**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**Jnana Sangama, Belgaum – 590018, Karnataka.**



**A Mini Project Report On**

# “HOSPITAL MANAGEMENT SYSTEM”

*Submitted in partial fulfilment of the requirement for the award of*

**BACHELOR OF ENGINEERING**

In

**INFORMATION SCIENCE AND ENGINEERING**

*Submitted by:*

**ANIRUDDHA L (1AP18IS004)**

**NAGABUSHAN V (1AP18IS016)**

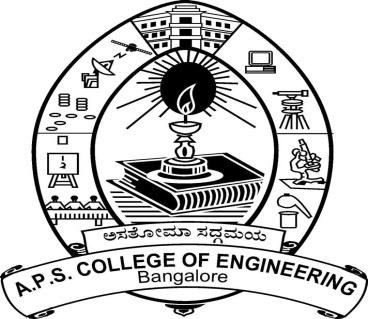
**RANJITHA H R (1AP18IS022)**

*Under the guidance of:*

## Mrs. Sushma s

Asst. Professor

Dept. of IS & E



**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**

# APS COLLEGE of ENGINEERING

Anantha Gnana Gangothri,

26th k.m, NH-209, Kanakapura Road, Bangalore-560082.

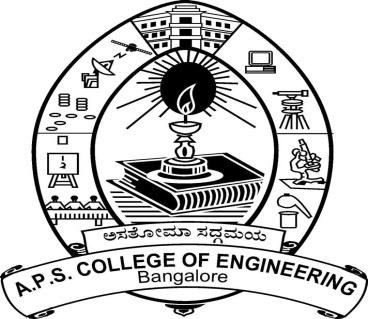
2020-2021

# APS COLLEGE of ENGINEERING

Anantha Gnana Gangothri,

26th k.m, NH-209, Kanakapura Road, Bangalore-560082.

**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**



## CERTIFICATE

Certified that the Project Work entitled "HOSPITAL MANAGEMENT SYSTEM" has been punctually carried out at APS College of Engineering, Bangalore by ANIRUDDHA L(1AP18IS004)

NAGABUSHAN V(1AP181S016) and RANJITHA H R (IAP171S032), bonafide student/s of sixth Semester, B.E. in partial fulfillment for the award of degree in Bachelor of Engineering in

Information Science & Engineering affiliated to Visvesvaraya Technological University, Belgaum during academic year 2020-2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

---------------------------- ---------------------------

**Mrs. Sushma s Dr. Nandeeswar S B**

Asst. Professor, Assoc. Professor &

Dept. of IS & E, Head Dept. of IS & E,

APSCE

**Name of the examiners Signature and Date** 1.

2.

. APSCE.



We hereby declare that the project work entitled "Hospital Management System" done at APS COLLEGE OF ENGINEERING, Bangalore, submitted to Visvesvaraya Technological University, in partial fulfillment of requirements for the degree of Bachelor of Engineering in Information Science & Engineering is a record of original work done by us and no part of it has been submitted for any degree or diploma of any institution previously.

Place: BENGALURU

Date:

Signature of the Students

ANIRUDHA L 1AP18IS004

NAGABUSHAN V 1AP18IS016

RANJITHA H R 1AP18IS022

#### ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task, but incomplete without the mention of the people who made it possible, whose constant and guidance and encouragement crowned my efforts with success.

I express my profound feelings of gratitude to this great institution of ours, A.P.S College of Engineering for providing me the excellent facilities.

I would like to thank Dr. B M Satish, Principal of A.P.S College of Engineering, for having permitted me to carry out the project on HOSPITAL MANAGEMENT SYSTEM successfully.

I would like to express my deepest thanks to our Project guideProf.Sushama s, Assistant Professor, Department of Information Science & Engineering, for his encouragement and for their insightful comments and for sharing their invaluable knowledge and experience with me.I really appreciate their help to improve the quality of report.

I would also like to thank all the teaching and non-teaching staff members Department of Information Science & Engineering for their suggestions and support.

I am deeply grateful to my parents and friends who have been supportive with their tolerance and genuine inspiration, in equal parts.

**STUDENT NAME (USN)**

1.ANIRUDDHA L (1AP18IS004)

2.NAGABHUSHAN V (1AP18IS016)

3.RANJITHA H R (1AP18IS022)

|  |  |
| --- | --- |
| **TABLE OF CONTENTS**        Declaration | III |
| Acknowledgement | IV |
| Table of Contents | V |
| List of Figures | VII |

**Chapter 1 INTRODUCTION**  1

* 1. File system structure
  + 1.1.1 Introduction to FS 1
  + 1.1.2 History of FS 1
  + 1.1.3 Advantages of file structure 2
  + 1.1.4 Disadvantages of file structure 4
  + 1.2 Project 5
  + 1.2.1 Overview of project 6

**Chapter 2 SYSTEM ANALYSIS**

* + 2.1 Existing system 7
  + 2.2 Proposed system **8**
  + 2.2.1 Goals of proposed system 8
  + 2.2.2 Objective and scope of project 9
  + 2.2.3 Future scope 10
  + 2.2.4 Future of project 10
  + 2.2.5 Application of project 11

**Chapter 3 REQUIREMENT SPECIFICATION**

* + 3.1 Specification requirements 11
  + 3.1.1 Hardware specification 11
  + 3.1.2 Hardware Requirements 12
  + 3.1.3 Software specification 12
  + 3.1.4 Software Requirements 12

**Chapter 4 SYSTEM DESIGN**

* + 4.1 E-R diagram **13**

**Chapter 5 SYSTEM IMPLEMENTATION**

* + 5.1 Introduction 14
  + 5.2 Source Code 14
  + 5.3 Result **37**

**Chapter 6 SNAPSHOTS**

* + 6.1 Home screen 38
  + 6.2 Doctors login 39
  + 6.3 Add patient’s details 39
  + 6.4 display patient’s details 40
  + 6.5 Delete patients record 40
  + 6.6 Search patients details with gender 41
  + 6.7 Update patient’s record 41

**CONCLUSION**  42

**BIBLIOGRAPHY**  42

#### CHAPTER 1

#### INTRODUCTION

#### 1.1 FILE SYSTEM STRUCTURE

###### 1.1. INTRODUCTION TO FS

The file system structure is the most basic level of organization in an operating system. Almost all of the ways an operating system interacts with its users, applications, and security model are dependent upon the way it organizes files on storage devices. Providing a common file system structure ensures users and programs are able to access and write files.

File systems break files down into two logical categories:

* Shareable vs. unsharable files
* Variable vs. static files

*Shareable* files are those that can be accessed locally and by remote hosts; *unsharable* files are only available locally. *Variable* files, such as documents, can be changed at any time; *static* files, such as binaries, do not change without an action from the system administrator.

The reason for looking at files in this manner is to help correlate the function of the file with the permissions assigned to the directories which hold them. The way in which the operating system and its users interact with a given file determines the directory in which it is placed, whether that directory is mounted with read-only or read/write permissions, and the level of access each user has to that file. The top level of this organization is crucial. Access to the underlying directories can be restricted or security problems could manifest themselves if, from the top level down, it does not adhere to a rigid structure.

###### 1.1.2 HISTORY OF FS

## In 1963, researchers came up with the idea of AVL trees for data in memory.

## AVL trees, however, did not apply to files because they work well when tree nodes are composed of single records rather than dozens or hundreds of them.

## In the 1970’s came the idea of B-Trees which require an O(logk N) access time where N is the number of entries in the file and k, the number of entries indexed in a single block of the B-Tree structure --> B-Trees can guarantee that one can find one file entry among millions of others with only 3 or 4 trips to the disk.

###### 1.1.3 ADVANTAGES OF FILE STRUCTURE

**1. Cost Effective**

File management system is extremely cost effective since it is a digital filing system. Whatever the type of document that needs to be stored as a paper can be in the form of digital. Therefore, there is no cost involved in building rents, purchasing cabinets and physical papers.

And also, this type of management greatly reduces the space that the office occupies for the documents.

**2. Security**

The traditional method of storing files cannot match the level of security provided by the file management system. In fact security is one of the reason why many organizations prefer to use file management system. The documents stored in the file management system is protected using authentication methods like username and password.

Additionally, all the data is encrypted which makes sure a document is confidential.

**3. Reliability**

The data that is stored in a file management system is far more reliable than physically storing it using papers and files. Unlike traditional methods of storing data, files here is very less likely to undergo damage or destruction. Any damages from nature or handling can be completely avoided in a file management system.

All the data of the users are stored inside the servers. Therefore, users can ensure that their documents are secured from potential damages.

**4. Data Sharing**

Data sharing is one of the key features of a file management system. FMS allows very efficient way of sharing data with each and every person. The same data that is stored on files can be shared with multiple users simultaneously.

**5. Data Retrieval**

Using file management system means that it will be very easy to retrieve data. File management system follows a digital approach that provides access to required data within few minutes. Users don't need to search copies of documents manually here. Thus, there is very less amount of time spent for data retrieval.

**6. Data Backup**

In case of a failure, file management system provides a seamless way for backing up data. For this purpose, computers on default offer functionalities. However, if needed there can be also third party application programs be used.

**7. Environment Friendly**

Due to the fact that file management system follows a digital system and there is no paper works involved, it can be said that this technique is more environment friendly.

###### 1.1.4 DISADVANTAGES OF FS

**1. Redundancy**

Redundancy is a kind of duplication that occurs if the same type of information exists in different locations. In this event, there is a possibility of memory wastage to take place resulting in higher storage costs.

**2. Inconsistency**

Due to the effect of data redundancy this often leads to data inconsistency. Which means that the same copies of data located in different places contain different values. For preventing this, there should be paper listing among different files.

**3. Accessibility**

Accessing data in file management system is not an easy process. It is not convenient as it should be. Whenever a user needs to access an information using different approaches, they must execute a special program.

**4. Integrity**

The data that is present on a file management system can get integrated. Meaning it is not correct and consistent. Most often this is caused in the presence of consistency constraints. Constraints are imposed by the programmers using programming codes. If the integrity continues, it can make the process of adding new constraints to be difficult.

**5. Atomicity**

Atomicity refers to the data that is incomplete. This often happens if the data is either completely entered or not entered at all. For an example, your system could fail in the middle of a transaction leading to data atomicity. Unlike in database management system, it is difficult to ensure atomicity in file management system.

**6. Data duplication**

Since data is stored in more than one location, there is a possibility of data duplication to take place. If file management system undergoes data duplication it will cause problems in the storage space. These duplications are difficult to correct due to the fact that they are independent to each other. Hence, it requires manual correction which can take time and effort.

**7. Data isolation**

If the data is stored in different locations, this could essentially mean that they are isolated in file management system. Under this circumstances, the formats of each file can vary significantly. As a result, extracting data from files can be difficult as it requires complex programming.

##### 1.2 PROJECT

###### 1.2.1 OVERVIEW OF PROJECT

The entire project mainly consists of modules, which are

* **Main**
* **Hospital**
* **Doctor**
* **patient**
* **Doctors**

**1.Add**

Patient’s id

Name

Gender[m/f]

Age

Description

**2.Search**

**3.Delete**

**4.Display**

**5.Update**

**6.Description based search**

**7.Gender based search**

* **Patient**

**1. Patient’s id**

**2**.**View patient details**

**CHAPTER 2**

**2. SYSTEM ANALYSIS**

**2.1 Existing system:**

The current manual system has a lot of paper work. To maintain the records of sale and service manually, is a Time-consuming task. With the increase in database, it will become a massive task to maintain the database. Requires large quantities of file cabinets, which are huge and require quite a bit of space in the office, which can be used for storing records of previous details. The retrieval of records of previously registered patients will be a tedious task. Lack of security for the records, anyone disarrange the records of your system. If someone want to check the details of the available doctors the previous system does not provide any necessary detail of this type.

All this work is done manually by the receptionist and other operational staff and lot of papers are needed to be handled and taken care of. Doctors have to remember various medicines available for diagnosis and sometimes miss better alternatives as they can’t remember them at that time.

**Advantages:**

1. No extra training required.

2. Easy to implement.

3. Can be stored anywhere.

4. Requires minimum effort.

**Disadvantages:**

1. Needs lots of paper.

2. Problem with maintenance.

3. Volumes of data becomes problem.

4. Once data is burned it cannot be reproduced easily.

5. Data handling is problem.

**2.2.1 Proposed system:**

The Hospital Management System is designed for any hospital to replace their existing manual paper-based system. The new system is to control the information of patients as well as doctors. These services are to be provided in an efficient, cost effective manner, with the goal of reducing the time and resources currently required for such tasks.

The complete set of rules & procedures related to Hospital’s day to day activities and generating report is called “Hospital Management System”. It is a computerized management system. This system also keeps the records of hardware assets besides software of this organization. The proposed system will keep a track of Doctors, Patients & Receptionist. This project has GUI based software that will help in storing, updating and retrieving the information through various user-friendly menu-driven modules.

**2.2.1 Goals of proposed system:**

i. The system should be easy to operate.

ii. The working in the organization will be well planned and organized.

iii. The level of accuracy in the proposed system will be higher.

iv. The reliability of the proposed system will be high due to proper storage of information.

v. Provide quick and efficient retrieval of information.

**Advantages:**

1. Low maintenance cost.

2. Volume of data is not an issue.

3. Data can be converted easily to information.

4. Data cannot be corrupted easily with proper backup

. 5. It can be expanded as well as data communication is possible.

**Disadvantages:**

1. High starting cost requires.

2. Additional manpower is necessary.

3. Data communication system will have an additional cost.

###### OBJECTIVE AND SCOPE OF PROJECT

**Objective**

Hospital are the essential part of our lives, providing best medical facilities to people suffering from various ailments, which may be due to change in climatic conditions, increased work-load, emotional trauma stress etc. It is necessary for the hospitals to keep track of its day-to-day activities & records of its patients, doctors, nurses, ward boys and other staff personals that keep the hospital running smoothly & successfully. But keeping track of all the activities and their records on paper is very cumbersome and error prone. It also is very inefficient and a time-consuming process Observing the continuous increase in population and number of people visiting the hospital. Recording and maintaining all these records is highly unreliable, inefficient and error-prone. It is also not economically & technically feasible to maintain these records on paper.

Thus keeping the working of the manual system as the basis of our project. We have developed an automated version of the manual system, named as “Administration support system for medical institutions”. The main aim of our project is to provide a paper-less hospital up to 90%. It also aims at providing low-cost reliable automation of the existing systems. The system also provides excellent security of data at every level of user-system interaction and also provides robust & reliable storage and backup facilities.

**2.2.3 FUTURE SCOPE**

FUTURE SCOPE All this work is done manually by the receptionist and other operational staff and lot of papers are needed to be handled and taken care of. Doctors have to remember various medicines available for diagnosis and sometimes miss better alternatives as they can’t remember them at that time. The limited time and resources have restricted us to incorporate, in this project, only main activities that are performed in a Hospital Management System, but utmost care has been taken to make the system efficient and user friendly.

Most of the analysis and interpretations, made for this report, are based on secondary data obtained. This data could have some inherent mistakes and errors. Finally, although due care has been taken those can be typing and compilation errors in the report itself. The tasks specified were not well defined because nothing was mentioned regarding validations in the project. Though we gave maximum effort to check the software. But it in no way alters the ultimate aim of the project and because it's highly USER FRIENDLY, it would be the choice of all kinds of personnel.

**2.2.4 FEATURES OF PROJECT**

* Doctor
* Doctor enter the patient’s details
* Manage patient’s, id/name/age
* View patient
* View patient’s details by search

**2.2.3 APPLICATION OF PROJECT**

In this project Implementation of hospital management system project helps to store all the kinds of records, provide coordination and user communication.

#### CHAPTER 3

#### REQUIREMENTS SPECIFICATION

##### 3.1 SPECIFICATION REQUIREMENTS

The Specific Requirements for **“Hospital management system”** is stated as follows:

**3.1.1 HARDWARE SPECIFICATION:**

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

###### 3.1.2 HARDWARE REQUIREMENTS

* Processor: Intel CORE i3
* RAM: 8GB
* Hard disk: 1TB

**3.2.3 SOFTWARE SPECIFICATION:**

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

###### 3.2.4 SOFTWARE REQUIREMENTS

* Operating system: Windows 7,9,10
* Front end design: C++

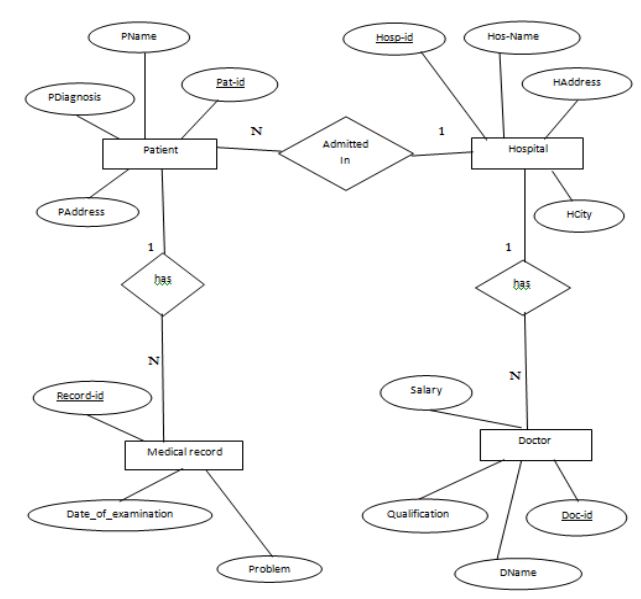
**CHAPTER 4**

##### 4.1 ENTITY RELATIONSHIP DIAGRAM

An entity–relationship model is usually the result of systematic analysis to define and describe what is important to processes in an area of a business. An E-R model does not define the business processes; it only presents a business data schema in graphical form. It is usually drawn in a graphical form as boxes (entities) that are connected by lines (relationships) which express the associations and dependencies between entities. Entities may be characterized not only by relationships, but also by additional properties (attributes), which include identifiers called "primary keys". Diagrams created to represent attributes as well as entities and relationships may be called entity-attributerelationship diagrams, rather than entity-relationship models.

An ER model is typically implemented as a database. In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a relational database a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity. There is a tradition for ER/data models to be built at two or three levels of abstraction. Note that the conceptuallogical-physical hierarchy below is used in other kinds of specification, and is different from the three-schema approach to software engineering. While useful for organizing data that can be represented by a relational structure, an entity-relationship diagram can't sufficiently represent semi-structured or unstructured data, and an ER Diagram is unlikely to be helpful on its own in integrating data into pre-existing information system.

Cardinality notations define the attributes of the relationship between the entities. Cardinalities can denote that an entity is optional.



**Fig. 3.2: Enhanced ER diagram of Automatic Timetable Generator**

#### CHAPTER 5

#### SYSTEM IMPLEMENTATION

#### 5.1 Introduction:

#### Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

**5.2 SOURCE CODE**

C++:

#### #include<stdio.h>

#### #include<cstring>

#### #include<fstream>

#### #include<conio.h>

#### #include<string.h>

#### #include<iostream>

#### using namespace std;

#### int no = 0;

#### char rec\_ind[5], rec\_gender[5],rec\_desc[55];

#### int rec\_flag = 0;

#### struct record

#### {

#### char age[5], ind[5];

#### char name[30];

#### char id[20];

#### char gender[10];

#### char description[100];

#### }rec[20];

#### struct index

#### {

#### char id[20], ind[20];

#### }in[20], temp;

#### void sort\_index()

#### {

#### int i, j;

#### for (i = 0;i < no - 1;i++)

#### for (j = 0; j < no - i - 1; j++)

#### if (strcmp(in[j].id, in[j + 1].id) > 0)

#### {

#### temp = in[j];

#### in[j] = in[j + 1];

#### in [j + 1] = temp;

#### }

#### }

#### void retrive\_record(char\* ind)

#### {

#### int flag = 0, i = 0;

#### fstream f9;

#### f9.open("record.txt", ios:in);

#### while (! f9.eof ())

#### {

#### f9.getline(rec[i].ind, 5, '|');

#### f9.getline(rec[i].id, 20, '|');

#### f9.getline(rec[i].name, 30, '|');

#### f9.getline(rec[i].age, 5, '|');

#### f9.getline(rec[i]. gender, 10, '|');

#### f9.getline(rec[i]. description, 100, '\n');

#### i++;

#### }

#### for (int j = 0;j < i;j++)

#### {

#### if (strcmp(rec[j].ind, ind) == 0)

#### {

#### strcpy\_s (rec\_ind, ind);

#### rec\_flag = 1;

#### std :: cout << "Patient record found\n";

#### std: cout << rec[j].id << "|" << rec[j].name << "|" << rec[j]. age << "|" << rec[j]. gender << "|" << rec[j]. description << "\n";

#### return;

#### }

#### }

#### }

#### int search\_index(char\* id)

#### {

#### int flag = 0;

#### fstream ff;

#### ff.open("index.txt", ios::in);

#### int i = 0;

#### while (!ff.eof())

#### {

#### ff.getline(in[i].id, 20, '|');

#### ff.getline(in[i].ind, 20, '\n');

#### i++;

#### }

#### for (int j = 0;j < i;j++)

#### {

#### if (strcmp(in[j].id, id) == 0)

#### {

#### retrive\_record(in[j].ind);

#### flag = 1;

#### }

#### }

#### //cout <<"flag is"<< flag ;

#### if (flag)

#### {

#### return 1;

#### }

#### else

#### return -1;

#### }

#### int search\_id(char\* id, int j)

#### {

#### int flag = 0;

#### for (int i = 0;i < j;i++)

#### if (strcmp(rec[i].id, id) == 0)

#### {

#### flag = 1;

#### break;

#### }

#### if (flag)

#### return 1;

#### else

#### return -1;

#### }

#### void delet(char\* st\_id)

#### {

#### rec\_flag = 0;

#### int fr = 0;

#### rec\_flag = search\_index(st\_id);

#### if (!rec\_flag)

#### {

#### std::cout << "deletion faild record not found\n";

#### return;

#### }

#### for (int i = 0;i < no;i++)

#### {

#### if (strcmp(rec[i].ind, rec\_ind) == 0)

#### {

#### fr = i;

#### break;

#### }

#### }

#### for (int i = fr;i < no - 1;i++)

#### {

#### rec[i] = rec[i + 1];

#### char str[3];

#### sprintf\_s(str, "%d", i);

#### strcpy\_s(rec[i].ind, str);

#### }

#### no--;

#### fstream f1, f2, f3;

#### f1.open("record.txt", ios::out);

#### f2.open("index.txt", ios::out);

#### f3.open("no.txt", ios::out);

#### f3 << no;

#### f3.close();

#### for (int i = 0;i < no; i++)

#### {

#### strcpy\_s(in[i].id, rec[i].id);

#### strcpy\_s(in[i].ind, rec[i].ind);

#### }

#### sort\_index ();

#### for (int i = 0;i < no;i++)

#### {

#### f1 << rec[i]. ind << "|" << rec[i].id << "|" << rec[i].name << "|" << rec[i]. age << "|" << rec[i]. gender << "|" << rec[i]. description << "\n";

#### f2 << in[i].id << "|" << in[i]. ind << "\n";

#### }

#### f1. close();

#### f2. close();

#### std::cout << "deletion successful\n";

#### }

#### int owner\_authen() {

#### char o\_username[20], o\_password[20];

#### char o\_user[20] = "APSC", o\_pass[20] = "apsc";

#### cout << endl << endl;

#### cout.width(25);

#### cout << "Enter Doctors User id and Password\n" << "------------------------------------------------------------------------------------\n";

#### cout << endl << "username: ";

#### cin >> o\_username;

#### cout << "password: ";

#### cin >> o\_password;

#### cout << "------------------------------------------------------------------------------------\n";

#### if (strcmp (o\_username, o\_user) == 0 && strcmp (o\_password, o\_pass) == 0)

#### return 1;

#### else

#### return 0;

#### }

#### void ret\_gender(char\* gender)

#### {

#### int flag = 0;

#### for (int i = 0;i < no;i++)

#### {

#### if (strcmp(rec[i].gender, gender) == 0)

#### {

#### strcpy\_s(rec\_gender, gender);

#### rec\_flag = 1;

#### std::cout << "patient record found\n";

#### std::cout << rec[i].id << "|" << rec[i].name << "|" << rec[i].age << "|" << rec[i].gender << "|" << rec[i].description << "\n";

#### }

#### }

#### }

#### void ret\_desc(char\* description)

#### {

#### int flag = 0;

#### for (int i = 0;i < no;i++)

#### {

#### if (strcmp(rec[i].description, description) == 0)

#### {

#### strcpy\_s(rec\_desc, description);

#### rec\_flag = 1;

#### std::cout << "patient record found\n";

#### std::cout << rec[i].id << "|" << rec[i].name << "|" << rec[i].age << "|" << rec[i].gender << "|" << rec[i]. description << "\n";

#### }

#### }

#### }

#### int main()

#### {

#### fstream file1, file2, file4;

#### cout << "---------------------------------------------------------------------------------------------------------------------";

#### std::cout << "\n\t\t\t \t \* HOSPITAL MANAGEMENT SYSTEM \* \n";

#### cout << "---------------------------------------------------------------------------------------------------------------------";

#### int ch;

#### char ind[5], st\_id[20], name[20], age[5], gender[10], description[30], id[10];

#### int i = 0, user;

#### label:

#### cout << "\n enter user : 1.doctor \t 2.patient \n";

#### std::cin >> user;

#### switch (user) {

#### case 1: {

#### int check;

#### check = owner\_authen();

#### if (check == 1)

#### {

#### cout << "Welcome Doctor\n" << "------------------------------------------------------------------------------------\n";

#### while (1)

#### {

#### std::cout << " \n \t 1.Add \n \t 2.Search \n \t 3.Delete \n \t 4.Display \n \t 5.Update \n \t 6.Description based search \n \t 7. Gender based search \n \t 8.exit \n";

#### cin >> ch;

#### switch (ch)

#### {

#### //add

#### case 1:

#### {

#### file1.open("record.txt", ios::app | ios::out);

#### int n;

#### cout << "\n Enter no of patients \t";

#### cin >> n;

#### cout << " Enter their details \n ";

#### file4.open("no.txt", ios::in);

#### file4 >> no;

#### for (i = no;i < no + n;i++)

#### {

#### cout << "\n Enter " << i + 1 << " patient \n";

#### cout << " \n enter patient's id \t";

#### cin >> rec[i].id;

#### std::cout << "\n Name: \t";

#### cin >> rec[i].name;

#### std::cout << "\n Gender[m/f]: \t";

#### cin >> rec[i].gender;

#### std::cout << "\n Age: \t";

#### cin >> rec[i].age;

#### std::cout << "\n Description: \t";

#### cin >> rec[i].description;

#### int q = search\_id(rec[i].id, i);

#### file1 << i << "|" << rec[i].id << "|" << rec[i].name << "|" << rec[i].age << "|" << rec[i].gender << "|" << rec[i].description << "\n";

#### }

#### file1.close();

#### no = no + n;

#### fstream file1, file3, file2;

#### file3.open("no.txt", ios::out);

#### file3 << no;

#### file3.close();

#### file2.open("index.txt", ios::out);

#### file1.open("record.txt", ios::in);

#### for (i = 0;i < no;i++)

#### {

#### file1.getline(ind, 5, '|');

#### file1.getline(id, 20, '|');

#### file1.getline(name, 30, '|');

#### file1.getline(age, 10, '|');

#### file1.getline(gender, 50, '|');

#### file1.getline(description, 30, '\n');

#### strcpy\_s(rec[i]. ind, ind);

#### strcpy\_s(in[i].id, id);

#### strcpy\_s(in[i].ind, ind);

#### }

#### sort\_index();

#### std::cout << "\n After sorting,index file contents are:\n";

#### for (i = 0; i < no; i++)

#### std:cout << in[i].id << " " << in[i].ind << endl;

#### for (i = 0;i < no;i++)

#### {

#### file2 << in[i].id << "|" << in[i].ind << "\n";

#### }

#### file1.close();

#### file2.close();

#### break;

#### }

#### //search

#### case 2:

#### {

#### fstream f4;

#### f4.open("no.txt", ios::in);

#### f4 >> no;

#### //cout << no;

#### cout << " \n Enter patient's id whose details are to be displayed: \t";

#### cin >> id;

#### int q = search\_index(id);

#### if (q == 1)

#### cout << "\n success search \n";

#### else

#### cout << "\n unsuccess search \t";

#### break;

#### }

#### //deletion

#### case 3: {

#### cout << "\n Enter patient id who is to be deleted \n ";

#### cin >> st\_id;

#### delet(st\_id);

#### break;

#### }

#### //display

#### case 4:

#### {

#### fstream file1;

#### file1.open("record.txt", ios::in);

#### std::cout << "ID \t NAME\t AGE \t GENDER \t DESCRIPTION \n";

#### while (!file1.eof())

#### {

#### file1.getline(ind, 5, '|');

#### file1.getline(id, 20, '|');

#### file1.getline(name, 30, '|');

#### file1.getline(age, 10, '|');

#### file1.getline(gender, 50, '|');

#### file1.getline(description, 30, '\n');

#### cout << id << "\t" << name << "\t" << age << "\t " << gender << "\t \t " << description << "\n" << endl;

#### }

#### file1.close();

#### break;

#### }

#### //exit

#### case 8:

#### {

#### cout << "\n Ending prog";

#### goto label;

#### }

#### //gender

#### case 7:

#### {

#### fstream f4;

#### f4.open("no.txt", ios::in);

#### f4 >> no;

#### //cout << no;

#### cout << "Enter the Gender to be searched upon: \t";

#### cin >> gender;

#### ret\_gender(gender);

#### break;

#### }

#### //desc search

#### case 6:

#### {

#### fstream f4;

#### f4.open("no.txt", ios::in);

#### f4 >> no;

#### //cout << no;

#### cout << "Enter the Description to be searched upon: \t";

#### cin >> description;

#### ret\_desc(description);

#### break;

#### }

#### //update

#### case 5:

#### {

#### rec\_flag = 0;

#### int fr = 0;

#### cout << "\n Enter patientt's id to be updated : \t";

#### cin >> st\_id;

#### rec\_flag = search\_index(st\_id);

#### if (rec\_flag == -1)

#### {

#### std::cout << "\n Failed record not found";

#### break;

#### }

#### for (int i = 0; i < no;i++)

#### {

#### if (strcmp(rec[i]. ind, rec\_ind) == 0)

#### {

#### std::cout << "\nThe old values of the patient record are ";

#### std::cout << "\n id = " << rec[i].id;

#### std::cout << "\n name = " << rec[i].name;

#### std::cout << "\n age = " << rec[i].age;

#### std::cout << "\n gender = " << rec[i].gender;

#### std::cout << "\n description = " << rec[i].description;

#### std::cout << "\nEnter the new values \n";

#### std::cout << "\nid = "; cin >> rec[i].id;

#### std::cout << "\nname = "; cin >> rec[i].name;

#### std::cout << "\nage = "; cin >> rec[i].age;

#### std::cout << "\ngender = "; cin >> rec[i].gender;

#### std::cout << "\n description = "; cin >> rec[i].description;

#### break;

#### }

#### }

#### fstream f1, f2;

#### f1.open("record.txt", ios::out);

#### f2.open("index.txt", ios::out);

#### for (int i = 0; i < no;i++)

#### {

#### strcpy\_s(in[i].id, rec[i].id);

#### strcpy\_s(in[i].ind, rec[i].ind);

#### }

#### sort\_index();

#### for (int i = 0;i < no;i++)

#### {

#### f1 << rec[i].ind << "|" << rec[i].id << "|" << rec[i].name << "|" << rec[i].age << "|" << rec[i].gender << "|" << rec[i].description << "\n";

#### f2 << in[i].id << "|" << in[i].ind << "\n";

#### }

#### f1.close();

#### f2.close();

#### std::cout << "\n updation successful\n";

#### }

#### }

#### }

#### }

#### else {

#### cout << "\n Invalid login";

#### }

#### break;

#### }

#### case 2: {

#### fstream f4;

#### f4.open("no.txt", ios::in);

#### f4 >> no;

#### cout << " \n Enter the Patient's id whose details are to be displayed \t";

#### cin >> id;

#### int q = search\_index(id);

#### if (q == 1)

#### cout << "\n success search";

#### else

#### cout << "\n unsuccess search";

#### break;

#### }

#### }

#### return 0;

#### }

##### 5.2 PROBLEM DESCRIPTION

Hospital management system, this system contains an admin side from where a user can manage all the records easily. The Admin plays an important role in the management of this system. In this project, the user has to perform all the main functions from the Admin side. Even it contains patient’s side and users side from where they can easily access the details.

The system used are:

* **Doctors Login**
* **hospital**
* **patient details**

The system details are as follows:

**Doctors Login**

“username”; “password”;

**Patient’s**

“patient id”; “Details”;

**Doctors Login**

“patient id”; “name”; “age”; “gender”;” designation”;

**system**

“add”; “display”; “delete”; “update”; “search”;

“description”;

##### 5.3 RESULT

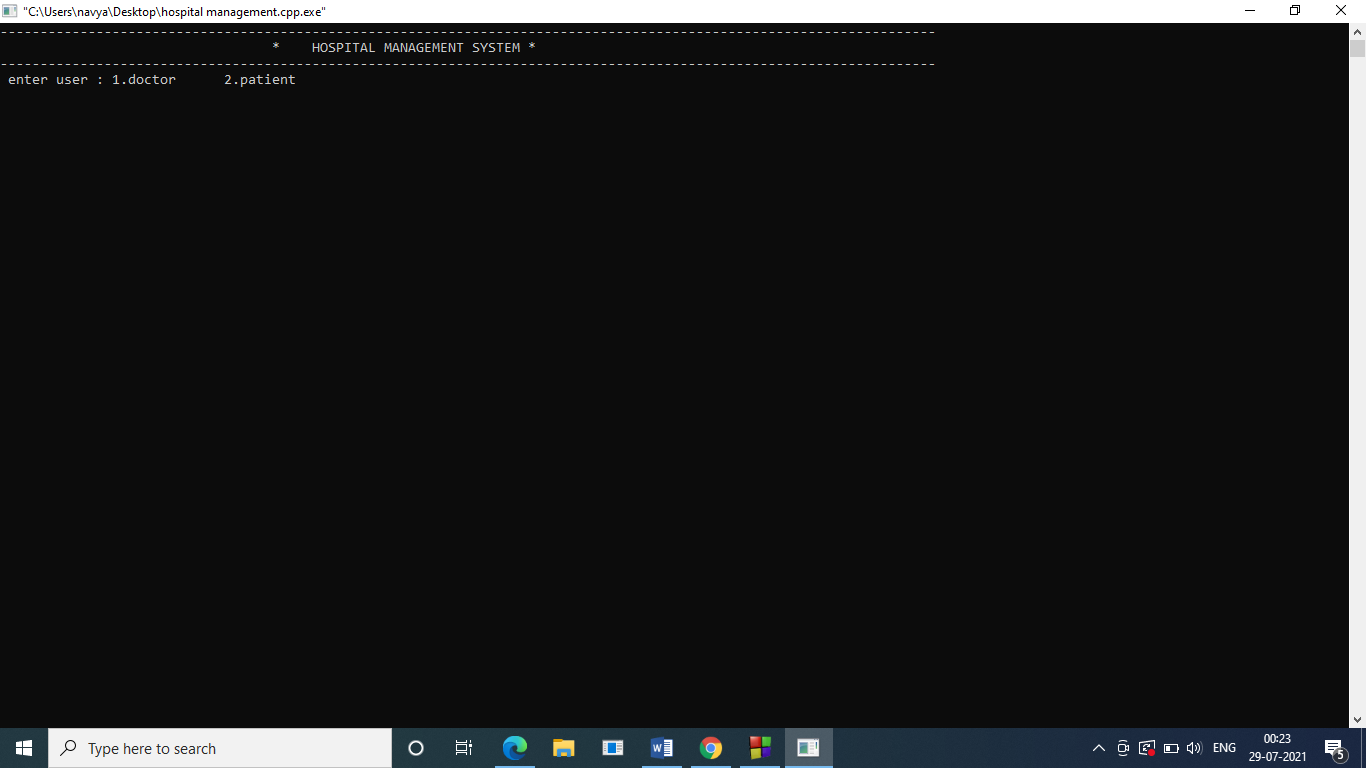
The resulting system is able to:

* Add patients id, name
* To generate the record
* To search data

#### CHAPTER 6

#### SNAPSHOTS

##### 6.1 HOME PAGE

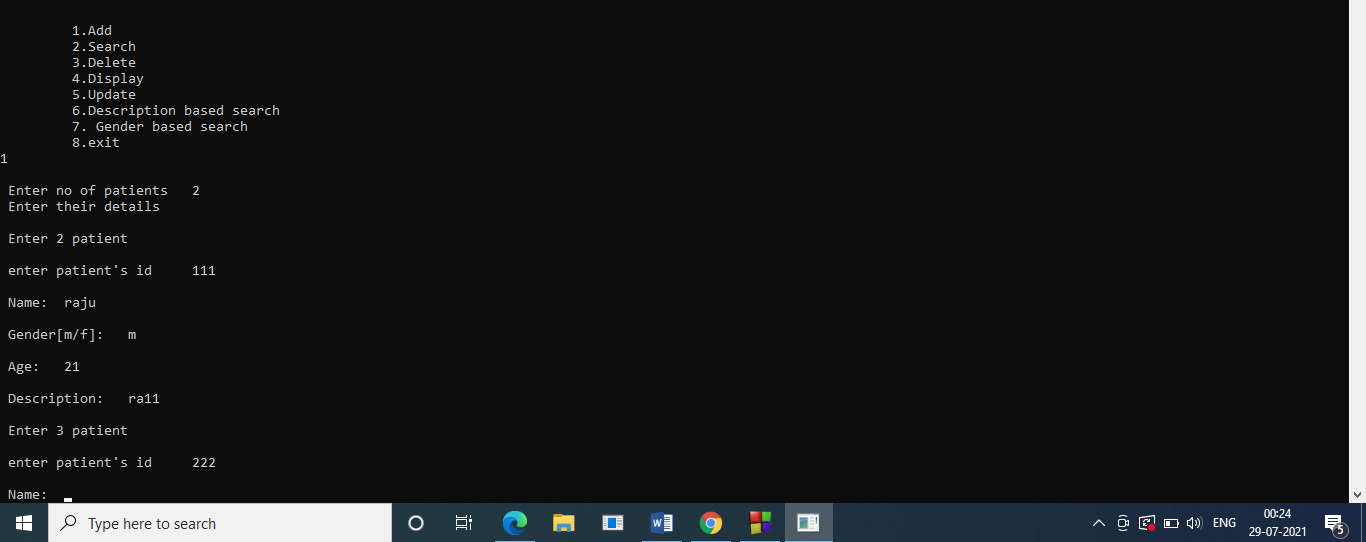
****

**Fig. 5.1: Home Page**

##### 6.2 DOCTORS LOGIN

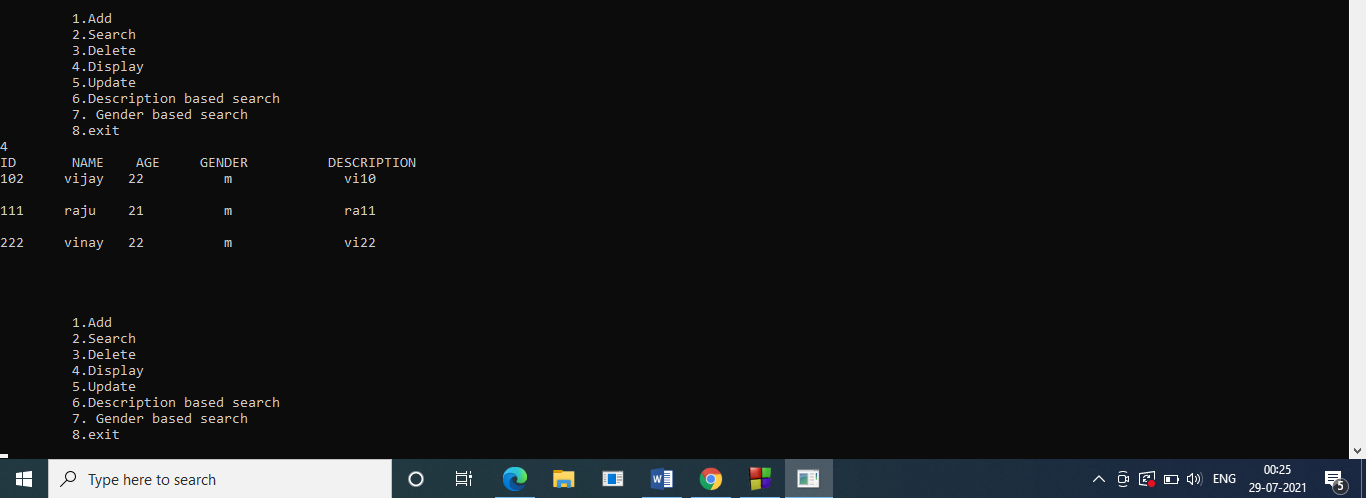
**Fig. 5.2: Doctors Login**

##### 6.3 ADD PATIENT’S DETAILS



**Fig. 5.3: Add patient’s details**

##### 6.4 PATIENT’S DETAILS

****

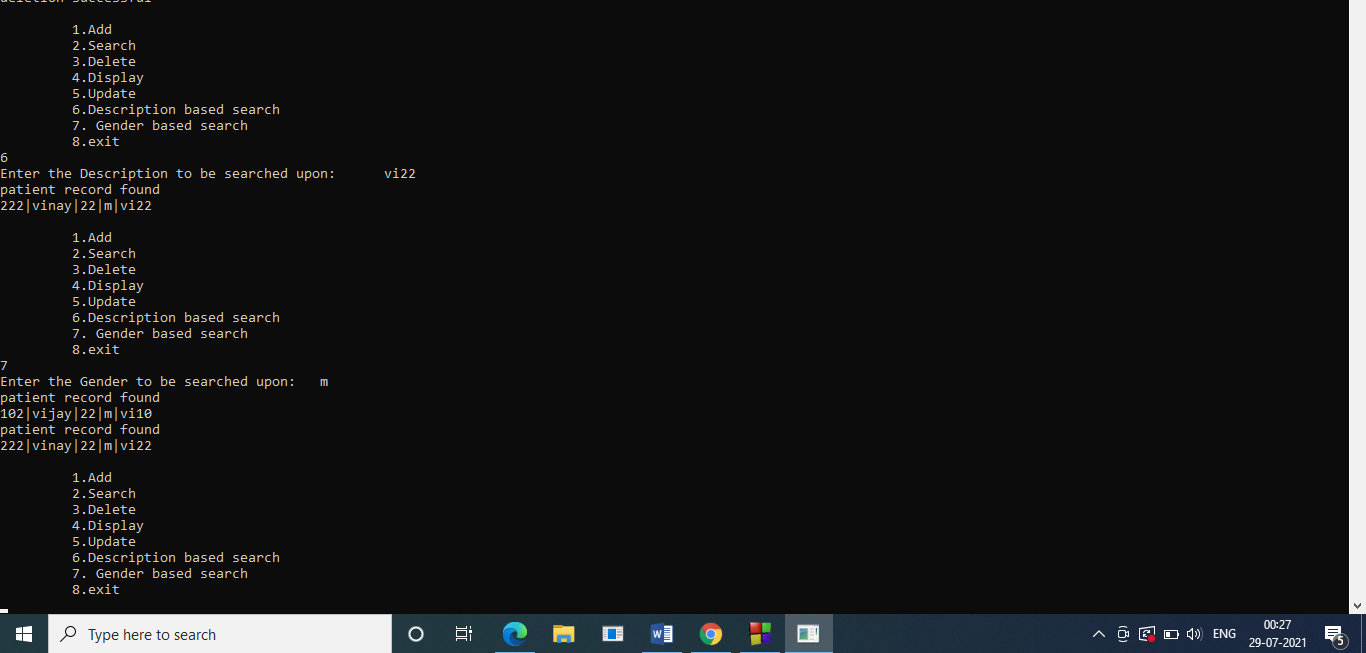
**Fig. 5.4: patients Details**

##### 6.5 DELETE PATIENT’S RECORD

****

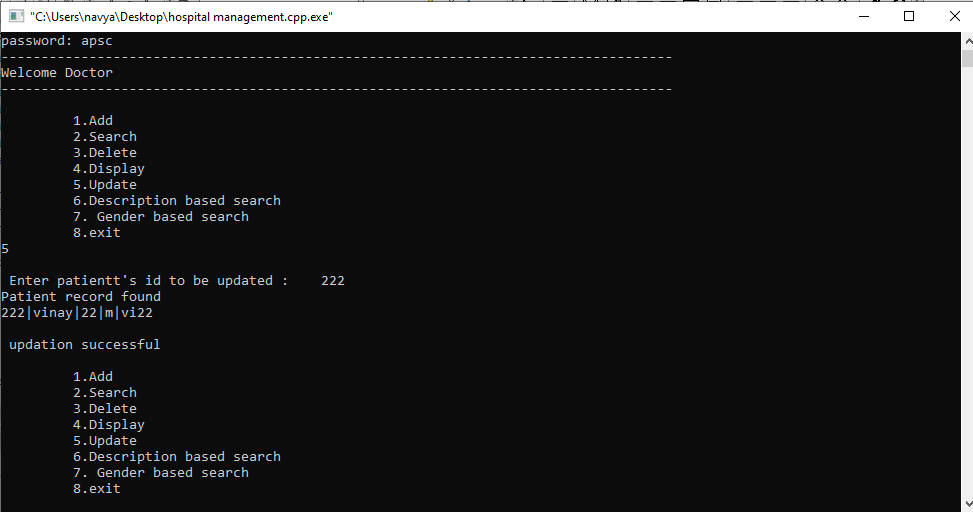
**Fig. 5.5: delete patients record**

##### 6.6 SEARCH PATIENT’S DETAILS WITH GENDER



**Fig. 5.6: search patients Details with gender**

**6.7 UPDATE PATIENT’S RECORD**



**Fig. 5.12: update patient’s record**

#### CONCLUSION

Since we are entering details of the patients electronically in the” Hospital Management System”, data will be secured. Using this application, we can retrieve patient’s history with a single click. Thus, processing information will be faster. It guarantees accurate maintenance of Patient details. It easily reduces the book keeping task and thus reduces the human effort and increases accuracy speed.

Hospital Management System is essential for maintaining detail about the Doctor, Patient, Hospital staff etc. we understand that by using of Hospital Management System project the work became very easy and we save lot of time. Hospital administrators would be able to significantly improve the operational control and thus streamline operations. This would enable to improve the response time to the demands of patient care because it automates the process of collecting, collating and retrieving patient information. Accounting sometimes becomes awfully pathetic and complex. This product will eliminate any such complexity.

#### BIBLIOGRAPHY

* Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson
* Abraham Silberschatz, Henry F. Korth and S. Sudarshan “Sixth Edition Database System Conceptsreleased”, January 28, 2010. 206-253 pp.
* Reference tutorial [https://www.youtube.com](https://www.youtube.com/watch?v=OuOZVoTwqm8)