# Design Specification Document: Hospital Management Software

Abdulaziz Al-Mass Abdullah Al-Noomes Faisal Al-Ghanim Faisal Al-Omar
Mohammad AlToorah Yousef Al-Failakawi Asmaa Al-Hajeri

22nd of November, 2021

**Abstract**

This Document will act as Design Specification Document(DSD) for the virtual Hospital Management System(VHMS). It will discuss the required features and how they will be implemented.

## Contents

## 1 Introduction

This project will consist in building a virtual Hospital Management System(VHMS). The VHMS will be able to simulate the administrative functions of the hospital and manage them autonomously. The aim is to create a system that requires as little human intervention as possible.
And it will have the following features:

1. Patient appointment & Scheduling
2. Automated Billing
3. employee tracking
4. Stock management

Further elaboration can be found in each items respective section under Requirement Specification.

## 2 Requirements

This section will discuss the requirements for VHMS in detail, clearly outlining the scope of the project.

## 2.1 User Interface

**Requirements:**

The user interface of the VHMS will be split into three parts: patient, Doctor, and master interface. Each interface will be further broken up into parts, see fig 2 for how UI will be split up.

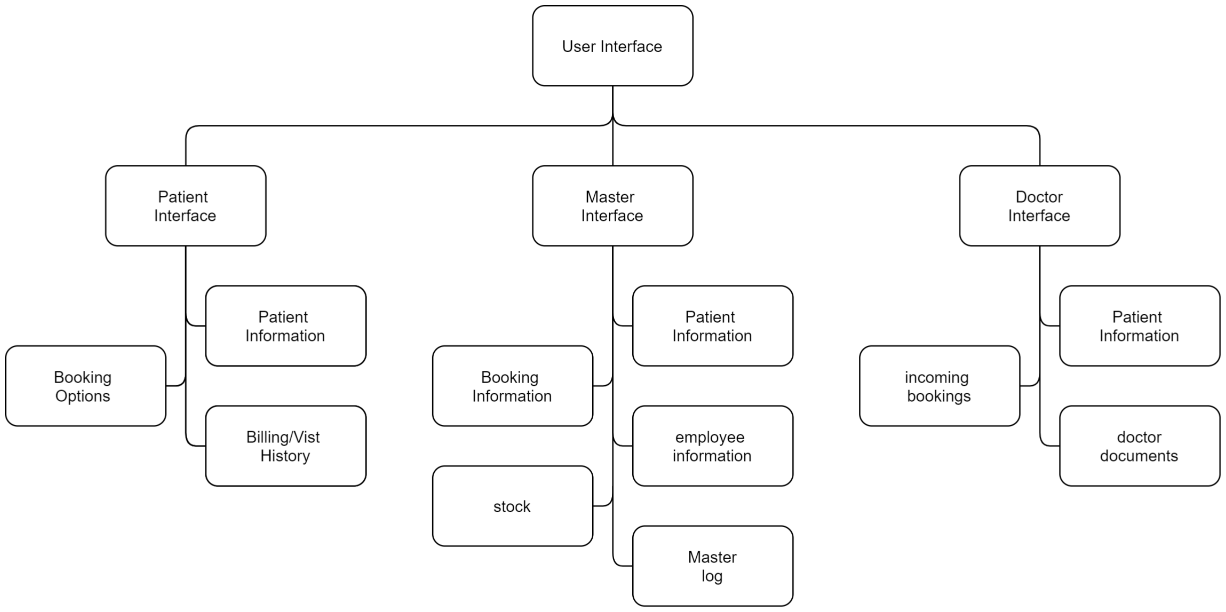

Figure 1: VHMS UI outline

### Patient Interface

As the figure above shows, the patient interface will be split into three main parts:

- Patient Information: This section will consist of all information attainable in regard to the patient. For example, things such as medication the patient might be on, chronic conditions, medical records(such as x-rays, blood tests, etc.), and any other pieces of information that would be relevant to the doctor to make medical evaluations.
- Booking Options: A booking page will be available for the patient. This page will show patients/visitors what times they can come in for an appointment.
- Billing/Visit History: Patients will have access to a timeline of visits. This timeline will show information such as: purpose of visit, any medication prescribed in the visit, and an itemized list of all things the patient is billed for in the visit.

### Master Interface

The master interface is the most expansive of the three interfaces, it will consist of five major parts:

- Patient Information: Patient information in the master will differ from the one available to patients. The master patient information will show a list of all patients where all the information (described in patient interface 2.1) can be accessed and edited.
- Booking Information: In this page persons with access to the master can see all bookings, edit them, and assign new booking if need be.
- Employee information: This page will host a lost similar to that of patient information page, but, it will have information regarding employees, such as: static info like; name, role, rank, etc.) and more 'dynamic' info like: patients and wards the employee is assigned to, tools and equipment used or is being used by the employee, etc.
- Stock: Stocks will be list of all items needed for running the hospital. stocks will be split into few parts, such as: medication, medical supplies, cleaning supplies, beds and rooms will fall under stock, and other miscellaneous items that are need for the running of the hospital. further details about the stock system can be found in the stock section found in 2.6
- Master log: The log will be record of all changes happening in the software. Any edits, additions, removals of anything will be seen in the log with the time and file edited listed.

### Doctor Interface

The doctor interface will display the same info as patient information interface with some of the edit access of the master, and will consist of three parts:

- Patient information: the doctors Patient information page will be similar to that of the master( descried above in 2.1), with notable exception that the doctor will only have access to the patients that they have an appointment with or ones that are assigned to them.
- incoming bookings: This page will show the doctor all his appointments and with which patients.
- doctor documents: doctor documents will consist of paper work the doctor might want to access, such as prescribing x-rays, MRIs, blood tests or other producers that require paper work.

### Preliminary Study:

The current consideration is to build the ui in either SceneBuilder, with JavaFX libary, or Codename One. further investigation in each programs needs to be made before a final decisions is made in which program to build in.

## 2.2 Virtual hospital simulation

**Requirements:**

The virtual hospital simulation will simulate a busy day at the hospital. This will act as a way to test and demonstrate our code/project.

**Preliminary Study:**

The current consideration for implementing the simulation would be a separate program that will run on top of our code and it will mimic patients and doctors using the our program.

## 2.3 Patient appointment & Scheduling

**Requirements:**

Patient appointment & Scheduling system will help in organizing appointments and selecting the best doctors for patient to see. This system should reduce time and make running hospital more efficient

- Patient information is collected in order to determine most suitable doctor/department to assign the patient.
- Fixing the schedule based on the availability of the supplier
- Automated reminders to both the patient and physician.

**Preliminary Study:** ]

The best method for implementing this section has not yet been identified.

## 2.4 Automated Billing

**Requirements:**

The automated billing system will Handel all required billing with as little manual input as possible. This will include:

- invoices
- payment collection
- approvals and provisioning.

**Preliminary Study:**

The best method for implementing this section has not yet been identified.

## 2.5 employee tracking

**Requirements:**

Employee tracking will be an integral part of our system. This system will keep track of:

- Employee Role
- Employee work hours
- Employee rank
- And other times yet to be specified

**Preliminary Study:**

The best method for implementing this section has not yet been identified.

## 2.6 Stock and inventory management

**Requirements:**

The stock and inventory system will keep track of all items, and some non items, used in the hospital.See fig2for breakdown of the stock and inventory system:

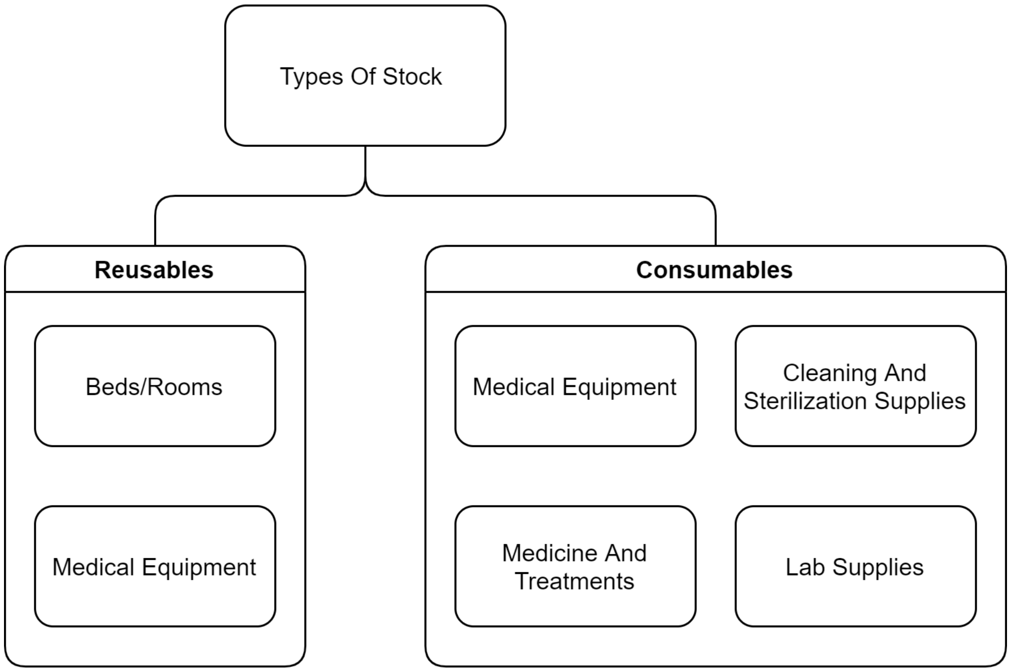

Figure 2: Types of stocks breakdown

As the figure 2 shows there two types of stocks the will be tracked: consumables and reusables.

**Consumables**

Consumables will be items that are of limited in use. They will be the items that are discarded after use and needed constant replenishment.

These items will include:

- Medical equipment
- Medicine And Treatments
- Lab Supplies
- Cleaning And sterilization Supplies

**Reusables**

Reysables will be items that see long term use

These items will include:

- Beds/Rooms
- Medical equipment(These differ from the one mentioned in the consumables section)

**Preliminary Study:**

A class, or multiple if deemed necessary, will be created. Each item will be stored as its own object.

note: changes and tweaks my be applied throughout the development cycle that could alter some of requirements.

## 3 Implementation

This section will be concerned with documenting the architecture and Implementation of VHMS.

# 4 Managing Neurodivergencies

This section is solely dedicated to Asmaa Al-Hajeri's project, and is not directly connected with the project described above.

## 4.1 App description:

A productivity app that incorporates gaming into everyday tasks. A character is custom created by the user, the user then goes on to add their to-do list with all their tasks (different to-do lists are provided if the user works and does chores) After the user chooses a task to start, the character starts to work on a task of their own (can be an artwork that's shown at the end of the day or something else) Once the user finishes their task (after a certain amount of time or manually) they check the task as "done!" then the user is able to spend their break listening to music with their character, check on their character's plants, make a meal with their character (recipes can be provided if needed) Then the cycle repeats until all the tasks are completed. The user is also allowed to create a certain schedule, meaning that certain tasks are repeated every time the user wants it to be (users can add their own recipes to be repeated on certain days of the week, they can also add a playlist for them to listen to on their breaks, they can add facts they know about the pets/plants they own so when the online feature is added, other people also know these facts).

## 4.2 How does it help the community:

The target demographic is neurodivergent people. Due to their struggles of finding a good incentive to start and finish tasks, this app rewards them after every task while their in-game character takes a break with them.

## 4.3 Algorithm:

**Main algorithm:**

1. Use the Scanner class to get the input from the user
2. Use the while loop to keep taking input from the user when the user wants to add a task, add a fact, or add a recipe
3. If the user chooses to add a recipe, the user is asked if they would like to categorize it as a "breakfast", "lunch", "dinner" meal, or a "snack". Otherwise, store the recipe as a "meal"
4. Once the user chooses to add a recipe, the user is asked to submit the number of steps it takes to complete the recipe and then is asked to enter the steps based on the number assigned

5. If the user chooses to add a task, the user is asked if the task is “work” related or “home” related or “self” related, work tasks include “respond to emails” and “submit assignments”, home related tasks include “do the dishes” and “cook”, self related tasks include “read” and “watch a movie”
6. Once the user chooses the category the task belongs to, the user is asked how much time it would take for them to complete it. If the user sets an approximation, the user will be reminded about the task after the time for that task is over, otherwise the task will just be set without an approximate time of its completion
7. The user is then asked to choose how long the breaks between the tasks to be
8. Use the for loop to keep taking input from the user to take their to do list
9. Use Math.random and assign it to a bundle of welcoming messages for when the user first enters the app in the beginning of day
10. Use ArrayList in order to put a limit to the amount of characters written in a task
11. Create a method that stops the task from looping if the user enters “Done”
12. Set a timer that starts once step 10 is completed
13. The timer is set based on how long the user chose the break between the tasks to be.
14. Stop the timer once the set amount of time has passed and print the second task in the To-Do list

**Side algorithm:**

1. Use Array to encrypt messages for the user to decrypt and complete a quest
2. Use switch, case, and Array to create a mini TicTacToe game as a quest
3. Declare a plant object that can be planted by the user and grows over time
4. Declare a cat object that can play with the user and requires to be fed
5. Create an object that outputs random vocabulary and a Scanner object that takes the input of the user (word association)
6. Create a method that creates a randomised order of “verb,” “adjective,” “proper noun,” “common noun,” and “adverb” in a story where the user enters the wanted sentence part to complete the story
7. Create a URL object that and a URL method that accepts a hyperlink from music players set by the user

8. Create method that calculates the average amount of steps taken in the set amount of time

### 4.4 How to improve it in the future:

Add an online feature where people could visit other people's places and leave notes, water their plants, etc.