A logo of a planet

Description automatically generated

**INDIVIDUAL ASSIGNMENT**

**TECHNOLOGY PARK MALAYSIA**

**CT018-3-1-ICP**

**INTRODUCTION TO C PROGRAMMING**

**APU/D1F 2309 CS(CYB)/IT/CS(DF)/ME/CS(IS)/CS/CGD/CE/EEE/PE/ME**

**HAND OUT DATE: 2 April 2024**

**HAND IN DATE: 24 June 2024**

**WEIGHTAGE: 50%**

**INSTRUCTIONS TO CANDIDATES:**

1. Submit your assignment online in Moodle unless advised otherwise
2. Late submission will be awarded zero(0) unless Extenuating Circumstances (EC) are upheld
3. Cases of plagiarism will be penalized
4. You must obtain at least 50% in each component to pass this module

Table of Contents

[1. Introduction and Assumptions 3](#_Toc170139467)

[1.1. Objectives 3](#_Toc170139468)

[1.2. System Overview 4](#_Toc170139469)

[1.3. Assumptions 4](#_Toc170139470)

[2. Program Design 6](#_Toc170139471)

[2.1. Patient Module Design 6](#_Toc170139472)

[2.2. Doctor Module Design 8](#_Toc170139473)

[2.3. Staff Nurse Module Design 10](#_Toc170139474)

[2.4. Hospital Administrator Module Design 11](#_Toc170139475)

[3. Implementation 12](#_Toc170139476)

[3.1. Patient Implementation 12](#_Toc170139477)

[3.2. Doctor Implementation 16](#_Toc170139478)

[3.3. Prompts the user to enter a username and password. 25](#_Toc170139479)

[3.4. Staff Nurse Implementation 25](#_Toc170139480)

[3.4.1. Header File Explanation: staffnurse.h 25](#_Toc170139481)

[3.4.2. Main file of StaffNurse Explanation: staffNurse.c 26](#_Toc170139482)

[3.5. Hospital Administrator Implementation 28](#_Toc170139483)

[3.6. Database Access Implementation 32](#_Toc170139484)

[4. Sample Input/Output 36](#_Toc170139485)

[4.1. Patient I/O 36](#_Toc170139486)

[4.2. Doctor I/O 39](#_Toc170139487)

[4.3. Staff Nurse I/O 41](#_Toc170139488)

[4.4. Hospital Admin I/O 43](#_Toc170139489)

[5. Conclusion 47](#_Toc170139490)

[6. References 48](#_Toc170139491)

[7. Workload Matrix 49](#_Toc170139492)

# Introduction and Assumptions

This documentation is about the implementation of our Hospital Management System, written in C. The system is designed into 4 main components, which handles patients’ records, doctor details, staff nurse information, and hospital administration functionalities. The main goal of this system is to provide an efficient and seamless process for managing hospital operations conveniently.

## Objectives

1. **Streamlined Patient Record Management**

* Efficiently and securely store retrieve patient information including health records, medical history, treatments, etc.
* Allow patients to easily review their EHR, curate appointments and payments to their requirements.

1. **Comprehensive Doctor Details Management**

* Maintain a database to manage Hospital Doctor profiles, including schedules, specializations, etc.
* Facilitate the allocation of patients to doctors based on availability and specialization, optimizing resource utilization.

1. **Staff Nurse Information Handling**

* Keep track of staff nurses, their assigned duties, schedules, etc.
* Enables efficient communication between nurses and other healthcare providers, ensuring coordinated patient care.

1. **Robust Hospital Administration Functionalities**

* Support administrative tasks such as user management, role assignments, and system maintenance
* Provide tools for hospital administrators to manage the overall operations, ensuring compliance with hospital policies and procedures.

## System Overview

The Hospital Management System is structured to provide a user-friendly experience while maintaining a high level of functionality. Every module is designed to address the specific need of its user group, ensuring that the system satisfies every user’s needs, while being integrated and seamlessly sewn together, providing a smooth operation and data handling experiences, despite it being a terminal-based program.

* **Patient Module:** Allows patient view, update, and remove appointments, view their HER, and also manage their payments.
* **Doctor Module:** Allows doctors to schedule for availability, view HER records, view and search for reports and analytics
* **Staff Nurse Module:** Allows staff nurse to access doctor schedules, manage inventory, viewing and searching for reports and analytics
* **Hospital Administrator:** Allows admins to register new users, view appointment trends, view and search inventory.

## Assumptions

Users have a basic understanding of terminal commands

It is assumed that all users interacting with the system are familiar with basic terminal commands and navigation. This knowledge is necessary to operate the command-line interface effectively and execute the program.

The hospital management system will be run on a system with a C compiler:

The system is designed to be compiled and run on any computer that has a C compiler installed, such as GCC (GNU Compiler Collection). This ensures that the program can be compiled and executed without issues related to compatibility.

All users (patients, doctors, staff nurses, admins) will interact with the system through a command-line interface:

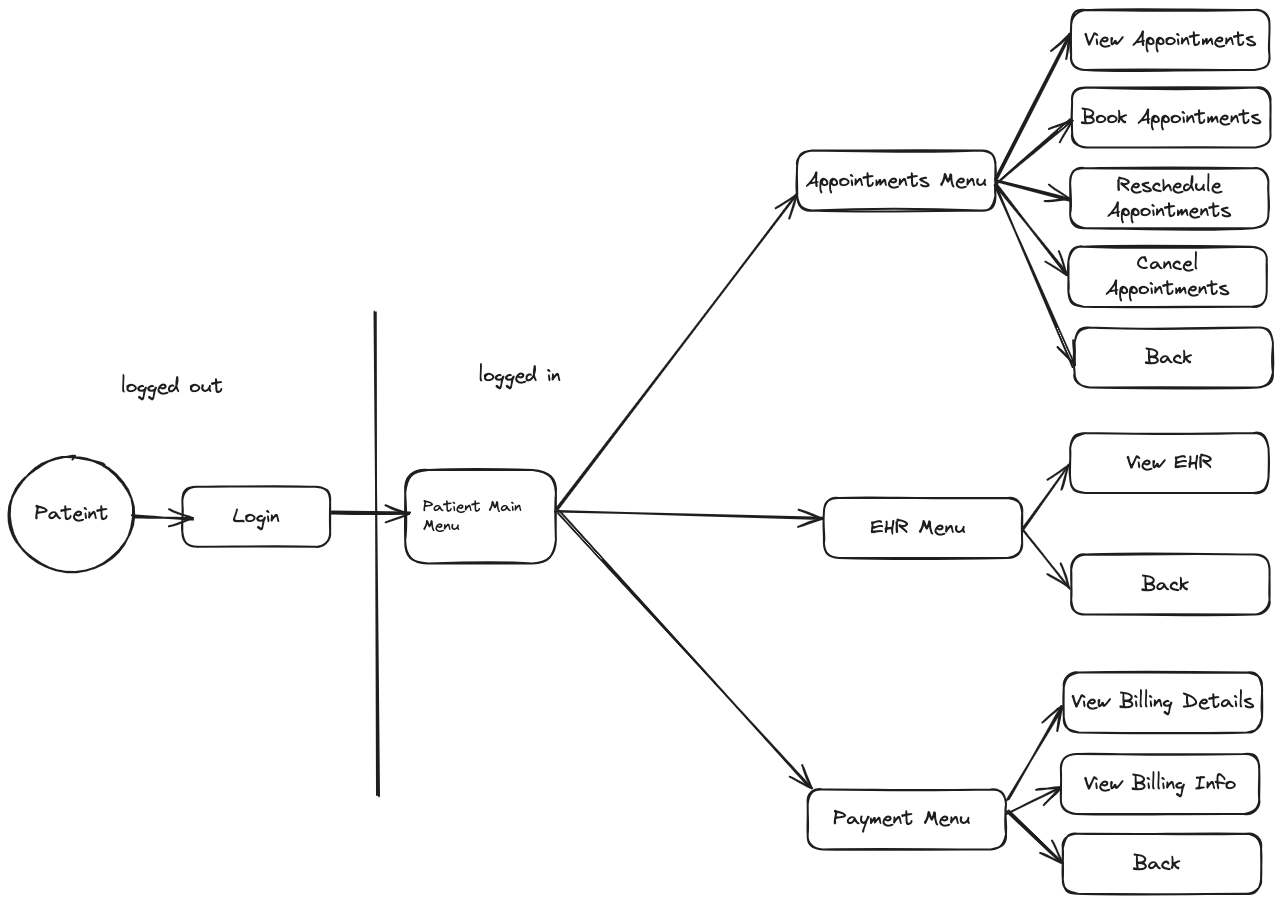
The interface for this system is entirely text-based, requiring users to input commands and data via a terminal. This simplifies the user interface and focuses on core functionality without the need for graphical components.

The database is a text file storing user and patient data:

The system uses a simple text file to store all relevant data for users and patients. This approach allows for easy read and write operations and simplifies the data management process. However, it is assumed that users understand the limitations and potential security issues associated with using plain text files for data storage.

# Program Design

## Patient Module Design

****

*(Diagram 2.1: Patient Module Design State Flow Diagram)*

This state flow diagram shows the main structure (backbone) of the Patient Module. Below is the breakdown of the structure:

**1. Patient Status:**

**Logged out:** The patient is not currently signed in to the portal.

**Logged in:** The patient has successfully signed into the portal.

**2. Main Menu:**

After logging in, the patient can access the main menu, which serves as the primary navigation point for various features of the portal.

**3. Appointments Menu:**

This section allows the patient to manage their appointments.

* **Book Appointments:** Schedule new appointments.
* **View Appointments:** See a list of upcoming and past appointments.
* **Reschedule Appointments:** Change the date and time of an existing appointment.
* **Cancel Appointments:** Cancel an existing appointment.
* **Back:** Return to the Main Menu.

**4. EHR (Electronic Health Record) Menu:**

This section allows the patient to access their health records.

* **View EHR:** View their personal health records and medical history.
* **Back:** Return to the Main Menu.

**5. Payment Menu:**

This section allows the patient to manage billing and payment information.

* **View Billing Details:** See details of the bills, including itemized charges.
* **View Billing Info:** Access overall billing information and payment history.
* **Back:** Return to the Main Menu.

**6. Login:**

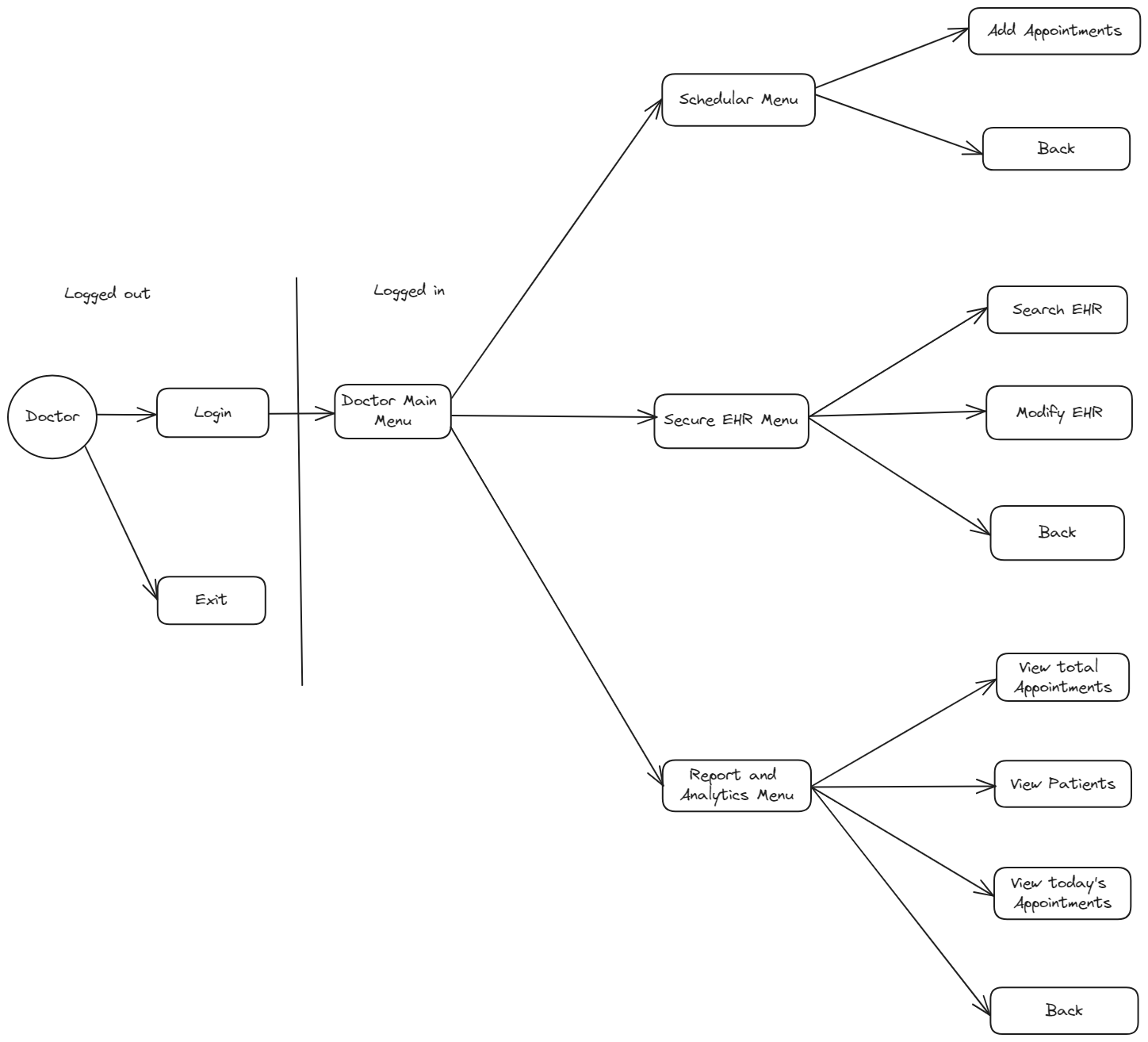
This is the option for the patient to log into the portal, usually found on the initial screen or when the patient is logged out.

**Navigation Flow**

1. **Patient logs in** to the portal.
2. **Main Menu** is displayed with options to access different menus.
3. **Appointments Menu:** The patient can manage their appointments.
4. **EHR Menu:** The patient can view their electronic health records.
5. **Payment Menu:** The patient can view billing details and information.
6. **Back:** Navigation buttons allow the patient to return to the previous menu or the main menu.

This structure provides a user-friendly way for patients to navigate through the portal, ensuring they can easily access and manage their health-related information and appointments.

## Doctor Module Design



The diagram depicts the flow of the Doctor interface within the Hospital Management System (HMS), with each box symbolizing a menu or interface state for a doctor. This design intends to offer an organized and intuitive structure for the doctor's functionalities, ensuring an efficient workflow between the states.

Initially, the doctor encounters options to "Login" or "Exit" the system. After selecting "Login" and successfully entering credentials, the doctor is directed to the "Doctor Main Menu," which serves as the central hub for accessing various functionalities.

From the "Doctor Main Menu," the doctor can proceed to three primary menus: the "Scheduler Menu," the "Secure EHR Menu," and the "Report and Analytics Menu."

1. **Scheduler Menu:** This section focuses on managing appointments. Doctors can "Add Appointments" and navigate "Back" to the main menu, ensuring smooth and efficient scheduling.
2. **Secure EHR Menu:** This section is dedicated to managing Electronic Health Records (EHR). Doctors can "Search EHR" to retrieve patient records, "Modify EHR" to update information, and use the "Back" option to return to the main menu seamlessly. This feature ensures quick and secure access to vital patient data.
3. **Report and Analytics Menu:** This section offers tools for generating reports and analyzing data. Doctors can "View total Appointments" for an overview, "View Patients" to access patient lists, and "View today's Appointments" to focus on the current day's schedule. A "Back" option is also provided to return to the main menu.

Several considerations are factored into this design: usability, security, responsiveness, error handling, and scalability. The interface is user-friendly and intuitive, allowing efficient navigation and task performance. Security is ensured through secure login and access control to protect sensitive medical data. The interface is responsive to different devices and screen sizes, ensuring accessibility for all staff. Robust error handling manages invalid inputs or system errors gracefully. Finally, the design accommodates future enhancements and additional functionalities without major redesigns.

In summary, this program design aims to provide a structured and organized interface for doctors within the HMS. By clearly defining each menu and interface state, it ensures an efficient and user-friendly experience, allowing doctors to perform their tasks effectively while maintaining the integrity and security of the system.

## Staff Nurse Module Design

A diagram of a patient

Description automatically generated

The diagram illustrates the flow of the Staff Nurse interface within the Hospital Management System (HMS), with each box representing a menu or interface state of function of a staff nurse. This design aims to provide an organized and intuitive structure for the nurse's functionalities, ensuring an efficient workflow between the states.

Starting at the Main Menu (HMS), the nurse logs in through the Nurse Login Page, which then verifies the nurse's credentials before granting access to the Main Menu (Nurse). From this main menu, the nurse can access three primary functionalities: Access Doctor Schedule, Manage Inventory, and Generate Reports.

The "Access Doctor Schedule" functionality allows nurses to view and manage doctor schedules. Within this section, they can view current schedules, add new appointments, remove existing appointments, and modify appointments.

The "Manage Inventory" functionality is crucial for maintaining medical supplies. Here, nurses can view the list of current inventory items, search for specific items, identify items that are running low, add new items to the inventory, remove items, and edit details of existing inventory items.

Lastly, the "Generate Reports" functionality enables nurses to generate various reports for administrative and medical purposes. They can generate patient lists, lists of patients under specific doctors, and patient appointment details.

Several factors are considered in this design: usability, security, responsiveness, error handling, and scalability. The interface is designed to be user-friendly and intuitive, allowing nurses to navigate through menus and perform tasks efficiently. Security is ensured through secure login and access control to protect sensitive medical data. The interface is responsive to different devices and screen sizes, ensuring accessibility for all staff. Robust error handling manages invalid inputs or system errors gracefully. Finally, the design accommodates future enhancements and additional functionalities without major redesigns.

In conclusion, this program design aims to provide a structured and organized interface for staff nurses within the HMS. By clearly defining each menu and interface state, it ensures an efficient and user-friendly experience, allowing nurses to perform their tasks effectively while maintaining the integrity and security of the system.

## Hospital Administrator Module Design

# Implementation

## Patient Implementation

To implement the design of the Patient Module, the approach taken was to implement each feature individually, and use the standardized structs in order to store and access information with the database. After that, the main menu was designed referring g to the design above, where it was split into 4, which are the appointment management section, the EHR Viewing section, the payment management section, and the logout function. The main menu was linked to these 4 functions, with the Appointments menu giving the patient the option to book, reschedule, view, and cancel the appointments according to their needs, as well as go back to the main menu. As for the EHR Menu, its function lets the patient view their HER Record from the database, and go back to the main menu. The payment menu has the function that allows the patient to display their bill and also go back to the main menu. Last but not least, the logging out function allows patients to log out, and takes them back to terminal.

*// Function to provide access to patient features*

void *patientAccess*();

*// Function to authenticate patient login with username and password*

bool *patientLogin*(const char\* username, const char\* password);

*// Function to display a list of upcoming appointments for the patient*

void *patientViewAppointments*();

*// Function to reschedule an existing appointment*

void *patientRescheduleAppointment*();

*// Function to cancel an existing appointment*

void *patientCancelAppointment*();

*// Function to book a new appointment*

void *patientBookAppointment*();

*// Function to display the patient's Electronic Health Record (EHR)*

void *patientDisplayEHR*();

*// Function to display the patient's billing history*

void *patientViewBillingHistory*();

*// Duplicate function declaration (should be removed)*

void *patientMainMenu*();

*// Function to display the appointments menu*

void *displayAppointmentsMenu*();

*// Function to display the EHR menu*

void *displayEHRMenu*();

*// Function to display the payment menu*

void *displayPaymentMenu*();

*(The code above are the list of functions used to create the Patient Module)*

**List of Functions and their Explanations**

* **patientAccess();**

This function is the entry point for the patient access system. It prompts the patient to enter their username and password, and it calls the ‘patientLogin’ function to authenticate the credentials entered by the patient, and if the log in is successful, it displays the patient’s main menu, while if the login was unsuccessful, it displays an error message.

* **patientLogin(const char \* username, const char\* password);**

This function is made to authenticate the patient’s login credentials. It takes two parameters, which are ‘username’ and ‘password’, which are the credentials required in order to be entered by the patient. The function retrieves a list from the database using the ‘getAllPatients’ function, iterates through the list and checks if the ‘username’ and ‘password’ input match the credentials stored in the database for each patient. If a match is found, it sets the ‘loggedInPatient’ variable to the authenticated patient’s details, including their patientID, name, etc. The function returns a Boolean value indicating whether the login was successful or not. If the login is successful, the function returns ‘true’; otherwise, it returns ‘false’.

* **patientMainMenu();**

This function displays the pateint’s main menu, which allows them to navigate to different sections of the system. The functionalities are:

* Display a menu with options to view appointments, view EHR, manage payment, and logout
* Prompts on the patient’s choice, it calls the corresponding function (e.g. ‘displayAppointmentsMenu’, ‘displayEHRMenu’, etc.)
* **patientViewAppointments();**

This function displays a list of upcoming appointments for the logged-in patient. It

* retrieves a list of appointments from the database using the ‘getAllAppointments’ function.
* iterates through the list of appointments and checks if each appointment belongs to the logged-in patient by comparing the appointment's patient ID with the ‘loggedInPatient’ ID.
* If an appointment belongs to the patient, it prints out the appointment details, including the appointment ID, doctor ID, date, and time.
* If no appointments are found, it displays a message indicating that no appointments are scheduled.
* **patientRescheduleAppointment();**

This function allows the patient to reschedule an existing appointment. Here's how it works:

* it prompts patient to insert appointment ID they want to reschedule
* retrieves the appointment details from the database using the ‘getAppointment’ function
* Checks if the appointment belongs to the logged-in patient, and if not, it displays an error message.
* If appointment is valid, it prompts the patient to enter a new date and time for the appointment
* It updates the appointment details in the database using the ‘removeAppointment’ and ‘addAppointment’ functions.
* **patientCancelAppointment();**

This function allows the patient to cancel an existing appointment. It:

* Prompts patients to enter the appointment ID they want to cancel
* Retrieves the appointment details from the database using the ‘getAppointment’ function.
* Checks if the appointment belongs to the logged-in patient and if it exists in the database. If not, it displays an error message.
* If the appointment is valid, it removes the appointment from the database using the ‘removeAppointment’ function.
* **patientBookAppointment();**

This function allows the patient to book a new appointment, it works like this:

* It prompts the patient to enter the doctor ID, date, time, for the new appointment
* It creates a new appointment object with the entered details
* It adds the new appointments to the database using the ‘addAppointment’ function
* Displays a success message if the appointment is booked successfully
* **patientDisplayEHR();**

This function displays the patient’s Electronic Health Record (HER), heres what it does:

* It retrieves the patient’s EHR from the database using the ‘getALLEHRs’ function
* It iterates through the EHR data and formats it for display
* It prints out the EHR data, including the patient’s medical history, prescriptions, allergies, and billing information.
* **patientViewBillingHistory();**

This function displays the patient’s billing history. Here is what if does:

* Retrieves a list of bills from the database using the ‘getAllBills’ function
* Iterates through the list of bills and checks if each bill belongs to the logged-in patient by comparing the bill’s patient ID with the ‘loggedInPatient’’s ID
* If a bill belongs to the patient, it prints out the bill details, including the bill ID, patient ID, invoice date, service provider, and amount due.
* **displayAppointmentsMenu();**

This function displays the appointments menu, which allows the patient to view, book, reschedule, or cancel appointments.

* It displays a menu with options to view appointments, book an appointment, reschedule an appointment, and cancel an appointment
* It prompts patient to enter their choice
* Based on the patient’s choice, it calls the corresponding function (e.g. ‘patientViewAppointments’)
* **displayEHRMenu();**

This function displays the appointments menu, which allows the patient to view HER.

* It displays a menu with options to view EHR
* It prompts patient to enter their choice
* Based on the patient’s choice, it calls the corresponding function (e.g. ‘patientDisplayEHR’)
* **displayPaymentMenu();**

This function displays the appointments menu, which allows the patient to view their bill history.

* It displays a menu with options to view billing history
* It prompts patient to enter their choice
* Based on the patient’s choice, it calls the corresponding function (e.g. ‘patientViewBillingHistory’)

## Doctor Implementation

**1. Overall Structure**

The program is designed to manage users and appointments for a medical system. It includes functionalities for validating user credentials, adding appointments, checking appointment availability, saving and loading data from files, and displaying menus for user interaction.

**2. Constants and Data Structures**

**Constants:**

* **‘max\_users’**, **‘max\_username\_length’**, **‘max\_password\_length’**: Limits for user accounts and their credentials.
* **‘max\_appointment’**, **‘max\_filename\_length’**: Limits for appointment records and filename lengths.

**Structures:**

* **struct User**: Represents a user with a **‘username’** and **‘password’**.
* **struct Appointment**: Represents an appointment with a **‘date’**, **‘time’,** and **‘doctorName’**.

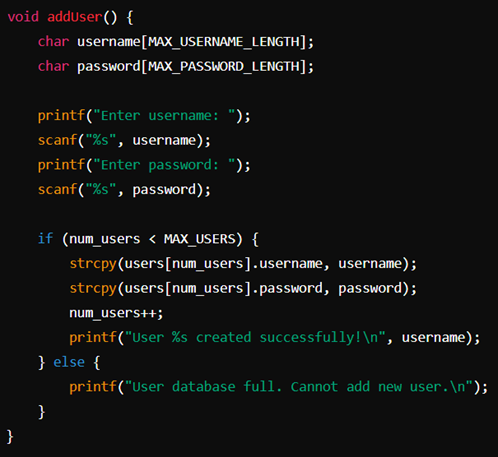
**Global Variables:**

* **users**: Array of **‘User’** structures to store user accounts.
* **num\_users**: Counter for the current number of users.
* **Appointments**: Array of **‘Appointment’** structures to store appointment records.
* **num\_appointments**: Counter for the current number of appointments.

**{ Functionality }**

**1. User Management**

**Adding Users**



Prompts the user to enter a username and password.

Adds the new user to the **‘users’** array if there is space.

* Increments **‘num\_users’**.
* **Validating User Credentials**

· Checks if the provided username and password match any existing user.

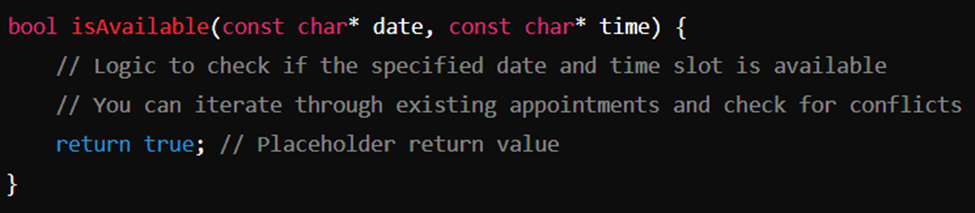
· Returns 1 for a valid user, 0 for an invalid user.

**2. Appointment Management**

* **Adding Appointments**



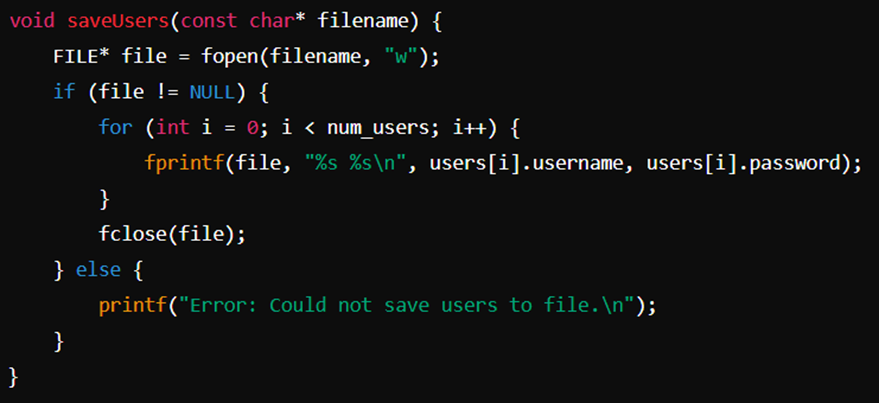
* Prompts the user to enter the date, time, and doctor's name for an appointment.
* Adds the appointment to the **‘appointments’** array if there is space.
* Increments **‘num\_appointments’**.
* **Checking Availability**



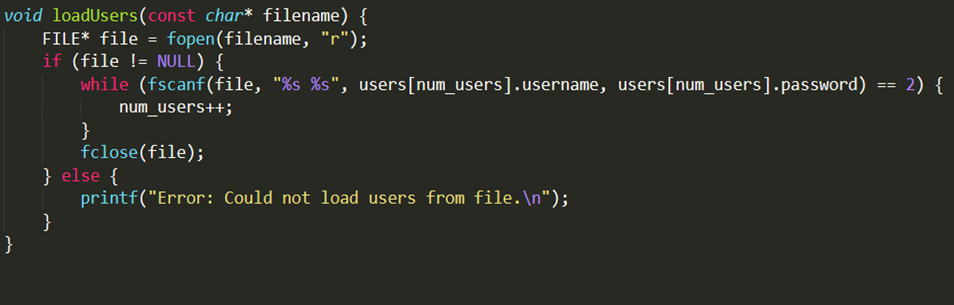
* Checks if the specified date and time slot is available.
* Currently a placeholder, returns **‘true’** by default.

**3. File Operations**

* **Saving Users to a File**

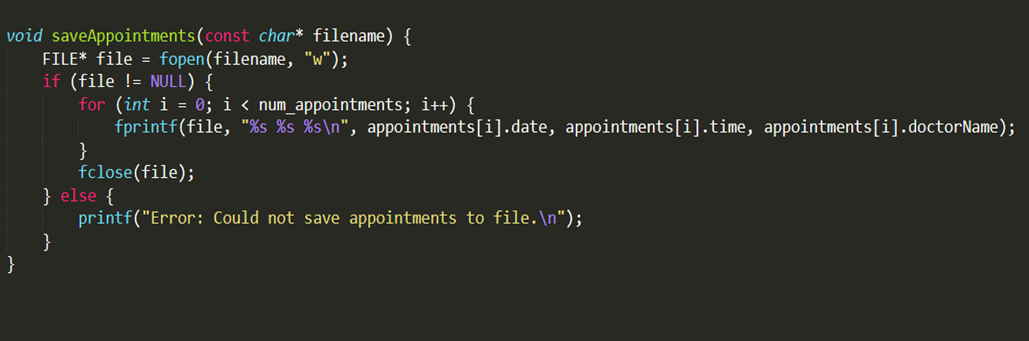


* Saves the user data to a specified file.
* Writes each user's username and password to the file.
* **Loading Users from a File**



* Loads user data from a specified file.
* Reads each user's username and password from the file and populates the **‘users’** array.

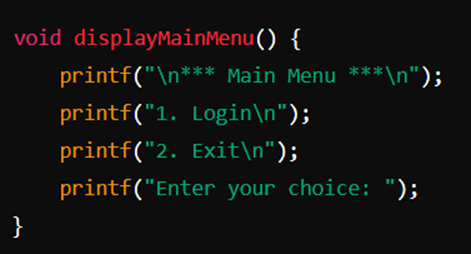
* **Saving Appointments to a File**



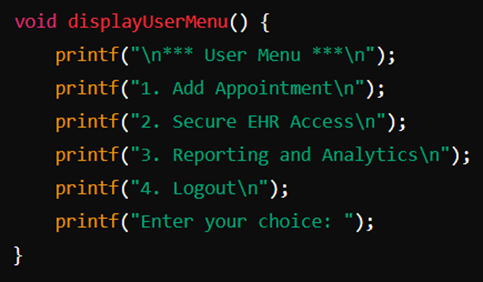
* Saves appointment data to a specified file.
* Writes each appointment's date, time, and doctor's name to the file.

**4. Menu Display Functions**

* + **Displaying Main Menu**



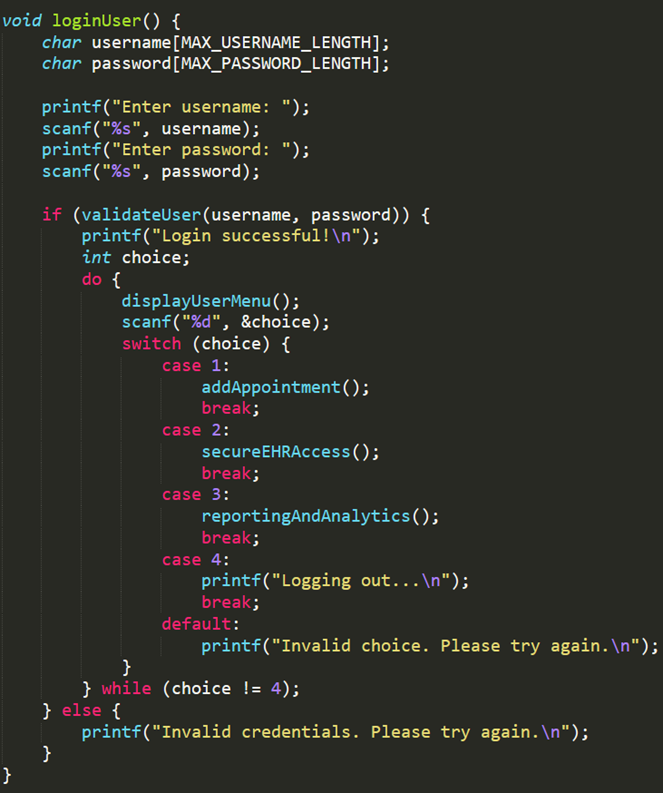
* + Displays the main menu options to the user.
* **Displaying User Menu**



* Displays the user menu options after successful login.

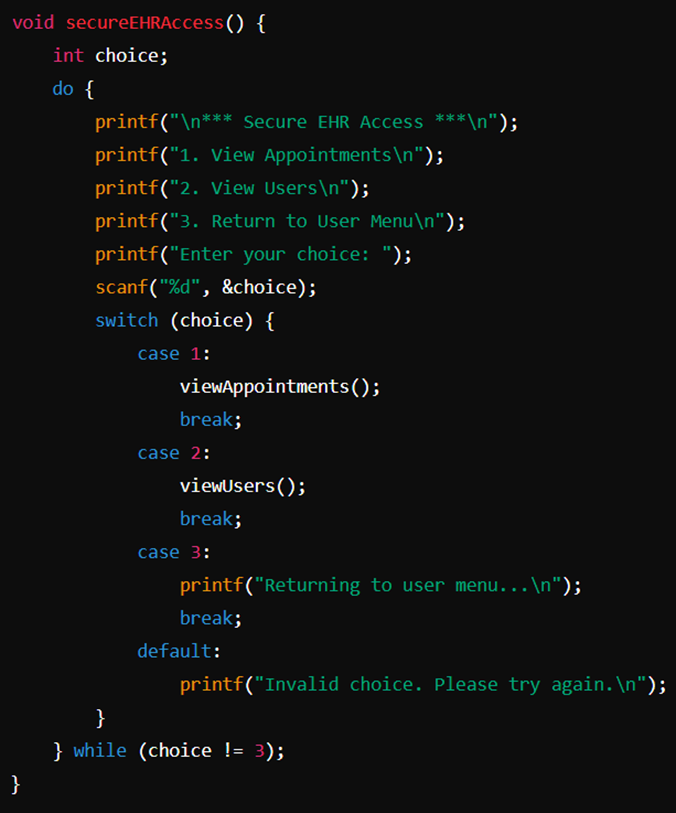
**5. User Interaction Functions**

**User Login**

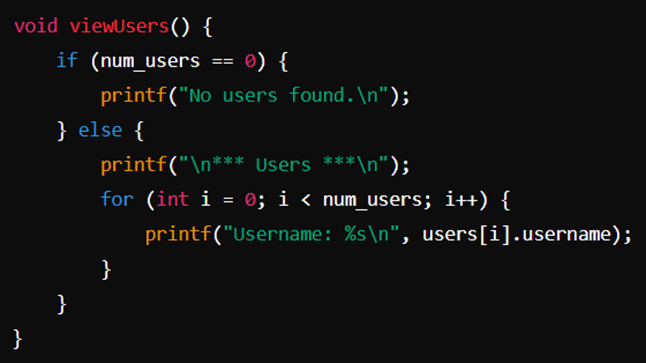


* Prompts for username and password.
* Validates credentials and displays the user menu if successful.
* Allows the user to add appointments, access secure EHR, and perform reporting and analytics.

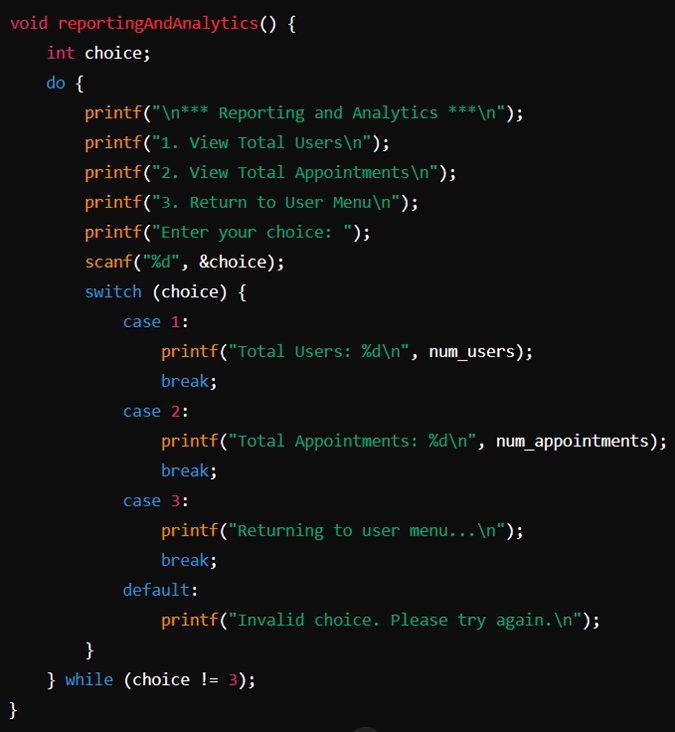
* **Secure EHR Access**



* Displays options for secure EHR access.
* Allows the user to view appointments and users.
* **Viewing Users**



* Displays all stored users.
* **Reporting and Analytics**



* Displays options for reporting and analytics.
* Shows the total number of users and appointments.

## Prompts the user to enter a username and password.

## Staff Nurse Implementation

### Header File Explanation: staffnurse.h

#pragma once

#ifndef STAFFNURSE\_H

#define STAFFNURSE\_H

These lines ensure that the header file is included only once during the compilation process to prevent duplicate definitions.

#include <stdio.h>

#include "include\cmdui\_utils.h"

#include "include\doctordb.h"

#include "include\staffnursedb.h"

#include "include\appointmentdb.h"

#include "include\patientdb.h"

#include "include\datautils.h"

#include "include\inventorydb.h"

These #include directives include the necessary header files required for the Staff Nurse functionalities. Each of these headers contains definitions and declarations related to different parts of the system:

* stdio.h : Standard I/O functions.
* cmdui\_utils.h: Utility functions for command-line UI.
* doctordb.h: Functions related to doctor database operations.
* staffnursedb.h: Functions related to staff nurse database operations.
* appointmentdb.h: Functions related to appointment database operations.
* patientdb.h: Functions related to patient database operations.
* datautils.h: Utility functions for data manipulation.
* inventorydb.h: Functions related to inventory database operations.

*void* staffNurseLogin();

*void* staffNurseMainMenu();

*void* staffNurseAccessDoctorSchedules();

*void* staffNurseManageInventory();

*void* staffNurseGenerateReports();

*void* staffNurseLoggedOutMenu();

These lines declare the functions that will be implemented in the source file:

* staffNurseLogin(): Handles the login process for staff nurses.
* staffNurseMainMenu: Displays the main menu for staff nurses.
* staffNurseAccessDoctorSchedules(): Manages doctor appointments and availability.
* staffNurseManagementInventory(): Manages inventory items.
* staffNurseGenerateReports(): Generates reports and analytics.
* staffNurseLoggedOutMenu(): handles the menu display after logout.

#endif // STAFFNURSE\_H

This line marks the end of the include guard.

### Main file of StaffNurse Explanation: staffNurse.c

#include "staffNurse.h"

This line is for including the previous header file for StaffNurse main C file.

StaffNurse loggedInNurse;

A global variable loggedInNurse is declared to store the currently logged-in staff nurse's information.

//Define login function with staff nurse username and password which will also check the user validity

*void* staffNurseLogin() {

    clearScreen();

*char* username[50], password[50];

    printf("Enter username: ");

    scanf("%s", username);

    printf("Enter password: ");

    scanf("%s", password);

*int* nNurses;

    StaffNurse \*nurses = getAllStaffNurses(&nNurses);

    for(*int* i=0; i<nNurses; i++) {

        if ((strcmp(nurses[i].name, username) == 0)

         && (strcmp(nurses[i].password, password) == 0)) {

            printf("Login successful!\n");

            getchar();

            staffNurseMainMenu();

            return;

        }

    }

    printf("Invalid username or password. please try again.\n");

    getchar();

    free(nurses);

}

This function handles the login process for staff nurses. It prompts the user to enter their username and password, retrieves the list of staff nurses, and checks the entered credentials against the stored ones. If the credentials match, it calls the staffNurseMainMenu() function. If not, it will display an error message.

//Define nurseMenu Function which will show the options that Staff Nurse can do

*void* staffNurseMainMenu() {

    clearScreen();

The function displays the main menu for the staff nurse, providing options to manage doctor appointments, inventory, generate reports, or logout. The user’s choice determines which function to call next.

*void* staffNurseAccessDoctorSchedules() {

    clearScreen();

This function provides options for managing doctor appointments, including viewing, adding, removing, and rescheduling appointments. It uses a loop to keep presenting the menu until the nurse chooses to exit.

*void* staffNurseManageInventory() {

    clearScreen();

This function handles the inventory management functionalities for the staff nurse. It provides options to view inventory, search items, view low stock items, add new items, remove items, and modify existing items. The loop continues to display the menu until the nurse chooses to exit.

*void* staffNurseGenerateReports() {

    clearScreen();

This function handles the Report Generation functionalities for the staff nurse. It provides options to view all the patient’s data, patients with the doctors, and patients’ appointments. The loop will continue to display the menu for generating reports until the nurse chooses to exit.

*void* staffNurseLoggedOutMenu() {

    clearScreen();

This function displays the initial menu for the staff nurse, providing options to Login as Staff Nurse, and Exit the HMS. The loop continues to display the menu until the nurse chooses to exit.

## Hospital Administrator Implementation

**Menu Printing functions**

This group of functions has as its sole purpose displayingspecific menu options. Under the Hospital Administrator module, all of these make use of the Menu struct and its utility functions defined under the cmd\_utils module. All of these functions are called at some point by a UI State function.

void hospAdminDisplayHospitalAdminMenu();

void hospAdminDisplayUserMngmntMenu();

void hospAdminDisplayPatientMngmntMenu();

void hospAdminDisplayDoctorMngmtMenu();

void hospAdminDisplayStaffNurseMngmtMenu();

void displayAppointmentMngmntMenu();

void displayEHRMngmntMenu ();

void displayEHRmodifyMenu();

void displayBillMngmntMenu ();

void hospAdminDisplayInventoryMngmntMenu();

**printDefaultSeparator()**

This miscellaneous function is used as an UI utility specifically for the Hospital Administrator module. It prints a horinzontal bar or *separator* making use of a character from the extended ASCII set.

void printDefaultSeparator()

**UI State Functions**

This group of functions are the backbone of the module, each of them working as a node with an specific purpose within the Hospital Administrator interface.

Among them, we find the following:

void hospitalAdminLogin(SystemState \*sysState);

**hospitalAdminLogin():** Once hospital admin access is chosen from the main HMS menu,

void hospitalAdminMenu(SystemState \*sysState);

**hospitalAdminMenu():** This is the first menu with which a hospital administrator interacts with once logged in.

void hospAdminPatientManagementMenu(SystemState \*sysState);

**hospAdminPatientManagementMenu():** Displays the main menu for patient management functionalities available to the hospital administrator.

void hospAdminSeeAllPatients(SystemState \*sysState);

**HospAdminSeeAllPatients**()**:** Provides a list of all patients currently registered in the system.

void hospAdminSearchPatient(SystemState \*sysState);

**hospAdminSearchPatient():** Allows the administrator to search for a specific patient using various search criteria.

void hospAdminModifyPatient(SystemState \*sysState);

**hospAdminModifyPatient():** Enables the administrator to modify the details of an existing patient.

void hospAdminRegisterPatient(SystemState \*sysState);

**hospAdminRegisterPatient():** Facilitates the registration of a new patient into the system.

void hospAdminUnregisterPatient(SystemState \*sysState);

**hospAdminUnregisterPatient():** Allows the administrator to remove a patient from the system.

void hospAdminDoctorManagementMenu(SystemState \*sysState);

**hospAdminDoctorManagementMenu():** Displays the main menu for doctor management functionalities available to the hospital administrator.

void hospAdminSeeAllDoctors(SystemState \*sysState);

**hospAdminSeeAllDoctors():** Provides a list of all doctors currently registered in the system.

## Database Access Implementation

**Metadata Management Code**

*deserializeMetadata*(DBMetadata \*targetMetadata, const char \*metadataFilePath);

*serializeMetadata*(DBMetadata \*targetMetadata, int nInvTypes, const char \*metadataFilePath);

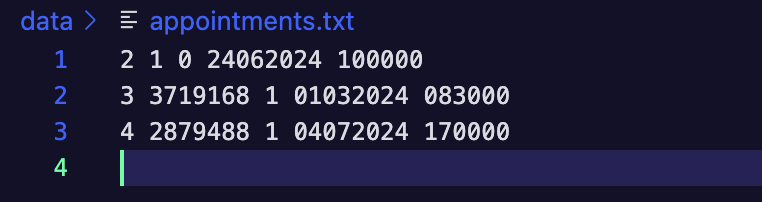
* **deserializeMetadata(DBMetadata \*targetMetadata, const char \*metadataFilePath);**

This function reads metadata from a file and loads it into a ‘DBMetadata’ struct, allowing the program to access and use the metadata.

* **serializeMetadata(DBMetadata \*targetMetadata, int nInvTypes, const char \*metadataFilePath);**

This function takes the metadata from a ‘DBMetadata’ struct and writes it to a file, saving the metadata for later use.

* **Appointment Database**



* **Doctor Database**

**A blue and white text on a black background

Description automatically generated**

* **EHR Database**

A computer screen with white text

Description automatically generated

* **Inventory Database**

**A computer screen shot of a program

Description automatically generated**

* **Patient database**

**A screenshot of a computer program

Description automatically generated**

* **Staff Nurse Database**

A blue background with white text

Description automatically generated

**CMDUI\_UTILS (User Interface) Implementation**

*setCMDcolor*(ANSIcolor clr);

*printGreen*(char \*txt);

*printRed*(char \*txt);

*printYellow*(char \*txt);

*displayErrorMsg*(char \*errorMsg);

*displaySuccessMsg*(char \*successMsg);

*resetCMDcolor*();

*clearScreen*();

*waitForInput*();

*getSafeIntInput*(int \*input);

*printTextBox*(char target[]);

*printMenu*(Menu \*menuptr);

**Color functions**

The code defines several functions for setting the text color using ANSI escape codes:

* **setCMDcolor(ANSIcolor clr);**

Sets the text color to the specified color (red, green, or yellow).

* **printGreen(char \*txt), printRed(char \*txt), and printYellow(char \*txt);**

Print a line of text in the specified color with a newline character.

**displayErrorMsg(char \*errorMsg) and displaySuccessMsg(char \*successMsg);** Print error or success messages with a prefix and in the specified color.

* **resetCMDcolor():**

Resets the text color to its original state.

**Screen clearing and waiting functions**

**The code defines two functions:**

* **clearScreen();**

Clears the terminal screen using the cls command on Windows and clear command on Linux.

* **waitForInput();**

Waits for the user to press a key, halting execution.

**Menu rendering functions**

The code defines two functions for rendering menus:

* **printTextBox(char target[]);**

Prints a text box with a title and optional padding.

* **printMenu(Menu \*menuptr);**

Prints a menu with a title, options, and padding.

**Other functions**

The code defines a few other functions:

* **getSafeIntInput(int \*input);**

Safely reads an integer input from the user using scanf and atoi.

* **saveScreen(TerminalInfo\* termInfo) and restoreScreen(TerminalInfo\* termInfo);**

These functions are commented out and appear to be incomplete or unused.

# Sample Input/Output

**A screen shot of a computer

Description automatically generated**

This is the Hospital Management System Main Menu; it is the first thing that is displayed. There are multiple login options for the user to choose from, which are patient, doctor, staff nurse, and hospital admin access. The last option, ‘exit’, allows the user to return to the terminal and exit the executed program.

## Patient I/O

A screenshot of a computer program

Description automatically generated

This is the Patient login page, when “patient access’ is selected from the main menu, it will forward you to this page. Here, the user is required to log in, in order to verify the patient credentials and permit access to access the patient information, based on the user. The photo above is the user, logging in.

A screen shot of a computer screen

Description automatically generated

Here is the patient module’s main menu. Here, the options given for the patient is to manage appointments, access EHR Records, Manage Payment, and log out. This page requires the user’s input to reforward them into the allocated menu (eg. Choice 1 will transport the user into the Appointment management menu.

A black background with white text

Description automatically generated

Above, is the Appointments menu. In this page, users get to view, book, reschedule, or cancel appointments based on their liking by entering the number required. They also can go back to the main menu by picking that option.

A screenshot of a computer

Description automatically generated

Here, this is the Cancelling appointment feature, where the user chooses to cancel appointment number 3.

A screenshot of a computer

Description automatically generated

Here is the Book Appointment menu, where the patient gets to set a date and time, as well as choose the doctor they want to book an appointment with.

## Doctor I/O

A screenshot of a computer screen

Description automatically generated

This is the Doctor Menu, when the user chooses the Doctor option. This display is the login portal, which takes the login information from the user.

A close-up of white text

Description automatically generated

This display is the Secure EHR access menu, where the doctor is available to access and modify the EHR records of the patient. This page takes the user input in order to transport to the next screen.

A black background with white text

Description automatically generated

This page is the EHR Page, where it shows the medical history, prescriptions, and allergies.

A screenshot of a computer

Description automatically generated

In this page, the doctor gets to view their schedule, which also accesses the database to obtain the information for the appointments that have been booked by the patients to see the doctor, as well as the details of the patient.

## Staff Nurse I/O

A screen shot of a computer

Description automatically generated

This is the login page of Staff Nurse. The nurse user can make an option either login in or exit the system.

A screenshot of a computer screen

Description automatically generated

If the user enters 1, then it will go to this current page and the user needs to enter the username and password of Staff Nurse. If validation is successful, it will be directed to the next page. If not, it will show the error message.

A screenshot of a computer

Description automatically generated

Here is the four options that the staff nurse can make; Doctor Appointments, Inventory management, Generating Reports & Analytics, and Logout.

A screen shot of a black screen

Description automatically generated

This page will be shown if the user chooses option 1 which is ‘Doctor Appointments’. There are four options here: view doctors’ appointments, add new appointments, remove appointments, reschedule appointments, and exit. If the user enters option 5, it will ask the user if he/she wants to go back to the previous menu or not. (picture below)

A black background with white text

Description automatically generated

This page is a terminal menu which based on the input of the user, gives the option to either go back to the previous menu, or to exit the program.

A screenshot of a computer

Description automatically generated

In this page, it displays the Inventory management page, with options suych as viewing, searching, adding, and modifying inventory items, as well as exiting to the main page.

A screenshot of a computer program

Description automatically generated

This page is showing the list of inventory in our hospital with the description, category, and quantity. After that it will ask you to go back to the previous menu or exit the system.

A screenshot of a computer

Description automatically generated

This page is for report generation for staff nurse by being able to view all the patient data, patients with responsible doctors, and patients’ appointments. As usual, it will ask you to go back to the previous menu or if the user wants to exit the system.

## Hospital Admin I/O

**A screenshot of a computer screen

Description automatically generated**

This is the Hospital Administrator login portal, which is where the users who pick to log in as Hospital Administartors choose to log in.

**A black sign with white text

Description automatically generated**

This is the Hospital Admin main menu, when the user gets logged in and access gets granted.

**A black sign with white text

Description automatically generated**

In this page, the admin gets to manage users profiles, which are the pateints, doctors, and staff nurse.

**A black and white sign with white text

Description automatically generated**

In this page, the admin gets to manage patients by seeing, searching, modifying, registering, and unregistering patients.

**A screenshot of a computer screen

Description automatically generated**

In this page, the admin gets to manage doctors by seeing, searching, modifying, registering, and unregistering doctors.

**In this page, the admin gets to manage patients by seeing, searching, modifying, registering, and unregistering patients.**

**A screenshot of a computer program

Description automatically generated**

**A sign with white text

Description automatically generated**

This page is viewing the staff nurse management from the Admin side. There are altogether 6 options; To see all staff nurses in the hospital, To search nurse, To modify staff nurse’s profile, To register new staff nurse, To Unregister Staff nurse, and Go back.

**A computer screen shot of a computer program

Description automatically generated**

In this section, the Hospital Admin gets to view the patient’s profile, and load their information.

A computer screen shot of a computer code

Description automatically generated

In this page, it searches and displays details for patient ID 1, showing ID, IC/passport, name, password, and isLocal status.

A screenshot of a computer program

Description automatically generated

In this page, it loads and displays inventory details for 4 items, including ID, name, description, category, and quantity.

# Conclusion

As a conclusion, this Hospital Management System in C provides a basic yet visually appealing and functional framework for managing hospital operations. By dividing functionalities among distinct modules for patients, doctors, staff nurses, and administrators, the system ensures a seamless and concise approach to handling healthcare data.

Some planned features to be implemented in order to improve this service in the future are:

**Sophisticated Database System**

We plan on transitioning from a text-file based system into a more sophisticated database which uses relational database management system (RDBMS) like MySQL to improve data integrity, security, and performance.

**Graphical User Interface (GUI)**

Developing a full-scale GUI to provide visual elements with buttons, forms, etc which can enhance usability, especially for users who are not comfortable with command-line interfaces or using terminals.

**Enhanced Security Features**

We plan to develop and implement an advanced user authentication and authorization mechanism to ensure that only authorized personnel have access to sensitive data. We also want to implement a feature that encrypts the stored data to protect patient information from unauthorized access and breaches.

To conclude this documentation, this project is a valuable tool for learning and applying basic programming, database management, and system design concept. It’s a good starting point for developing more advanced and feature-rich management systems. The flexibility and modularity of the current design ensure that it can be easily expanded and adapted to meet future requirements.

# References

* *W3Schools.com*. (n.d.). <https://www.w3schools.com/c/c_intro.php>
* GeeksforGeeks. (2023, September 25). *Hospital Management System in C*. GeeksforGeeks. <https://www.geeksforgeeks.org/c-program-for-hospital-management-system/>
* *Hospital Management System Project using C Language - StudyTonight*. (n.d.). <https://www.studytonight.com/c-projects/hospital-management-system-project-using-c-language>
* *C struct (Structures)*. (n.d.). <https://www.programiz.com/c-programming/c-structures>

# Workload Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Criteria** | NWAY  YUPAR  AUNG | SANTIAGO GALLARDO DOMINGUEZ | SARA ALSABSABI | ANAS WADHAH MOHAMMED QAID | **Total** |
| Design | 25% | 25% | 25% | 25% | 100% |
| Coding / Implementation (C code) | 20% | 40% | 20% | 20% | 100% |
| Documentation | 25% | 25% | 25% | 25% | 100% |
| Demonstration | 25% | 25% | 25% | 25% | 100% |