Table of Contents

[1. Product backlog - 2 -](#_Toc18665684)

[1.1. Requirements Engineering - 2 -](#_Toc18665685)

[1.2. Introduction: - 2 -](#_Toc18665686)

[Current System description. - 2 -](#_Toc18665687)

[Object Oriented Analysis. - 2 -](#_Toc18665688)

[Object Oriented Design. - 2 -](#_Toc18665689)

[Testing. - 2 -](#_Toc18665690)

[Implementation. - 2 -](#_Toc18665691)

[2. Introduction - 3 -](#_Toc18665692)

[2.1.1 Purpose of the system - 3 -](#_Toc18665693)

[2.1.2 Scope of the system - 3 -](#_Toc18665694)

[2.1.3 Objectives - 3 -](#_Toc18665695)

[2.1.4 Overview. - 4 -](#_Toc18665696)

[2.2 Current System - 4 -](#_Toc18665697)

[2.3 Proposed System - 5 -](#_Toc18665698)

[2.3.1 Overview - 5 -](#_Toc18665699)

[2.3.2 Functional requirements - 5 -](#_Toc18665700)

[2.3.3 Non-functional requirements - 6 -](#_Toc18665701)

[3 Object Oriented Analysis. - 7 -](#_Toc18665702)

[3.1 System model. - 7 -](#_Toc18665703)

[3.1.1 Activity Diagrams - 7 -](#_Toc18665704)

[4 Use case Diagrams - 16 -](#_Toc18665705)

[5 Class diagrams - 19 -](#_Toc18665706)

[5.1 Analysis Class Diagrams - 19 -](#_Toc18665707)

[5.2 Design Class Diagram. - 22 -](#_Toc18665708)

[6 Technical Documentation - 23 -](#_Toc18665709)

[7 System testing - 33 -](#_Toc18665710)

[8 Dump screens. - 34 -](#_Toc18665711)

[9 User Documentation - 37 -](#_Toc18665712)

[10 Conclusion - 38 -](#_Toc18665713)

[11 REFERENCES - 39 -](#_Toc18665714)

## Product backlog

## Requirements Engineering

## Introduction:

* + 1. Purpose.
    2. Scope.
    3. Objectives.
    4. Definitions.
    5. Acronyms and abbreviations.
    6. Overview.

## Current System description.

* 1. Proposed System description:
     1. Overview.
     2. Functional requirements.
     3. Nonfunctional requirements.
  2. Glossary/data dictionary.

## Object Oriented Analysis.

* 1. System models.
     1. Activity diagrams.
     2. Use case diagrams.
     3. Analysis class diagrams.

## Object Oriented Design.

* 1. Design class diagrams.
  2. Sequence diagrams.

## Testing.

## Implementation.

## Introduction

The already existing system is a manually operated one, where everything is recorded manually using pen and paper. A lot of paper files are created on daily basis as normal business is undertaken.

## Purpose of the system

This Health Care Information System is going to focus on the data management functions of Centurion Hospital located in Masvingo. The data management department functions include recording clients’ details and their health data. That is: full name, age, phone number, occupation, City, Disease, date of admission, date of discharge, department referred to for treatment, bill.

## Scope of the system

Project is aiming to correct the problems that are being faced by the already existing system. The already existing system is a manual system, where there is no electronic capturing, processing or storage of data, therefore, we want to develop a system that does all the aforementioned tasks electronically (using the computer). The whole system shall be broken down into small modules so that each individual problem will be handle separately and completely. The solved problems/modules will then be interlinked to make up a complete system that achieve all the set objectives. These modules will include data capturing and storage, data processing i.e. searching, editing, viewing, listing, deleting etc.

## Objectives

* Increased speed in capturing clients’ details through computerisation.
* Faster processing of data, which means capturing or accessing data shall be fast.
* Reliable data capturing and storage, no more hustles of reading different kinds of handwritings, dirt record (rubbed/cancelling).
* Security and identification, only authorised persons can do the capturing, amending, viewing and deleting of data.
* Searching and viewing a specific individual’s record.
* Producing a list of all available records.
* Having a backup file will help if there is any form of data loss. Which might be caused by various things, known or unknown e.g. deleting purposely or by mistake, crushing of files or computers.

## Overview.

Basically, the existing system is primitive, slow, inefficient and has a lot of loopholes. Everything solely depends on the state/mood of the workers i.e. the tidiness of the records, speed of recording and access, safety of records in the file room etc. This is because everything is manually done, therefore computerization takes over more than half of those human done tasks which will bring very significant changes because computers don’t get tired or moody. Capturing and access will be fast, efficient searching and reference etc.

## Current System

Centurion Hospital is currently operating the old school way, no computer or computer system is in use. All the data capturing is done by manually recording health data on paper using pens and the records are filed in arch lever or flat files for reference, future use and safe keeping. When a client comes, he/she goes to the data capture clerk who takes down data about the client, i.e. full name, age, phone number, occupation, City, Disease, date of admission, date of discharge, department referred to for treatment, bill. This information serves the purpose of helping the doctors(s) and nurse(s) understand the patient’s history, current situation and health state for current and future treatments. Upon any need, the file room workers have to search for every needed record in the file room and give to the staff members who requested, the doctor or nurse who then make amendments or add information into the file if necessary. After use, the file is then taken back to its location in the file room. Periodically these file room workers must organize all the records in the file room in a certain way that allows them to search and find files quickly whenever they are needed.

## Proposed System

## Overview

The Health Care Information System software will be designed to eliminate redundancy, slow data capture and access service delivery, accountability issues upon mistake/errors, safety and security of data about clients which all will improve the quality of work and effectiveness of the business.

## Functional requirements

Functional engineering describes the intended functionality of the system and how it is going to work. It is meant to assure that the design is adequate to develop the intended system and that it meets the user expectations. The system has access levels which are:

1. **Administrator.**

* Creates users.
* Deletes users.
* Edits user information.

1. **Supervisor.**

* Edits existing records.
* Deletes unwanted records.
* Views all records in database

1. **Data capture clerk**

* Creates new records
* Searches records

In the Centurion Hospital domain, the functional units are as follows:

* Casualty
* Accounts
* Inpatients
* Outpatients
* Pharmacy
* Surgery
* Maternity
* Radiology

## Non-functional requirements

The system shall be:

* Reliable – it shall allow accessing all the records already entered whenever they are needed.
* User friendly – the interface will be easy to use and understand.
* Fast – there is improved speed on all tasks since the is use of computers and efficient programs.
* Secure – data about clients will be electronically stored and it will be protected from unauthorized access or theft.
* Compatible with any windows operating system, the system is lite and not resource hungry.
* Better presentation of records, a good layout on the view and print and in the end neat lists and or records are produced.

## Object Oriented Analysis.

This is a process that groups items that interact with one another, typically by class, data or behaviour, to create a model that accurately represents the intended purpose of the system as a whole.

## System model.

Here I shall use models to conceptualize and construct the system.

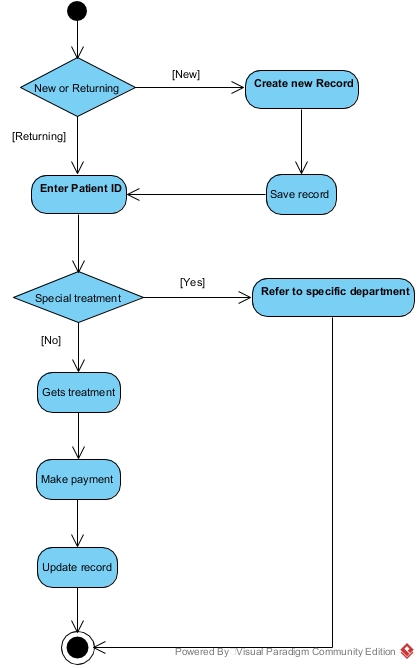
## Activity Diagrams

* Centurion Hospital domain has functional units which include: Casualty, Accounts, Inpatients, Outpatients, Pharmacy, Surgery, X-Ray, Maternity.
* Accounts is the first unit a patient gets to and for that reason it is different in functionality. All the other functional units have the same functional setup, differing only on the service being provided.

* 1. **Casualty Department.**

This is the arrival point for all patients, new patients will have new records created for them here. If the patient needs specific attention e.g. x-ray, surgery, etc. the patient is referred thereon otherwise the patient receives treatment and the record is updated.

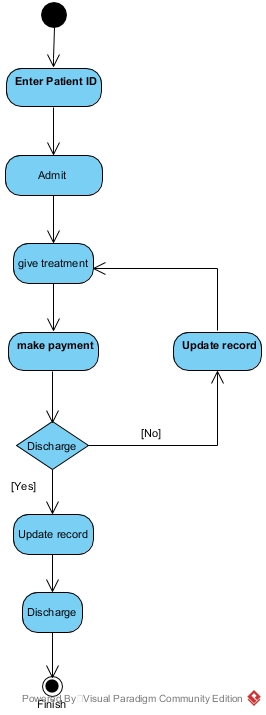
***Treatment***



* 1. **Inpatients**

Every patient referred to this department is to be admitted. The patient then gets treatment, makes payment for the drugs and admission. The patient will then be checked if he/she is fit for discharge, if not then treatment continuous otherwise record is updated and the patient is discharged.

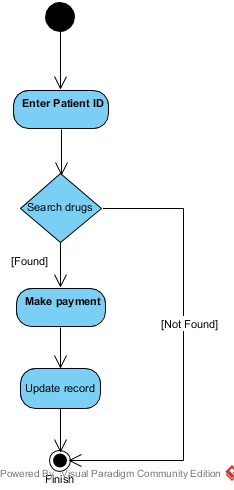
***Admission***



* 1. **Pharmacy**

After a patient ID is entered, the prescribed drugs are searched in the system. If they are available then payment is made and the patient is given the drugs, if not found then the transaction is terminated thereon.

***Dispensing drugs***



* 1. **Outpatients**

At the outpatient’s department, patients who were once treated will be coming for review according to previous recommendations. So a patient ID is entered, the patient gets the review and also gets the new recommendation, them the record is updated.

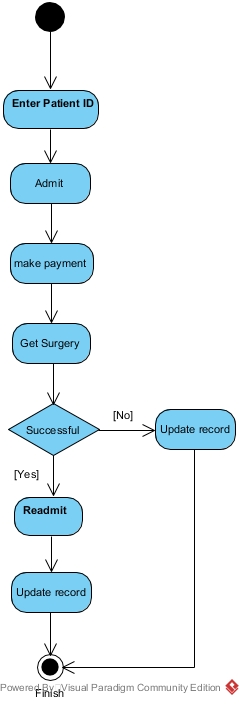
***Reviewing returning patient***



* 1. **Surgery**

This department also admits patients. So a patient ID is entered and a patient is admitted, he/she makes payment for the admission and the surgery to be conducted. Thereafter, a surgery is done, If the surgery is not successful the process is terminated. If it is successful, the patient is taken back to the ward he/she was admitted and gets treatment. Record is updated and the process ends.

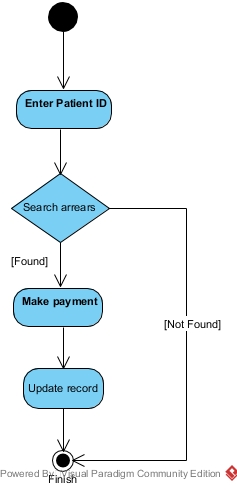
***Operation***



* 1. **Accounts**

A patient ID is entered, arrears are searched. If there are non-found, the process is terminated, otherwise payment of the reflected amount is made and the record is updated and the process ends.

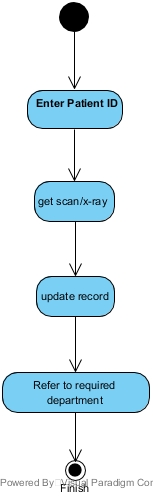
***Payment***



* 1. **Radiology**

A patient ID is entered, a scan or x-ray is conducted then the record is updated. The patient is then referred to the department that has to do with the result of the scan or x-ray and the process ends.

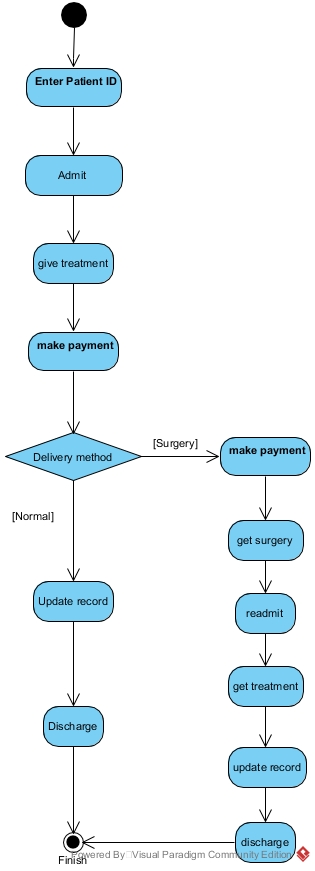
***Scan/X-Ray***



* 1. **Maternity**

A patient ID is entered, a patient is admitted, gets treatment and makes payment. If the patient has a normal delivery, record will be updated and she will be discharged when fit. Otherwise the patient has to pay for surgery, get the surgery, gets treatment, gets back to her ward till she is fit, the record is updated and then she will be discharged.

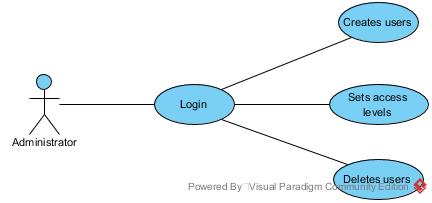
***Child delivery***



## Use case Diagrams

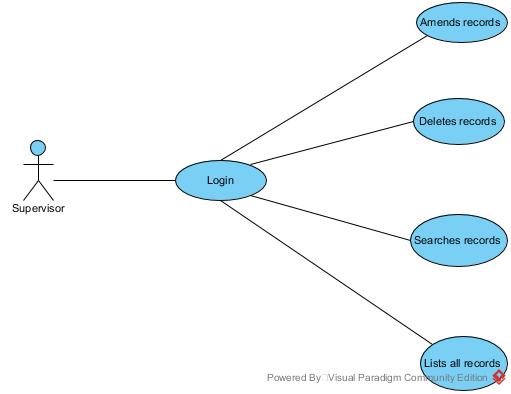
These are a representation of users’ interactions with the system that shows the relationship between the users and the different use cases in which the user is involved.

1. **The Administrator**



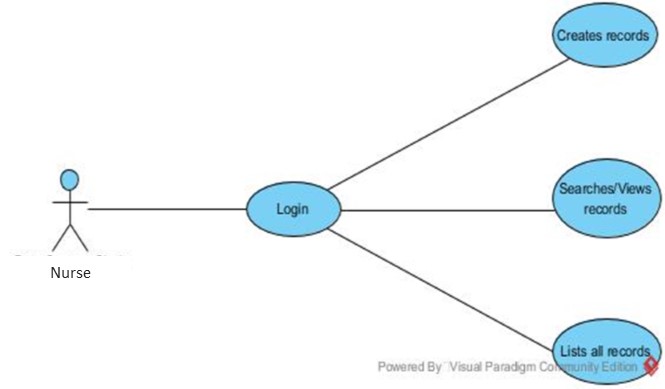
**Description:**

1. **Login** – the Administrator has to login in order to get into the system.
2. **Creates users** – the Administrator can create new users.
3. **Set access levels** – he/she can set the access levels of the created users, which determines what a user is privileged to do.
4. **Deletes users** – only the Administrator can delete users.
5. **The Supervisor**



**Description:**

1. **Login** – The Supervisor first has to login to get access into the system.
2. **Amends records** – he/she is the only person who can amend records if there were capturing mistakes or corrections needed.
3. **Deletes records** – only the supervisor can delete records.
4. **Searches records** – he/she can search individual records in the database.
5. **List all records** – he/she also has the privilege to list and view all records in the database.
6. **The Nurse**



**Description:**

1. **Login** – the nurse has to log in first in order to use the system.
2. **Creates records** – every time data about clients is captured, a new record is created. And it is only the nurse who can do that.
3. **Searches/views records** – the system allows him/her to search an individual record and view it.
4. **List all records** – the system also allows the nurse to list all records in the database file.

## Class diagrams

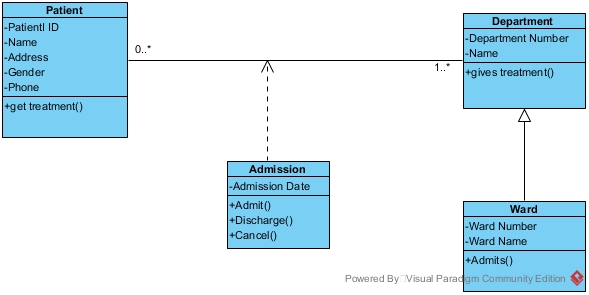
These are used to represent meaningful concepts in a problem domain. They show relationships between entities or classes of a system.

* Associations represent static relationships between classes, is reflected by a straight line.
* Generalization is a relationship in which one class (the child) is based on another class (the parent), has a triangle pointing to the superclass.
* Dependency relationship is a relationship in which one element, uses or depends on another element, are shown as a dashed line.

## Analysis Class Diagrams

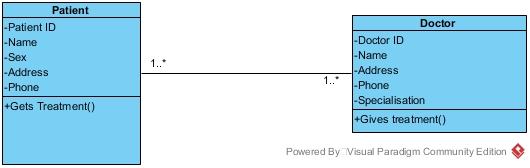
1. **Patient to Department.**

* A patient must have at least one department.
* A department may have may patients.
* Some department have wards for admitting patients.
* Admission depends on the patient and department.



1. **Patient to Doctor.**

* A patient must have at least one doctor.
* A doctor must have at least one patient.



1. **Doctor to Department.**

* A doctor must have one department.
* A department may have many doctors.



1. **Patient to Nurse.**

* A patient must have one nurse.
* A nurse must have at least one patient.



1. **Nurse to Department**

* A nurse must have one department.
* A department must have at least one nurse.

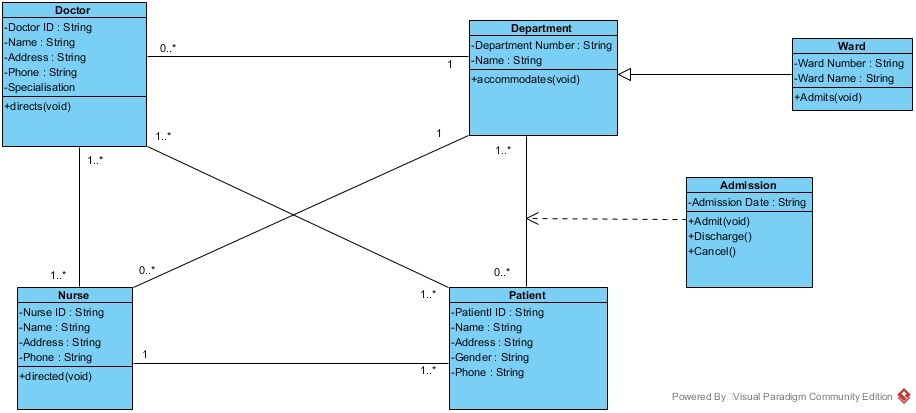


1. **Doctor to Nurse**

* A doctor must have at least one nurse.
* A nurse must have at least one doctor.



## Design Class Diagram.

****

## Technical Documentation

This part describes handling, functionality and architecture of the system we are developing. The language that was used in developing the system is C++. Below are snippets of the code following **CRUD**, which stands for Create, Read, Update, and Delete.

**Patient class**

#ifndef PATIENT\_H

#define PATIENT\_H

#include <string.h>

#include <string>

#include<stdlib.h>

#include<stdio.h>

#include<iomanip>

using namespace std;

class Patient {

int p\_ID;

int p\_age;

string fname;

string lname;

string occupation;

string p\_city;

string dis;

string addDate;

string relDate;

long double phnumber;

long double bill;

public:

Patient();

Patient(int pId, string name, string sname, string pOccu, string pCity, string disease, string admission, string discharge, long double phone, long double charge);

string getDept();

friend void NewPatient();

friend void EditPatient(Patient &pat);

friend void ViewPatient();

friend void DeletePatient();

friend void ViewAllPatients();

friend void NurseMenu();

virtual ~Patient();

};

#endif /\* PATIENT\_H \*/

* 1. **Creating a new patient record**

void NewPatient() {

Patient p; //ID,firstname,lastname,age,phnumber,occupation,city,disease,Adate,Rdate,bill,spec

cout << "Fill up the following details" << endl;

cout << "ID:" << endl;

cin >> p.p\_ID;

cout << "First Name:" << endl;

cin >> p.fname;

cout << "Last Name:" << endl;

cin >> p.lname;

cout << "Age:" << endl;

cin >> p.p\_age;

cout << "Phone No.:" << endl;

cin >> p.phnumber;

cout << "Occupation:" << endl;

cin >> p.occupation;

cout << "City:" << endl;

cin >> p.p\_city;

cout << "Disease/Symptoms:" << endl;

cin >> p.dis;

cout << "Admission date (DD/MM/YYYY):" << endl;

cin >> p.addDate;

cout << "Release date (DD/MM/YYYY):" << endl;

cin >> p.relDate;

cout << "Total Bill generated:" << endl;

cin >> p.bill;

cout << "Kind of Specialization required:" << endl;

string dept = p.getDept();

system("cls");

fstream patObj;

patObj.open("patient.txt", ios::app);

//ID,firstname,lastname,age,phnumber,occupation,city,disease,Adate,Rdate,bill,dept

patObj.seekp(0, ios::end);

patObj << p.p\_ID << endl << p.fname << endl << p.lname << endl << p.p\_age << endl << p.phnumber << endl << p.occupation << endl << p.p\_city << endl << p.dis << endl << p.addDate << endl << p.relDate << endl << p.bill << endl << dept << endl;

patObj.close();

cout << "\n\t\tNew Patient record created successfully!" << endl << endl << endl;

cout << "\n\n\n\nEnter 1 to return to Menu\nEnter 2 to Exit." << endl;

int c;

cin>>c;

if (c == 1) {

system("cls");

NurseMenu(); //back to Nurse Menu

} else

exit(0);

}

* 1. **Editing patient’s details**

void EditPatient(Patient &pat) {

int x;

cout << "\nEnter the ID of the Patient whose record you want to Edit:" << endl;

cin>>x;

fstream patObj("patient.txt");

patObj.seekg(0);

//ID,firstname,lastname,age,phnumber,occupation,city,disease,Adate,Rdate,bill,dept

string fname, lname, occupation, p\_city, dis, addDate, relDate, dept;

int p\_age, p\_ID, count = 0;

long double phnumber, bill;

while (patObj >> p\_ID >> fname >> lname >> p\_age >> phnumber >> occupation >> p\_city >> dis >> addDate >> relDate >> bill >> dept) {

if (p\_ID == x) {

cout << "1.ID:" << p\_ID << endl << "2.First Name:" << fname << endl << "3.Last Name:" << lname << endl << "4.Age" << p\_age << endl << "5.Phone No.:" << phnumber << endl << "6.Occupation:" << occupation << endl << "7.City:" << p\_city << endl << "8.Disease/Symptoms:" << dis << endl << "9.Admission date(DD/MM/YYYY):" << addDate << endl << "10.Release date(DD/MM/YYYY):" << relDate << endl << "11.Total Bill generated:" << bill << endl << "12.Kind of Specialization required:" << dept << endl;

count++;

break;

}

}

patObj.close();

if (count == 0) {

cout << "\nNo matching records found!" << endl;

exit(0);

}

//data transfer begins

fstream file("patient.txt");

file.seekg(0, ios::beg);

file.seekg(0);

int line;

ofstream tempObj;

tempObj.open("copy2.txt");

tempObj.seekp(0);

int skip = 0;

string fname1, lname1, occupation1, p\_city1, dis1, addDate1, relDate1, dept1;

int p\_age1;

long double phnumber1, bill1;

file.seekg(0);

file.seekg(0, ios::cur); //setting the pointer to beginning of file

while (file >> line >> fname1 >> lname1 >> p\_age1 >> phnumber1 >> occupation1 >> p\_city1 >> dis1 >> addDate1 >> relDate1 >> bill1 >> dept1) {

if (line != x) {

tempObj << line << endl << fname1 << endl << lname1 << endl << p\_age1 << endl << phnumber1 << endl << occupation1 << endl << p\_city1 << endl << dis1 << endl << addDate1 << endl << relDate1 << endl << bill1 << endl << dept1 << endl;

}

}

file.close();

tempObj.close();

remove("patient.txt");

int result;

char oldname[] = "copy2.txt";

char newname[] = "patient.txt";

result = rename(oldname, newname);

//data transfer ends

int edit;

cout << "\nEnter number corresponding to the Field you want to Edit:" << endl << endl;

cin>>edit; //ID,firstname,lastname,age,phnumber,occupation,city,disease,Adate,Rdate,bill,spec

switch (edit) {

case 1:

{

cout << "ID" << endl;

cin>>p\_ID;

break;

}

case 2:

{

cout << "First Name:" << endl;

cin>>fname;

break;

}

case 3:

{

cout << "Last Name:" << endl;

cin>>lname;

break;

}

case 4:

{

cout << "Age:" << endl;

cin>>p\_age;

break;

}

case 5:

{

cout << "Phone No.:" << endl;

cin>>phnumber;

break;

}

case 6:

{

cout << "Occupation:" << endl;

cin>>occupation;

break;

}

case 7:

{

cout << "City:" << endl;

cin>>p\_city;

break;

}

case 8:

{

cout << "Disease/Symptoms:" << endl;

cin>>dis;

break;

}

case 9:

{

cout << "Admission date (DD/MM/YYYY):" << endl;

cin>>addDate;

break;

}

case 10:

{

cout << "Release date (DD/MM/YYYY):" << endl;

cin>>relDate;

}

case 11:

{

cout << "Total Bill generated:" << endl;

cin>>bill;

break;

}

case 12:

{

cout << "Kind of Specialization required:" << endl;

dept = pat.getDept();

break;

}

default:

{

cout << "\nCome on!Enter a valid option." << endl;

break;

}

}

system("cls");

cout << right << setw(50) << setfill(' ') << "--Updated Record--" << endl << endl;

cout << "1.ID:" << p\_ID << endl << "2.First Name:" << fname << endl << "3.Last Name:" << lname << endl << "4.Age" << p\_age << endl << "5.Phone No.:" << phnumber << endl << "6.Occupation:" << occupation << endl << "7.City:" << p\_city << endl << "8.Disease/Symptoms:" << dis << endl << "9.Admission date(DD/MM/YYYY):" << addDate << endl << "10.Release date(DD/MM/YYYY):" << relDate << endl << "11.Total Bill generated:" << bill << endl << "12.Kind of Specialization required:" << dept << endl;

//updated data transfer

fstream newObj;

newObj.open("patient.txt", ios::app);

newObj.seekp(0, ios::end);

newObj << p\_ID << endl << fname << endl << lname << endl << p\_age << endl << phnumber << endl << occupation << endl << p\_city << endl << dis << endl << addDate << endl << relDate << endl << bill << endl << dept << endl;

newObj.close();

//after completion

cout << "\n\nEnter 1 to return to Menu\nEnter 2 to Exit." << endl;

int c;

cin>>c;

if (c == 1) {

system("cls");

SupervisorMenu(); //back to Supervisor Menu

} else

exit(0);

}

* 1. **Viewing a patient’s details**

void ViewPatient() {

int x;

cout << "\nEnter the ID of the Patient whose record you want to View:" << endl;

cin>>x;

fstream patObj("patient.txt");

patObj.seekg(0);

string fname, lname, occupation, p\_city, dis, addDate, relDate, dept;

int p\_age, p\_ID, count = 0;

long double phnumber, bill;

while (patObj >> p\_ID >> fname >> lname >> p\_age >> phnumber >> occupation >> p\_city >> dis >> addDate >> relDate >> bill >> dept) {

if (p\_ID == x) {

cout << "1.ID:" << p\_ID << endl << "2.First Name:" << fname << endl << "3.Last Name:" << lname << endl << "4.Age" << p\_age << endl << "5.Phone No.:" << phnumber << endl << "6.Occupation:" << occupation << endl << "7.City:" << p\_city << endl << "8.Disease/Symptoms:" << dis << endl << "9.Admission date(DD/MM/YYYY):" << addDate << endl << "10.Release date(DD/MM/YYYY):" << relDate << endl << "11.Total Bill generated:" << bill << endl << "12.Kind of Specialization required:" << dept << endl;

count++;

break;

}

}

patObj.close();

if (count == 0) {

cout << "\nNo matching records found!" << endl;

exit(0);

}

cout << "\n\nEnter 1 to return to Menu\nEnter 2 to Exit." << endl;

int c;

cin>>c;

if (c == 1) {

system("cls");

NurseMenu(); // back to Nurse Menu

} else

exit(0);

}

* 1. **Deleting a patient’s details**

void DeletePatient() {

int x;

cout << "\nEnter the ID of the Patient whose record you want to Delete:" << endl;

cin>>x;

fstream patObj("patient.txt");

patObj.seekg(0);

//ID,firstname,lastname,age,phnumber,occupation,city,disease,Adate,Rdate,bill,dept

string fname, lname, occupation, p\_city, dis, addDate, relDate, dept;

int p\_age, p\_ID, count = 0;

long double phnumber, bill;

while (patObj >> p\_ID >> fname >> lname >> p\_age >> phnumber >> occupation >> p\_city >> dis >> addDate >> relDate >> bill >> dept) {

if (p\_ID == x) {

cout << "1.ID:" << p\_ID << endl << "2.First Name:" << fname << endl << "3.Last Name:" << lname << endl << "4.Age" << p\_age << endl << "5.Phone No.:" << phnumber << endl << "6.Occupation:" << occupation << endl << "7.City:" << p\_city << endl << "8.Disease/Symptoms:" << dis << endl << "9.Admission date(DD/MM/YYYY):" << addDate << endl << "10.Release date(DD/MM/YYYY):" << relDate << endl << "11.Total Bill generated:" << bill << endl << "12.Kind of Specialization required:" << dept << endl;

count++;

break;

}

}

patObj.close();

if (count == 0) {

cout << "\nNo matching records found!" << endl;

exit(0);

}

cout << "Are you sure you want to delete this record?\n1.YES\n2.NO" << endl;

int n;

cin>>n;

if (n == 1) {

fstream file("patient.txt");

file.seekg(0, ios::beg);

file.seekg(0);

int line;

ofstream tempObj;

tempObj.open("copy2.txt");

tempObj.seekp(0);

int skip = 0;

string fname1, lname1, occupation1, p\_city1, dis1, addDate1, relDate1, dept1;

int p\_age1;

long double phnumber1, bill1;

file.seekg(0);

file.seekg(0, ios::cur); //setting the pointer to beginning of file

while (file >> line >> fname1 >> lname1 >> p\_age1 >> phnumber1 >> occupation1 >> p\_city1 >> dis1 >> addDate1 >> relDate1 >> bill1 >> dept1) {

if (line != x) {

tempObj << line << endl << fname1 << endl << lname1 << endl << p\_age1 << endl << phnumber1 << endl << occupation1 << endl << p\_city1 << endl << dis1 << endl << addDate1 << endl << relDate1 << endl << bill1 << endl << dept1 << endl;

}

}

file.close();

tempObj.close();

remove("patient.txt");

int result;

char oldname[] = "copy2.txt";

char newname[] = "patient.txt";

result = rename(oldname, newname);

system("cls");

cout << "\n\t\tRecord successfully deleted" << endl;

}

cout << "\n\nEnter 1 to return to Menu\nEnter 2 to Exit." << endl;

int c;

cin>>c;

if (c == 1) {

system("cls");

SupervisorMenu();

} else

exit(0);

}

* 1. **Viewing all patients**

void ViewAllPatients() {

fstream Obj("patient.txt");

Obj.seekg(0);

cout << right << setw(70) << setfill(' ') << "PATIENT MENU" << endl << endl;

string fname, lname, occupation, p\_city, dis, addDate, relDate, dept;

int p\_age, p\_ID, count = 0;

long double phnumber, bill;

while (Obj >> p\_ID >> fname >> lname >> p\_age >> phnumber >> occupation >> p\_city >> dis >> addDate >> relDate >> bill >> dept) {

count++;

cout << right << setw(50) << setfill(' ') << "Patient Record:" << count << endl;

cout << "1.ID:" << p\_ID << endl << "2.First Name:" << fname << endl << "3.Last Name:" << lname << endl << "4.Age" << p\_age << endl << "5.Phone No.:" << phnumber << endl << "6.Occupation:" << occupation << endl << "7.City:" << p\_city << endl << "8.Disease/Symptoms:" << dis << endl << "9.Admission date(DD/MM/YYYY):" << addDate << endl << "10.Release date(DD/MM/YYYY):" << relDate << endl << "11.Total Bill generated:" << bill << endl << "12.Kind of Specialization required:" << dept << endl << endl;

}

Obj.close();

if (count == 0) {

cout << "\nNo matching records found!" << endl;

exit(0);

}

cout << "\n\nEnter 1 to return to Menu\nEnter 2 to Exit." << endl;

int c;

cin>>c;

if (c == 1) {

system("cls");

SupervisorMenu();

} else

exit(0);

}

## System testing

In this chapter we’re testing the designed system to evaluate the system's compliance with its specified requirements.

* 1. **Login**

|  |  |  |
| --- | --- | --- |
| Test type | date | test |
| login | 24/07/19 | true |

|  |  |  |
| --- | --- | --- |
| Test type | description | notes |
| 1.user attempts to login | * A login page popped up * User is prompted to login. |  |
| 2.user attempts to login | * User decides to login | * Credentials were found |
| 3.user attempts to login | * A login page popped up * User decides to create an account | * Account was created successfully |

* 1. **Search for a patient**

|  |  |  |
| --- | --- | --- |
| Test type | date | test |
| View patient | 08/08/19 | true |

|  |  |  |
| --- | --- | --- |
| Test type | description | notes |
| Select View Patient | * A menu pops up, prompts user to enter patient ID | * record found * No errors found |

* 1. **Delete Patient**

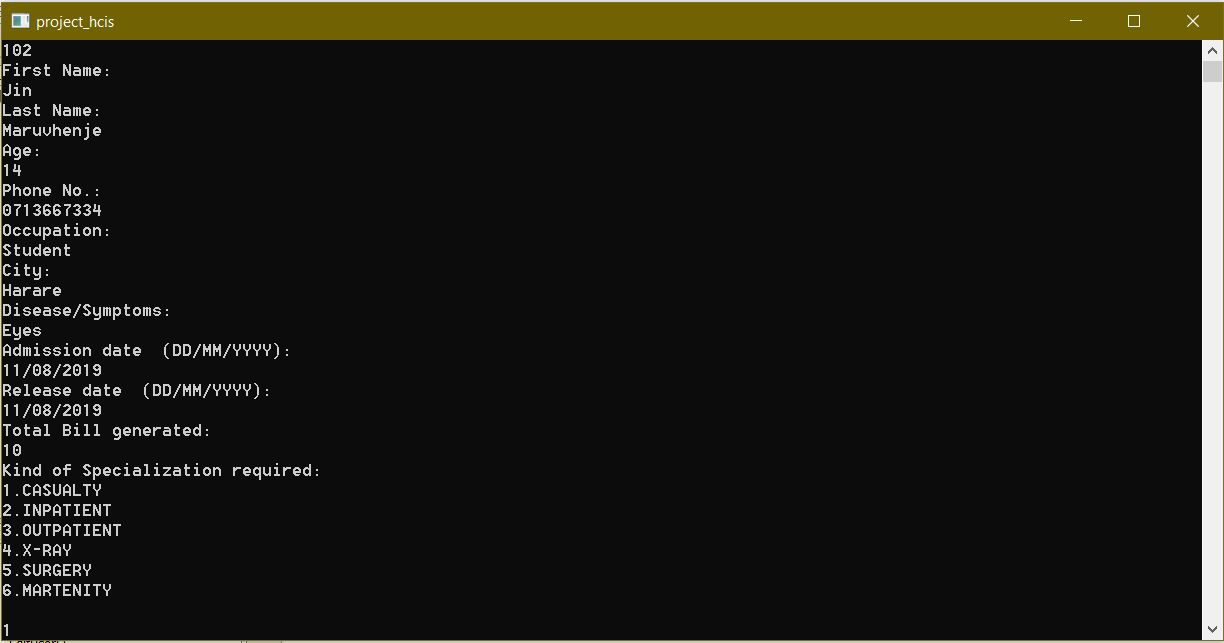
|  |  |  |
| --- | --- | --- |
| Test type | date | test |
| Select delete patient | 09/08/19 | true |

|  |  |  |
| --- | --- | --- |
| Test type | description | notes |
| Select delete patient | * A menu pops up, prompts user to enter patient ID | * Patient was deleted * No errors found |

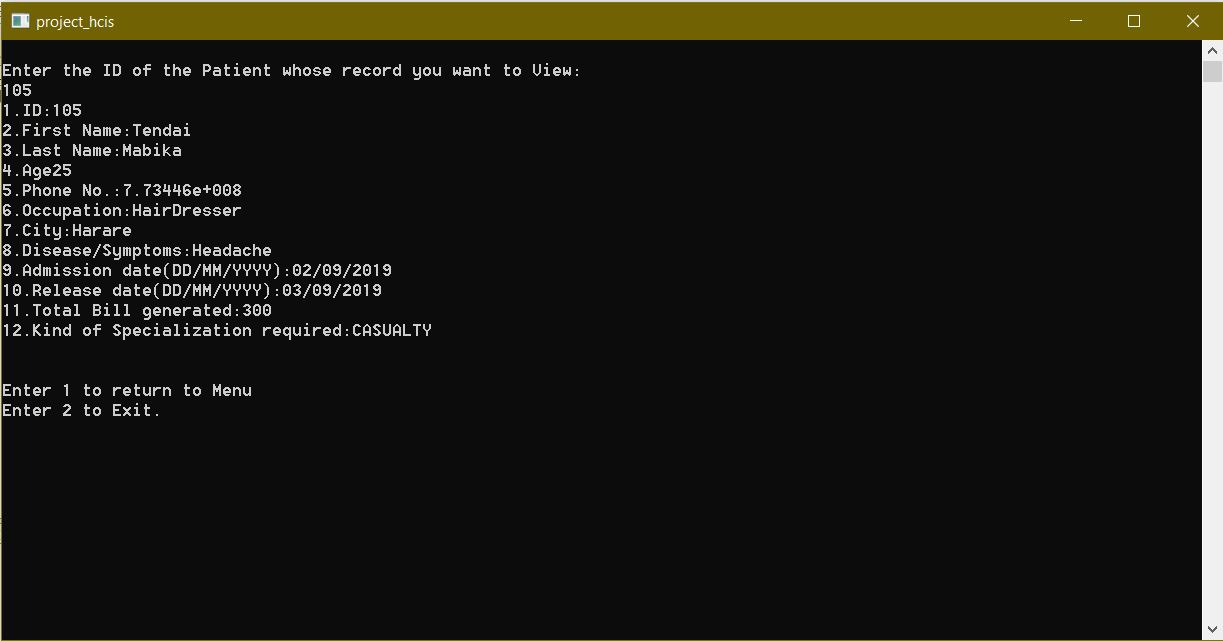
## Dump screens.

These show the exact user interface of the designed system by means of screenshots.

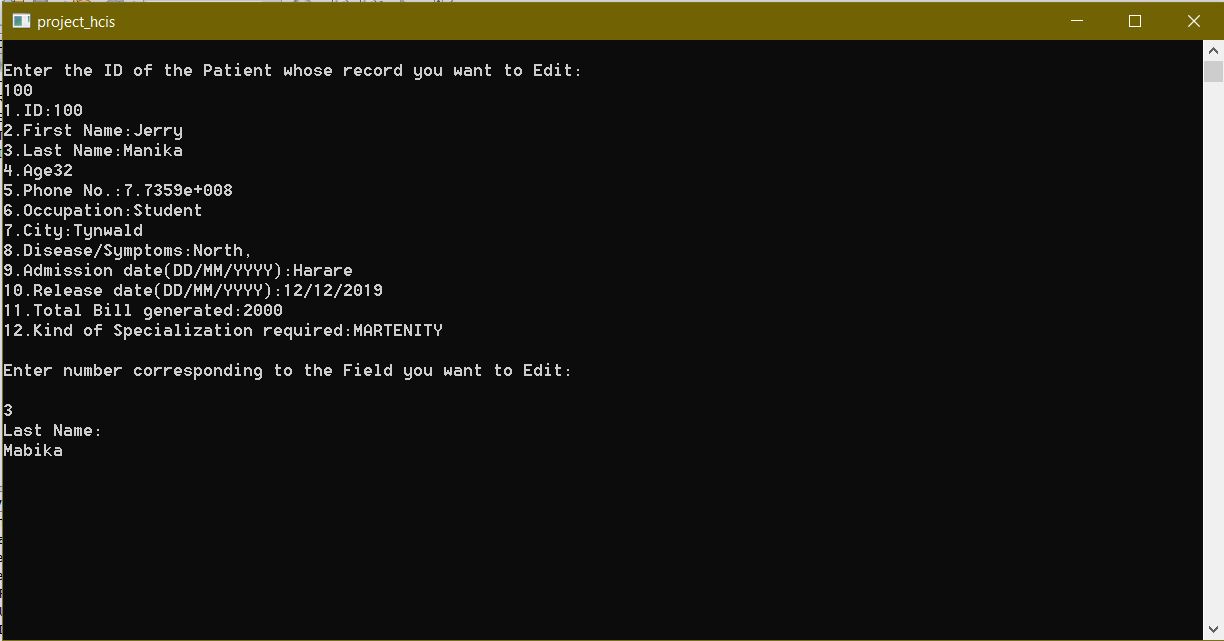
* 1. Creating a Patient record



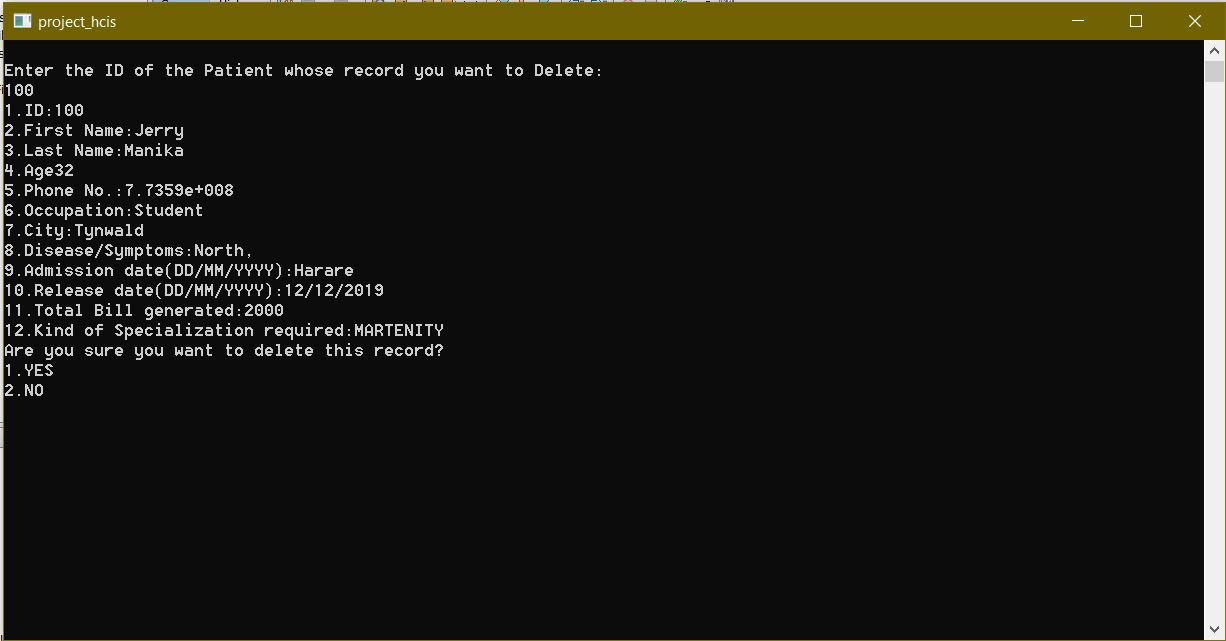
* 1. Viewing a patient record from file



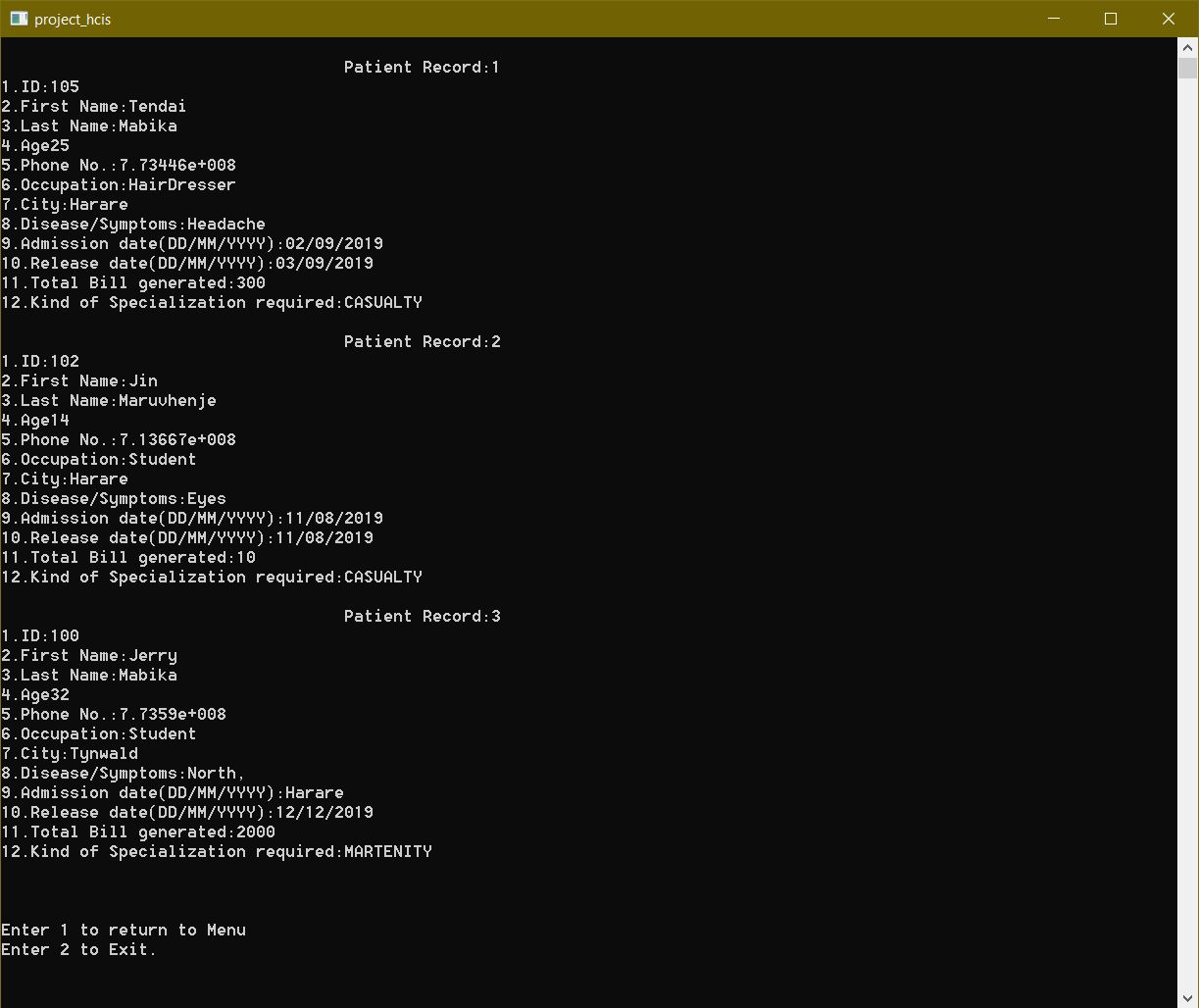
* 1. Editing a record in the file



* 1. Deleting a record from file



* 1. Viewing all patients



## User Documentation

* The initial login credentials are provided in the disk (CD) that accompanies this document.
* Please refer to the read me.txt file on the disk.

The system was designed with access level, the determine what a user can do *(see Use Case Model).*

1. Administrator – has access level 1.
2. Supervisor – has access level 2.
3. Nurse – has access level 3.

Only the Administrator can create users, after users are created and access levels are given to them, then can they start using the system.

## Conclusion

The designed systems has managed to meet most of the requirements, which includes

* + Security by the use of passwords and access level.
  + Computerized capturing of data.
  + Storing captured date into the computer.
  + Fast and efficient service since computer can process faster than humans.
  + Records are stored in a computer file that can be backed up and stored anywhere safe i.e. online, on flash drives or any form of storage device available.
  + Any other requirements that were not met by the system, it was due to limited time and information during the information gathering process.

## REFERENCES

* 1. <https://www.healthcatalyst.com/insights/healthcare-information-systems-past-present-future>
  2. Larry Grandia (2014): Healthcare Information Systems: A Look at the Past, Present, and Future
  3. <https://www.healthcarefirst.com/blog/4-types-healthcare-information-systems/>