BIGLEARN TRAINING INSTITUTE TRICHY



HOSPITAL MANAGEMENT SYSTEM

GUIDED BY: BY

AGNESMARRY.V E.G.PRADEEP

**INTRODUCTION**

A Hospital Management System (HMS) is a comprehensive software solution designed to streamline the operations of a healthcare facility. It integrates various functions like patient registration, appointment scheduling, medical records management, billing, and inventory control into a single platform. It ensures that vital information is easily accessible, allowing healthcare professionals to make informed decisions quickly. Additionally, it aids in maintaining accurate records, minimizing errors, and facilitating seamless communication across different departments within the hospital. This application is built using c language to enhance its functionality.

**FEATURES**

**Patient Management**: Handles patient registration, appointment scheduling, and maintains comprehensive patient records, including medical history and treatment details.

**Appointment Scheduling**: Allows patients to book appointments online, and helps staff manage the schedules of doctors and other healthcare professionals efficiently.

**Electronic Medical Records (EMR)**: Stores patient medical records electronically, ensuring easy access and secure storage of patient data.

**Billing and Invoicing**: Manages billing processes, including insurance claims, payments, and invoices, ensuring accurate and timely financial transactions.

**SYSTEM CONFIGURATION**

PROCESSOR Intel(R) Core(TM) I5-8265U CPU @ 1.60GHz 1.80GHz

Installed RAM 8.00 GB (7.88 GB usable)

System type 64-bit operating system, x64-based processor

**SYSTEM REQUIREMENTS**

1. GDB

**C PAKAGES USED**

**Standard I/O Library (stdio.h):**

* Provides functions for input and output operations, such as printf, scanf, fopen, and fclose.

**String Manipulation Library (string.h)**:

* Offers functions for handling and manipulating strings, such as strcpy, strcmp, strlen, and strcat.

**Standard Library (stdlib.h)**:

* Includes functions for memory allocation (malloc, free), process control (exit), and other utility functions.

**Time Library (time.h)**:

* Provides functions to manipulate date and time, such as time, difftime, and strftime.

**GDB**

GDB (GNU Debugger) is a powerful tool used to debug programs written in C (and other languages). It allows developers to run their programs step-by-step to identify where things go wrong. With GDB, you can set breakpoints to pause the execution at specific lines, inspect variables to see their values at different stages, and follow the program flow to understand its behavior. This helps in diagnosing issues like crashes, incorrect results, or unexpected behavior, making it easier to fix bugs and improve the code's reliability. Overall, GDB is an essential tool for C programmers to ensure their programs work correctly.

**PROGRAM**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_NAME\_LENGTH 100

#define MAX\_GENDER\_LENGTH 10

#define MAX\_ADDRESS\_LENGTH 200

#define MAX\_ROLE\_LENGTH 100

#define MAX\_DEPARTMENT\_LENGTH 100

#define MAX\_CONTACT\_LENGTH 50

typedef struct {

int id;

char name[MAX\_NAME\_LENGTH];

int age;

char gender[MAX\_GENDER\_LENGTH];

char address[MAX\_ADDRESS\_LENGTH];

} Patient;

typedef struct {

int id;

char name[MAX\_NAME\_LENGTH];

char role[MAX\_ROLE\_LENGTH];

char department[MAX\_DEPARTMENT\_LENGTH];

char contact[MAX\_CONTACT\_LENGTH];

} Staff;

int patientCount = 0;

int staffCount = 0;

void addPatient() {

FILE \*fp;

Patient p;

fp = fopen("patients.txt", "a");

if (fp == NULL) {

printf("Error opening file!\n");

return;

}

printf("Enter patient ID: ");

scanf("%d", &p.id);

printf("Enter patient name: ");

scanf(" %[^\n]", p.name);

printf("Enter patient age: ");

scanf("%d", &p.age);

printf("Enter patient gender: ");

scanf("%s", p.gender);

printf("Enter patient address: ");

scanf(" %[^\n]", p.address);

fprintf(fp, "%d %s %d %s %s\n", p.id, p.name, p.age, p.gender, p.address);

fclose(fp);

patientCount++;

printf("Patient added successfully.\n");

}

void addStaff() {

FILE \*fp;

Staff s;

fp = fopen("staff.txt", "a");

if (fp == NULL) {

printf("Error opening file!\n");

return;

}

printf("Enter staff ID: ");

scanf("%d", &s.id);

printf("Enter staff name: ");

scanf(" %[^\n]", s.name);

printf("Enter staff role: ");

scanf("%s", s.role);

printf("Enter staff department: ");

scanf("%s", s.department);

printf("Enter staff contact: ");

scanf("%s", s.contact);

fprintf(fp, "%d %s %s %s %s\n", s.id, s.name, s.role, s.department, s.contact);

fclose(fp);

staffCount++;

printf("Staff added successfully.\n");

}

void viewPatients() {

FILE \*fp;

Patient p;

fp = fopen("patients.txt", "r");

if (fp == NULL) {

printf("Error opening file!\n");

return;

}

printf("Patients List:\n");

while (fscanf(fp, "%d %s %d %s %s", &p.id, p.name, &p.age, p.gender, p.address) != EOF) {

printf("ID: %d\nName: %s\nAge: %d\nGender: %s\nAddress: %s\n\n",

p.id, p.name, p.age, p.gender, p.address);

}

fclose(fp);

}

void viewStaff() {

FILE \*fp;

Staff s;

fp = fopen("staff.txt", "r");

if (fp == NULL) {

printf("Error opening file!\n");

return;

}

printf("Staff List:\n");

while (fscanf(fp, "%d %s %s %s %s", &s.id, s.name, s.role, s.department, s.contact) != EOF) {

printf("ID: %d\nName: %s\nRole: %s\nDepartment: %s\nContact: %s\n\n",

s.id, s.name, s.role, s.department, s.contact);

}

fclose(fp);

}

void menu() {

int choice;

while (1) {

printf("Hospital Management System\n");

printf("1. Add Patient\n");

printf("2. Add Staff\n");

printf("3. View Patients\n");

printf("4. View Staff\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addPatient();

break;

case 2:

addStaff();

break;

case 3:

viewPatients();

break;

case 4:

viewStaff();

break;

case 5:

exit(0);

default:

printf("Invalid choice. Please try again.\n");

}

}

}

int main() {

menu();

return 0;

}

**OUTPUT**

****

**CONCLUTION**

In conclution,A hospital management system is a comprehensive software solution designed to streamline and automate the administrative, clinical, and financial operations of a hospital. It improves efficiency by managing patient records, appointments, billing, inventory, and staff schedules in one integrated platform.This system enhances patient care by providing quick access to accurate information, reducing errors, and facilitating better coordination among healthcare providers.