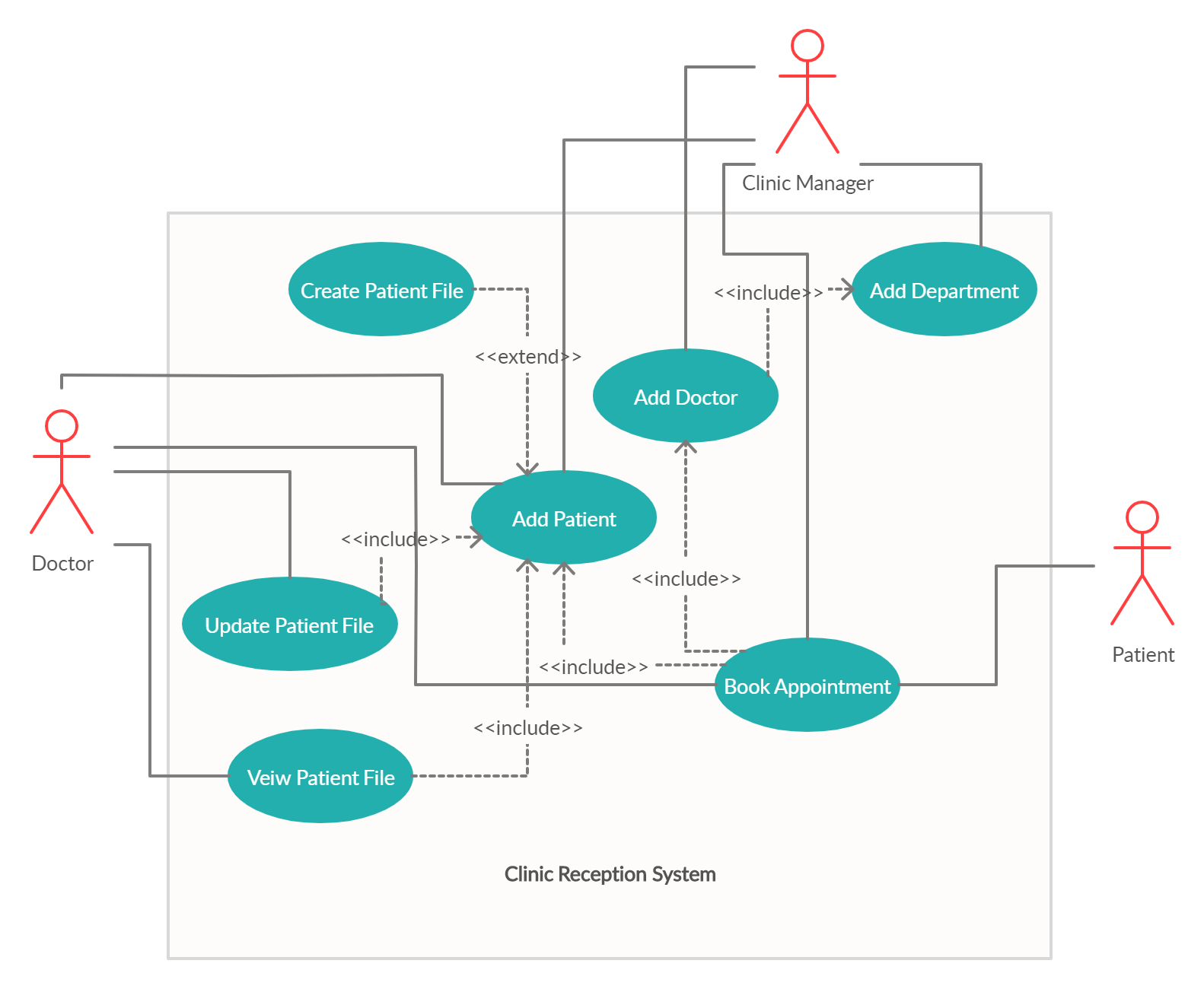


Figure ‎2‑2 Use Case Diagram

### Clinic Manager Use Cases

#### Use case: Add Department

**Diagram:**



**Brief Description**

The Clinic Manager Access the system and add department to the clinic.

**Initial Step-By-Step Description**

1. The Manager access the system.
2. The Manager choose “Add Department”.
3. The System returns a Success Response.

#### Use case: Add Doctor

**Diagram:**



**Brief Description**

The Clinic Manager Access the system and add a doctor to a specific apartment.

This Use Case Require *Add Department* Use Case.

**Initial Step-By-Step Description**

1. The Manager access the system.
2. The Manager choose “Add Doctor”.
3. The Manager Select the Department That New Doctor Belongs to.
4. The System returns a Success Response.

#### Use case: Add Patient

**Diagram:**



**Brief Description**

The Clinic Manager Access the system and add Patient.

This Use Case may extent *Create Patient File* Use Case.

**Initial Step-By-Step Description**

1. The Manager access the system.
2. The Manager choose “Add Patient”.
3. The System returns a Success Response.

#### Use case: Book Appointment

**Diagram:**



**Brief Description**

The Clinic Manager Access the system, Book Appointment for a specific patient at a specific doctor.

This Use Case Require *Add Doctor* and *Add Patient* Use Cases.

**Initial Step-By-Step Description**

1. The Manager access the system.
2. The Manager choose “Book Appointment”.
3. The Manager Search for the Patient.
4. The Manager Search for the Doctor.
5. The Manager Select the Time.
6. The System returns a Success Response.

### Doctor Use Cases

#### Use case: Add Patient

**Diagram:**



**Brief Description**

The Doctor Access the system and add Patient. But Doctor has permissions to add more details about Patient and his illness.

This Use Case may extent *Create Patient File* Use Case.

**Initial Step-By-Step Description**

1. The Doctor access the system.
2. The Doctor choose “Add Patient”.
3. The System returns a Success Response.

#### Use case: View Patient File

**Diagram:**



**Brief Description**

The Doctor Access the system and view Patient File.

This Use Case Require *Add Patient* Use Case.

**Initial Step-By-Step Description**

1. The Doctor access the system.
2. The Doctor Search for the Patient.
3. The Doctor Select “View Patient File”.
4. The System Shows the Patient File.

#### Use case: Update Patient File

**Diagram:**



**Brief Description**

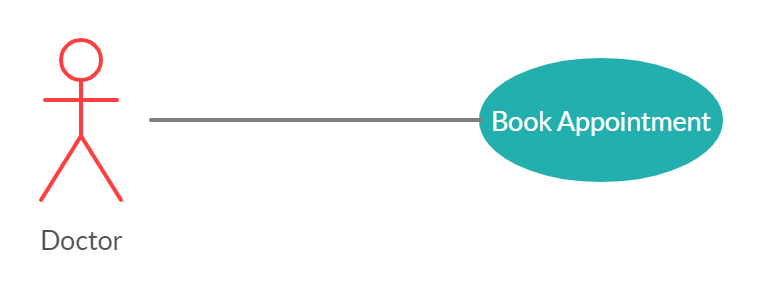
The Doctor Access the system and update patient file. Like adding more illness, the Patient has.

**Initial Step-By-Step Description**

1. The Doctor access the system.
2. The Doctor Search for the Patient.
3. The Doctor Select “Update Patient File”.
4. The Doctor Modifies Patient’s File.
5. The System returns Update Success.

#### Use case: Book Appointment

**Diagram:**



**Brief Description**

The Doctor Access the system and add book an appointment for a patient.

This Use Case Require *Add Patient* Use Case.

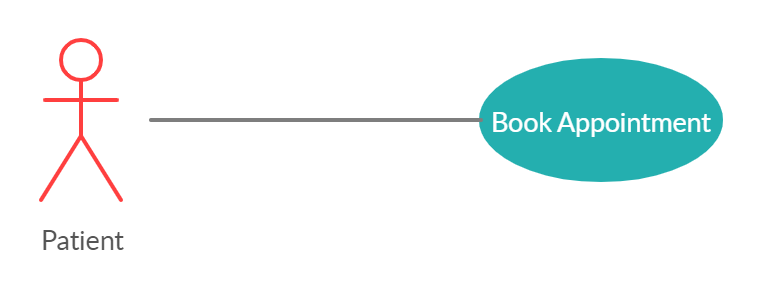
**Initial Step-By-Step Description**

1. The Doctor access the system.
2. The Doctor Search for the Patient.
3. The Doctor Select “Book Appointment”.
4. The Doctor choose Time.
5. The System returns Success Status.

### Patient Use Cases

#### Use case: Book Appointment

**Diagram:**



**Brief Description**

The Patient Access the system and add book an appointment at a specific doctor.

**Initial Step-By-Step Description**

1. The Patient access the system.
2. The Patient Search for the Doctor.
3. The Patient Select “Book Appointment”.
4. The Patient choose Time.
5. The System returns Success Status.

## User Characteristics

The clinic manager is expected to be familiar with the medical departments in general, able to classify doctors within their departments, and be able to use the computer efficiently.

The doctor expects him to be able to deal with patients 'data, amend it, and convert patients' general answers into accurate medical terms and is able to use the computer on average.

The patient is expected to be able to enter the Internet and deal with websites in general.

## Non-Functional Requirements

# Requirements Specification

## Functional Requirements

### Clinic Manager Use Cases

#### Use case: Add Department

|  |  |
| --- | --- |
| **Use Case Name** | Add Department |
| **Trigger** | Manager Click “Add Department” button |
| **Precondition** | The System Show the Fields Required For adding Department |
| **Basic Path** | 1. Manager adds department name 2. Manager adds department location 3. System send the data to database to be added 4. Database send back response (done – failed) 5. If response was (Done) System list the Departments 6. If response was (Failed) System Pops up “wrong message” and back to Precondition Status |
| **Alternative Paths** | Can Access to the Basic Path when adding doctor’s department if the Manager entered Non-existed department |
| **Postcondition** | System lists Departments available in the Clinic |
| **Exception Paths** | System Pops up “wrong message” and back to Precondition Status |
| **Other** |  |

#### Use case: Add Doctor

|  |  |
| --- | --- |
| **Use Case Name** | Add Doctor |
| **Trigger** | Manager Click “Add Doctor” button |
| **Precondition** | The System Show the Fields Required For adding Doctor |
| **Basic Path** | 1. Manager adds Doctor’s Name 2. Manager adds Doctor’s Specification (Department) 3. Manager adds Doctor’s Certification Source 4. Manager adds Doctor’s Start Work Time 5. Manager adds Doctor’s End Work Time 6. System send the data to database to be added 7. Database send back response (done – failed) 8. If response was (Done) System shows Doctor’s Info. 9. If response was (Failed) System Pops up “wrong message” and back to Precondition Status |
| **Alternative Paths** | No Alternative Paths |
| **Postcondition** | The System Shows Doctor’s Info |
| **Exception Paths** | Return to Precondition Status With A Popup Message |
| **Other** |  |

#### Use case: Add Patient

|  |  |
| --- | --- |
| **Use Case Name** | Add Patient |
| **Trigger** | Manager Click “Add Patient” button |
| **Precondition** | The System Show the Fields Required For adding Patient |
| **Basic Path** | 1. Manager Insert Patient’s ID Fields 2. System send the data to database to be added 3. Database send back response (done – failed) 4. If response was (Done) System Shows Patient Id Fields 5. If response was (Failed) System Pops up “wrong message” and back to Precondition Status |
| **Alternative Paths** | If the manager search for a patient how not exist System will show a button “Add new Patient” Which will return back to basic path. |
| **Postcondition** | System Shows Patient Id Fields |
| **Exception Paths** | System Pops up “wrong message” and back to Precondition Status |
| **Other** |  |

#### Use case: Book Appointment

|  |  |
| --- | --- |
| **Use Case Name** | Book Appointment |
| **Trigger** | Manager Click “Book Appointment” button |
| **Precondition** | The System Show the Fields Required For Booking Appointment |
| **Basic Path** | 1. Manager Search for Patient by his name in the Database 2. System return Search results from database and show it in table 3. Manager select Patient if appears in search results 4. System will show a button “Add new Patient” if Patient not exist in results. 5. Manager Select Department and a Doctor from the department 6. Manager Select Time 7. System will send time to database to check if it is has been reserved at the selected doctor. 8. Database return a response (done, failed) 9. If “Done” a new appointment added to database 10. If “failed” system ask manager to select different time. |
| **Alternative Paths** | No Alternative Paths |
| **Postcondition** | System shows appointment details |
| **Exception Paths** | System Pops up “wrong message” and back to Precondition Status |
| **Other** |  |

### Doctor Use Cases

#### Use case: Add Patient

|  |  |
| --- | --- |
| **Use Case Name** | Add Patient |
| **Trigger** | Doctor Click “Add Patient” button |
| **Precondition** | The System Show the Fields Required For adding Patient |
| **Basic Path** | 1. Doctor Insert Patient’s ID Fields 2. System send the data to database to be added 3. Database send back response (done – failed) 4. If response was (Done) System Shows Patient Id Fields 5. If response was (Failed) System Pops up “wrong message” and back to Precondition Status |
| **Alternative Paths** | If the Doctor search for a patient how not exist System will show a button “Add new Patient” Which will return back to basic path. |
| **Postcondition** | System Shows Patient Id Fields with all Patient File Info. |
| **Exception Paths** | System Pops up “wrong message” and back to Precondition Status. |
| **Other** | Note: Doctor Can add more information than Manager, He can add all patient file information. |

#### Use case: View Patient File

|  |  |
| --- | --- |
| **Use Case Name** | View Patient File |
| **Trigger** | Doctor Click “View Patient File” button |
| **Precondition** | System show the Patient Search with Filters |
| **Basic Path** | 1. Doctor Search for Patient by his name or any filter in the Database 2. System return Search results from database and show it in table 3. Doctor select Patient if appears in search results 4. System Shows the Selected Patient File data. 5. System will show a button “Add new Patient” if Patient not exist in results. |
| **Alternative Paths** | No Alternative Paths |
| **Postcondition** | Nothing |
| **Exception Paths** | System Pops up “wrong message” and back to Precondition Status. |
| **Other** |  |

#### Use case: Update Patient File

|  |  |
| --- | --- |
| **Use Case Name** | Update Patient File |
| **Trigger** | Any Changes Made By Doctor on Patient’s File. |
| **Precondition** | System Shows Patient’s File Details. |
| **Basic Path** | 1. Doctor clicks on details he wants to update. 2. After all changes Doctor Click Save Button 3. System send the updated information to Patient’s XML File. 4. If any problem appears with file, system will return problem details. |
| **Alternative Paths** | No Alternative Paths |
| **Postcondition** | Systems shows patient’s file with updated file |
| **Exception Paths** | System Pops up “wrong message” and back to Precondition Status. |
| **Other** |  |

#### Use case: Book Appointment

|  |  |
| --- | --- |
| **Use Case Name** | Book Appointment |
| **Trigger** | Doctor Click “Book Appointment” button |
| **Precondition** | The System Show the Fields Required For Booking Appointment |
| **Basic Path** | 1. Doctor Search for Patient by his name in the Database 2. System return Search results from database and show it in table 3. Doctor select Patient if appears in search results 4. System will show a button “Add new Patient” if Patient not exist in results. 5. Doctor Select Time 6. System will send time to database to check if it has been reserved at the selected doctor. 7. Database return a response (done, failed) 8. If “Done” a new appointment added to database 9. If “failed” system ask Doctor to select different time. |
| **Alternative Paths** | No Alternative Paths |
| **Postcondition** | System shows appointment details |
| **Exception Paths** | System Pops up “wrong message” and back to Precondition Status |
| **Other** |  |

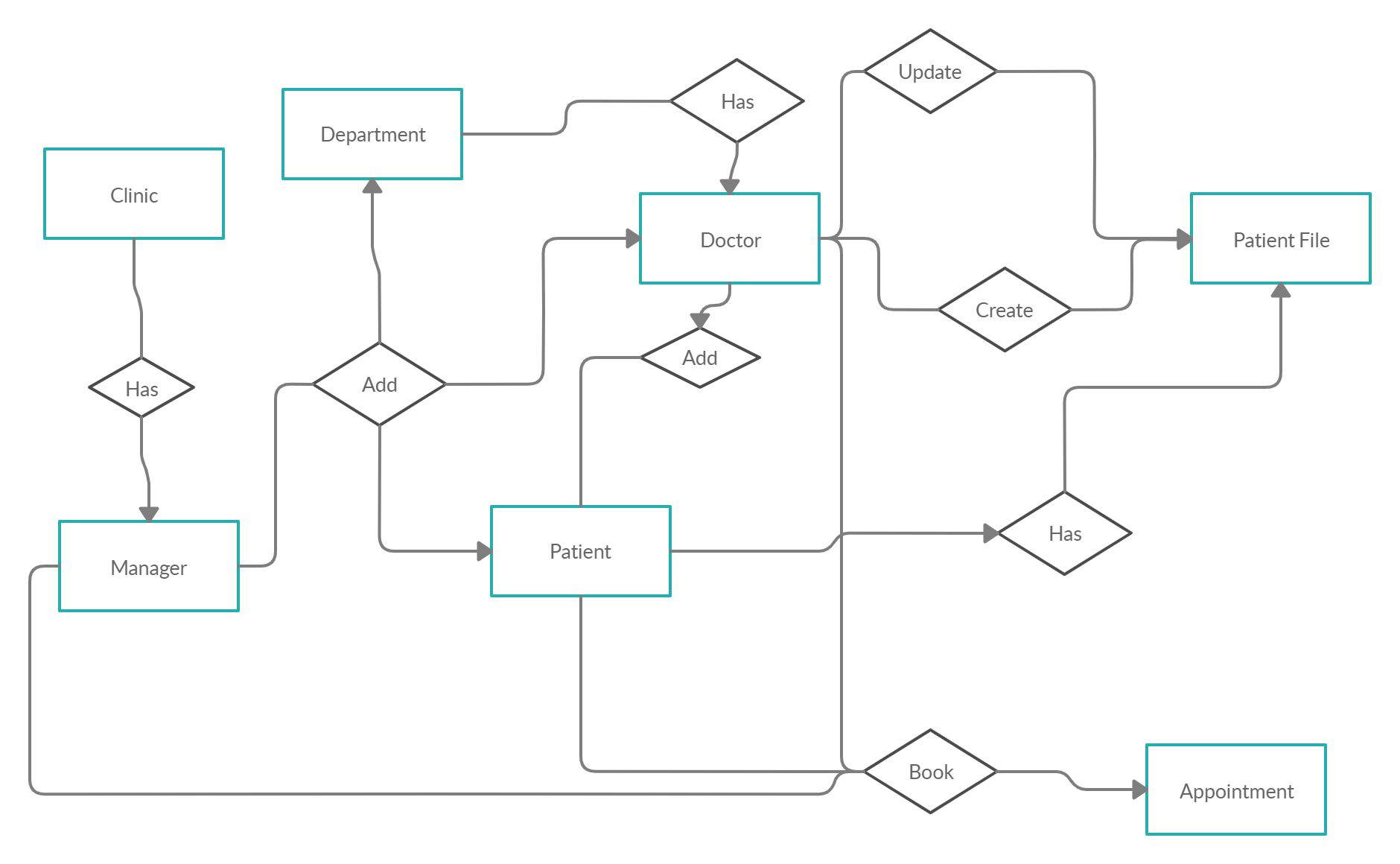
### Patient Use Cases

#### Use case: Book Appointment

|  |  |
| --- | --- |
| **Use Case Name** | Book Appointment |
| **Trigger** | Patient Click “Book Appointment” button |
| **Precondition** | The System Show the Fields Required For Booking Appointment |
| **Basic Path** | 1. Patient Search for Doctor by his name or other filters, in the Database 2. System return Search results from database and show it in table 3. Patient selects Doctor if appears in search results 4. Patient Selects Time 5. System will send time to database to check if it has been reserved at the selected doctor. 6. Database return a response (done, failed) 7. If “Done” a new appointment added to database 8. If “failed” system ask Patient to select different time. |
| **Alternative Paths** | No Alternative Paths |
| **Postcondition** | System shows appointment details |
| **Exception Paths** | System Pops up “wrong message” and back to Precondition Status |
| **Other** |  |

## Detailed Non-Functional Requirement

### Logical Structure of the Data



The data descriptions of each of these data entities is as follows:

**Clinic Data Entity:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Manager | Manager | Manager of the Clinic |  |
| Location | Text | Clinic’s Location |  |

**Department Data Entity:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Department’s Name |  |
| Number of Doctors | Int | Number of Doctors in the Department |  |

**Manager Data Entity:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Manager’s Name |  |

**Doctor Data Entity:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| ID | Manager | Doctor’s ID |  |
| Name | Text | Doctor’s Name |  |
| Specification | Text | Doctor’s Specification |  |
| Certification | Text | Doctor’s Certification Source |  |
| Start | Time | Start Work Time |  |
| End | Time | End Work Time |  |

**Patient Data Entity:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Manager | Manager | Manager of the Clinic |  |
| Location | Text | Clinic’s Location |  |

**Patient File Data Entity:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| ID | Patient ID | Patient’s Id Data |  |
| File Path | Text | Patient’s File Path |  |

**Appointment Data Entity:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| ID | Int | Appointment’s Id |  |
| P\_id | Int | Patient’s ID |  |
| D\_id | Int | Doctor’s ID |  |
| When | Date | Appointment’s Date |  |
| At | Time | Appointment’s Time |  |

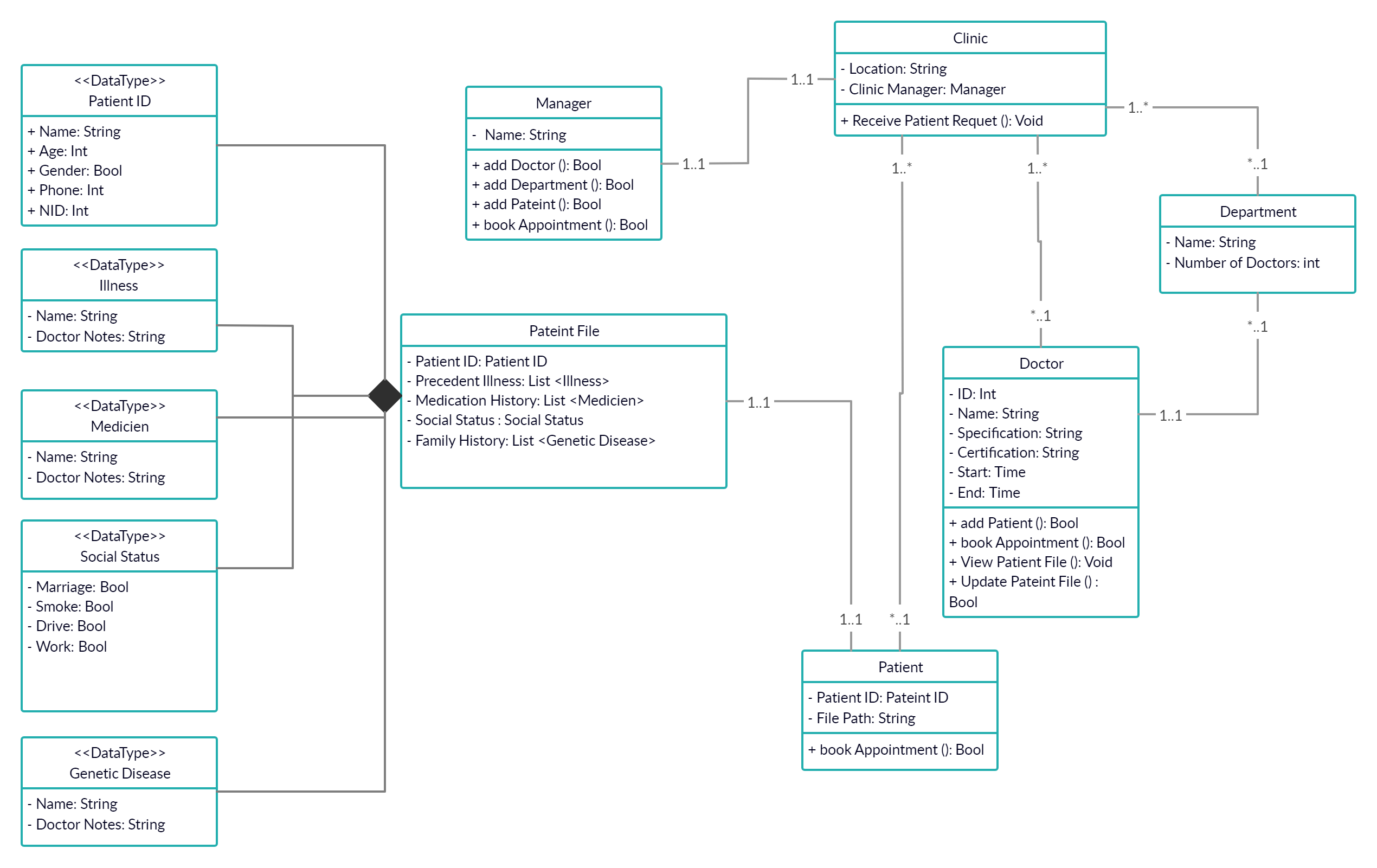
### Security

The Clinic server must prevent unauthorized access.

* Manager has (read, write, update, delete) access on departments, doctors data.
* Manager has only (read, write) access on specific patients’ data – just the patient’s ID data.
* Doctor has (read, write, update, delete) access on all patient’s data.
* Patient has (read) access on doctors’ data.

# Appendix A: Analysis Models

## A-1: Class Diagram



# Appendix B: Templet of Patient’s File (XML File)

<?xml version="1.0"?>

**<PatientFile>**

**<PatientID>**

<ID></ID>

<Name></Name>

<Age></Age>

<Gender></Gender>

<Phone></Phone>

<NID></NID>

**</PatientID>**

**<MainComplaint>**

<Name></Name>

<Onset></Onset>

<Frequency></Frequency>

<LastOccur></LastOccur>

<ReduceFactors></ReduceFactors>

<TriggerFactors></TriggerFactors>

**<MainComplaint>**

**<PrecedentIllness>**

<Illness>

<Name></Name>

<Notes></Notes>

</Illness>

<Illness>

<Name></Name>

<Notes></Notes>

</Illness>

**</PrecedentIllness>**

**<MedicationHistory>**

<Medicien>

<Name></Name>

<Notes></Notes>

</Medicien>

<Medicien>

<Name></Name>

<Notes></Notes>

</Medicien>

**</MedicationHistory>**

**<SocialStatus>**

<Marriage></Marriage>

<Smoke></Smoke>

<Drive></Drive>

<Work></Work>

**</SocialStatus>**

**<FamilyHistory>**

<GeneticDisease>

<Name></Name>

<Notes></Notes>

</GeneticDisease>

<GeneticDisease>

<Name></Name>

<Notes></Notes>

</GeneticDisease>

**</FamilyHistory>**

**</PatientFile>**