## VISVESVARAYA TECHNOLOGICAL UNIVERSITY



**Jnana Sangama, Belagavi – 590018.**

**FILE STRUCTURE MINI PROJECT REPORT**

**ON**

## “Patient Information System”

Submitted in partial fulfillment for the requirement of VI semester for the

## Degree of Bachelor of Engineering in INFORMATION SCIENCE & ENGINEERING

For the academic year 2022-23

**SUBMITTED BY:**

## SURYA.M [1DB20IS148] SUPREET RAJ[1DB20IS146]

### Under the guidance of:

### Mrs. DEEPIKA A B

**Assistant Professor, Dept. of ISE**

### Department of Information Science and Engineering DON BOSCO INSTITUTE OF TECHNOLOGY

**Kumbalagodu, Bengaluru-560074**

2022-23

# DON BOSCO INSTITUTE OF TECHNOLOGY



## Kumbalagodu, Bengaluru-560074

**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

**CERTIFICATE**

This is to certify that the Mini Project Report entitled **“Patient Information System ”** is a bonafide Mini Project work carried out by **Surya.M (1DB20IS148)** & **Supreeth Raj (1DB20IS146)** in partial fulfillment of VI semester for the Degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belgaum, during the academic year 2022-23.It is certified that all corrections/suggestions indicated for Internal Assessments have been incorporated with the degree mentioned.

### Project Guide Head of Department

**Mrs. Deepika A B Prof. Dr. B.K. Raghavendra**

Assistant Professor Head of Department

Dept. of ISE Dept. of ISE

DBIT, Bengaluru DBIT, Bengaluru

Name of the Examiners

**External Viva**

Signature with Date

1.

2.

# ACKNOWLEDGEMENT

At the various stages in making the mini project, a number of people have given me invaluable comment on the manuscript. We take this opportunity to express my deepest gratitude and appreciation to all those who helped me directly or indirectly towards the successful completion of this project.

We would like to thank our Principal **Prof. Nagabhushana B S**, Don Bosco Institute

of Technology for his support though out this project.

We express my whole hearted gratitude to **Prof. Dr. B.K. Raghavendra,** who is our respectable Head of Dept. of Information Science. We wish to acknowledge for his valuable help and encouragement.

In this regard We owe a heartfelt gratitude to my guide **Mrs. Deepika A B** Assistant Professor of Department of Information Science and Engineering, for timely advice on the project and regular assistance throughout the project work. We would also like to thank the teaching and non-teaching staff members of Department of Information Science and Engineering for their corporation.

**SURYA.M (1DB20IS148) SUPREETH RAJ (1DB20IS146)**

# ABSTRACT

A “Patient Information System” is a comprehensive digital platform designed to efficiently manage and organize patient-related data within a healthcare facility. It includes file operations such as storing, retrieving, searching, deleting and updating patient information, ensuring that healthcare providers have access to accurate and up-to-date data for effective decision-making and care coordination. A file structure is a combination of representations for data in files. It is also a collection of operations for accessing the data. It enables applications to read, write, and modify data. File structures may also help to find the data that matches certain criteria. Indexing involves creating a separate data structure that contains pointers to the location of the data in the file structure. This allows the system to quickly locate and retrieve data without having to search through the entire file structure. Hashing is a technique that can be used in patient management software to improve the efficiency of data storage and retrieval. Hashing involves taking a value, such as a patient unique id, and using it to generate a shorter fixed length value called a hash. The system typically includes functionalities such as patient full name, date of birth, gender, address and phone number, patient history, blood group and clinical decision support. By streamlining administrative tasks and improving data accessibility, a patient information system enhances the overall quality of patient care, enhances communication among healthcare professionals, and increases operational efficiency within healthcare organizations. Overall, a patient information system aims to streamline and enhance the management of patient information, ultimately improving patient care, safety, and outcomes. By centralizing data, ensuring accessibility, and supporting clinical decision-making, the system plays a vital role in modern healthcare delivery.

# CONTENTS

|  |  |  |
| --- | --- | --- |
| **Sl. No** | **CHAPTER** | Pg. No |
| **1** | **INTRODUCTION** | 1 |
| **1.1** | **Overview of file structures** | 2 |
| **1.2** | **Problem statement** | 2 |
| **1.3** | **Aim and Objectives** | 3 |
| **1.4** | **Existing system** | 4 |
| **2** | **PROPOSED SYSTEM** | 5 |
| **2.1** | **Advantages** | 5 |
| **3** | **SYSTEM REQUIREMENTS** | 6 |
| **3.1** | **Software (Front end & Back end) requirements** | 6 |
| **3.2** | **Hardware requirements** | 6 |
| **4** | **IMPLEMENTATION** | 7-9 |
| **4.1** | **File creation** | 7 |
| **4.2** | **Description of methods** | 8-9 |
| **5** | **SNAPSHOTS** | 17-19 |
| **6** | **CONCLUSION** | 20 |
|  | **REFERENCES** | 21 |

**CHAPTER 1 INTRODUCTION**

A Patient information system is an element of health informatics that focuses mainly on the administrational needsof hospitals. In many implementations, a PIS is a comprehensive, integrated information system designed to manage all the aspects of a hospital's operation, such as medical, administrative, financial , and legal issues and the corresponding processing of services. Hospital information system is also known as patient management software (PMS) or Patient management system.

Patient information systems provide a common source of information about a patient's health history, and doctorsschedule timing. The system has to keep data in a secure place and controls who can reach the data in certain circumstances. These systems enhance the ability of health care professionals to coordinate care by providing a patient's health information and visit history at the place and time that it is needed. Patient's laboratory test information also includes visual results such as X-ray, which may be reachable by professionals. PIS provide internaland external communication among health care providers.

The frontend is responsible for providing a user-friendly interface for users to interact with the system. The system's front-end comprises HTML, CSS, and JavaScript, which are responsible for rendering the graphical user interface(GUI) and handling user inputs. HTML is used to structure and organize the content of web pages, while CSS is responsible for styling the pages and creating visually appealing layouts. JavaScript, on the other hand, enables dynamic interactions between the user and the system, allowing for real-time updates and instant feedback.

The back-end of the Patient information System is built using Python, which provides robust functionality and flexibility for processing and storing data. The system employs a file-based database approach, where Patient related information is stored. The use of file-based databases allows for easy data manipulation and quick access to patient details. In summary, the patient information System is a software solution that utilizes a combination offront-end and back-end technologies such as HTML, CSS, JavaScript, and Python, and employs indexing and hashing techniques for optimized data retrieval and storage.

## Overview of the file structure

Hospitals currently use a manual system for the management and maintenance of critical information. The current system requires numerous paper forms, with data stores spread throughout the Patient management infrastructure. Often information (on forms) is incomplete, or does not follow management standards. Forms are often lost in transit between departments requiring a comprehensive auditing process to ensure that no vital information is lost. This system provides more detailed information about patients .We can also retrieve information of patients through search and the stored information is not lost.

## Problem statement

The Existing system includes problems like lack of time consuming, accuracy, high cost, security problems etc. The following problem statement highlights these issues:

* + - Inefficiency and Manual Processes.
    - Lack of Transparency and Auditability.
    - Security and Data Integrity.
    - Limited Scalability and Flexibility.
    - Complex Reconciliation and Reporting.
    - Data Analysis and Business Insights.

.

## Aim and Objectives Aim:

The Aim of Patient Information System is to efficiently record, manage, and process financial transactions, ensuring accuracy, security, transparency, and compliance within organizations.

## Objective:

* + - Automation and Efficiency.
    - Accuracy and Auditability.
    - Security and Data Protection.
    - Scalability and Adaptability.
    - Integration and Interoperability.
    - Reporting and Analytics.

## Existing system

Hospitals currently use a manual system for the management and maintenance of critical information. The current system requires numerous paper forms, with data stores spread throughout the Patient management infrastructure. Often information (on forms) is incomplete, or does not follow management standards. Forms are often lost in transit between departments requiring a comprehensive auditing process to ensure that no vital information is lost. Multiple copies of the same information exist in the hospital and may lead to inconsistencies in data in various data stores. A significant part of the operation of any hospital involves the acquisition, management and timely retrieval of great volumes of information. This information typically involves; patient personal information and medical history, doctor information, scheduling, and various facilities waiting lists. All of this information must be managed in an efficient and cost wise fashion so that an institution’s resources may be effectively utilized PMS will automate the management of the hospital making it more efficient and error free. It aims at standardizing data, consolidating data ensuring data integrity and reducing inconsistencies.

# CHAPTER 2 PROPOSED SYSTEM

There are many activities that can be computerized in this widely spread organization. The proposed system for managing patient information is a digitalized system that leverages modern technology Python,and indexing hashing file structure. This system aims to automate many of the processes that are currentlydone manually in the existing system, reducing the potential for errors and making the system more efficient. In the proposed system, healthcare faculty can access the patient information through a website.They can search for patient with unique id, view patient details and update, and make appointments and delete patient record online. The system stores all the necessary data in an indexed and hashed file structure,making it easy to retrieve and manage large amounts of data quickly.

## ADVANTAGES

* + - Enhanced Efficiency: By digitizing and automating various tasks, patient information systems streamline administrative processes, reduce paperwork, and enable faster access to patient information, leading to improved efficiency and productivity for healthcare providers.

* + - Improved Patient Care: Having a comprehensive and up-to-date electronic health record allows healthcare professionals to make better-informed decisions, coordinate care effectively, and providemore personalized and timely treatments to patients.

* + - Increased Accuracy and Safety: Patient information systems minimize the risks of errors associatedwith manual data entry and paper-based records. They provide alerts for drug interactions, allergies,and other critical information, promoting patient safety and reducing medical errors.

# CHAPTER 3

**SYSTEM REQUIREMENTS**

## Software (Front end & Back end) requirements

* + - Processor: Intel dual Core, i3
    - RAM: 1 GB
    - Hard Disk: 80 GB

## HARDWARE REQUIREMENTS

* + - Programming Language: Python
    - IDE Used: Visual Studio Codes

# CHAPTER 4

**IMPLEMENTATION**

## File creation

**ADD –**This option allows the user to add patient and the details of patient .

**Function:**

class InsertWindow:

def \_\_init\_\_(self):

self.window = tkinter.Tk()

self.window.wm\_title("Insert data")

self.window.configure(bg="#7ab6ea")

# Initializing all the variables

self.id = tkinter.StringVar()

self.fName = tkinter.StringVar()

self.lName = tkinter.StringVar()

self.address = tkinter.StringVar()

self.phone = tkinter.StringVar()

self.email = tkinter.StringVar()

self.history = tkinter.StringVar()

self.doctor = tkinter.StringVar()

self.genderList = ["Male", "Female", "Transgender", "Other"]

self.dateList = list(range(1, 32))

self.monthList = ["January", "February", "March", "April", "May", "June", "July", "August", "September",

"October", "November", "December"]

self.yearList = list(range(1900, 2020))

self.bloodGroupList = ["A+", "A-", "B+", "B-", "O+", "O-", "AB+", "AB-"]

# Labels

tkinter.Label(self.window, text="Patient ID", width=25).grid(pady=5, column=1, row=1)

tkinter.Label(self.window, text="First Name", width=25).grid(pady=5, column=1, row=2)

tkinter.Label(self.window, text="Last Name", width=25).grid(pady=5, column=1, row=3)

tkinter.Label(self.window, text="D.O.B", width=25).grid(pady=5, column=1, row=4)

tkinter.Label(self.window, text="M.O.B", width=25).grid(pady=5, column=1, row=5)

tkinter.Label(self.window, text="Y.O.B", width=25).grid(pady=5, column=1, row=6)

tkinter.Label(self.window, text="Gender", width=25).grid(pady=5, column=1, row=7)

tkinter.Label(self.window, text="Home Address", width=25).grid(pady=5, column=1, row=8)

tkinter.Label(self.window, text="Phone Number", width=25).grid(pady=5, column=1, row=9)

tkinter.Label(self.window, text="Email ID", width=25).grid(pady=5, column=1, row=10)

# Fields

# Entry widgets

self.idEntry = tkinter.Entry(self.window, width=25, textvariable=self.id)

self.fNameEntry = tkinter.Entry(self.window, width=25, textvariable=self.fName)

self.lNameEntry = tkinter.Entry(self.window, width=25, textvariable=self.lName)

self.addressEntry = tkinter.Entry(self.window, width=25, textvariable=self.address)

self.phoneEntry = tkinter.Entry(self.window, width=25, textvariable=self.phone)

self.emailEntry = tkinter.Entry(self.window, width=25, textvariable=self.email)

self.historyEntry = tkinter.Entry(self.window, width=25, textvariable=self.history)

self.doctorEntry = tkinter.Entry(self.window, width=25, textvariable=self.doctor)

self.idEntry.grid(pady=5, column=3, row=1)

self.fNameEntry.grid(pady=5, column=3, row=2)

self.lNameEntry.grid(pady=5, column=3, row=3)

self.addressEntry.grid(pady=5, column=3, row=8)

self.phoneEntry.grid(pady=5, column=3, row=9)

self.emailEntry.grid(pady=5, column=3, row=10)

self.historyEntry.grid(pady=5, column=3, row=12)

self.doctorEntry.grid(pady=5, column=3, row=13)

# Combobox widgets

self.dobBox = tkinter.ttk.Combobox(self.window, values=self.dateList, width=20)

self.mobBox = tkinter.ttk.Combobox(self.window, values=self.monthList, width=20)

self.yobBox = tkinter.ttk.Combobox(self.window, values=self.yearList, width=20)

self.genderBox = tkinter.ttk.Combobox(self.window, values=self.genderList, width=20)

self.bloodGroupBox = tkinter.ttk.Combobox(self.window, values=self.bloodGroupList, width=20)

self.dobBox.grid(pady=5, column=3, row=4)

self.mobBox.grid(pady=5, column=3, row=5)

self.yobBox.grid(pady=5, column=3, row=6)

self.genderBox.grid(pady=5, column=3, row=7)

self.bloodGroupBox.grid(pady=5, column=3, row=11)

# Button widgets

tkinter.Button(self.window, width=20, text="Insert", command=self.Insert).grid(pady=15, padx=5, column=1,row=14)

tkinter.Button(self.window, width=20, text="Reset", command=self.Reset).grid(pady=15, padx=5,

column=2, row=14)

self.windowdestroy.mainloop()

def Insert(self):

self.values = Values()

self.database = Database()

self.test = self.values.Validate(self.idEntry.get(), self.fNameEntry.get(), self.lNameEntry.get(),

self.phoneEntry.get(), self.emailEntry.get(), self.historyEntry.get(), self.doctorEntry.get())

if (self.test == "SUCCESS"):

self.database.Insert(self.idEntry.get(), self.fNameEntry.get(), self.lNameEntry.get(), self.dobBox.get(),self.mobBox.get(), self.yobBox.get(), self.genderBox.get(), self.addressEntry.get(), self.phoneEntry.get(), self.emailEntry.get(), self.bloodGroupBox.get(),

self.historyEntry.get(), self.doctorEntry.get())

tkinter.messagebox.showinfo("Inserted data", "Successfully inserted the above data in the database")

else:

self.valueErrorMessage = "Invalid input in field " + self.test

tkinter.messagebox.showerror("Value Error", self.valueErrorMessage)

**SEARCH AND DELETE:** Here search and delete function allows us to search and delete or remove the details.

**Function:**

class SearchDeleteWindow:

def \_\_init\_\_(self, task):

window = tkinter.Tk()

window.wm\_title(task + " data")

# Initializing all the variables

self.id = tkinter.StringVar()

self.fName = tkinter.StringVar()

self.lName = tkinter.StringVar()

self.heading = "Please enter Patient ID to " + task

# Labels

tkinter.Label(window, text=self.heading, width=50).grid(pady=20, row=1)

tkinter.Label(window, text="Patient ID", width=10).grid(pady=5, row=2)

# Entry widgets

self.idEntry = tkinter.Entry(window, width=5, textvariable=self.id)

# Button widgets

if (task == "Search"):

def Search(self):

self.database = Database()

self.data = self.database.Search(self.idEntry.get())

self.databaseView = DatabaseView(self.data)

def Delete(self):

self.database = Database()

self.database.Delete(self.idEntry.get())

**UPDATE:** Here update function allows us to modify the patient details using there patient id.

**Function:**

def Update(self):

self.updateIDWindow = tkinter.Tk()

self.updateIDWindow.wm\_title("Update data")

# Initializing all the variables

self.id = tkinter.StringVar()

# Label

tkinter.Label(self.updateIDWindow, text="Enter the ID to update", width=50).grid(pady=20, row=1)

# Entry widgets

self.idEntry = tkinter.Entry(self.updateIDWindow, width=5, textvariable=self.id)

self.idEntry.grid(pady=10, row=2)

# Button widgets

tkinter.Button(self.updateIDWindow, width=20, text="Update", command=self.updateID).grid(pady=10, row=3)

self.updateIDWindow.mainloop()

## 4.1Description of methods

### INDEXES:

All indexes are based on same basic concepts-keys and reference fields. The types of index we see are called simple indexes because they are represented using simple arrays of structures that contain the keys and reference fields.

A Simple Index for Entry-Sequenced Files

[**Simple index:** An index in which the entries are a key ordered linear list.](http://www.blogger.com/blogger.g?blogID=1341810718493854175)

* + - Simple indexing can be useful when the entire index can be held in memory.

* + - Changes (additions and deletions) require both the index and the data file to be changed.

* + - Updates affect the index if the key field is changed, or if the record is moved.

* + - An update which moves a record can be handled as a deletion followed by an addition.

Object Oriented Support for Indexed, Entry Sequenced Files

[**Entry-sequenced file:** A file in which the record order is determined by the order in which they](http://www.blogger.com/blogger.g?blogID=1341810718493854175)

[are](http://www.blogger.com/blogger.g?blogID=1341810718493854175) [entered. The index should be read into memory when the data file is opened.](http://www.blogger.com/blogger.g?blogID=1341810718493854175)

* + - [Indexes That are too Large to Hold in Memory](http://www.blogger.com/blogger.g?blogID=1341810718493854175)
    - [Searching of a simple index on disk takes too much time.](http://www.blogger.com/blogger.g?blogID=1341810718493854175)
    - [Maintaining a simple index on disk in sorted order takes too much time.](http://www.blogger.com/blogger.g?blogID=1341810718493854175)
    - [Tree structured indexes such as B-trees are a scalable alternative to simple indexes.](http://www.blogger.com/blogger.g?blogID=1341810718493854175) [Hashed organization is an alternative to indexing when only a primary index is needed.](http://www.blogger.com/blogger.g?blogID=1341810718493854175)

### File Handling:

Many real-life problems handle large volume of data and in such situations, we need to use some devices such as Floppy disk or hard disk to store the data. The data is stored in these devices using the concept of files. A file is a collection of related data stored in a particular area on the disk. Programs can be designed to perform the read and write operations on these files.

Class for file stream operation:

The I/O system of CPP contain a set of classes that define the file handling methods. This includes ifstream ofstream and fstream. These classes are derived from fstream base and the corresponding iostream class, the classes, designed manage the disk, are declared in fstream and therefore we must include this file in any program that uses file.

Mode for opening the file:

1) ios::app ->To append at the end of file 2)ios::out->its open file in write only mode.

3) ios::in->its open file in read only mode. Read file syntax:

Stream —object.read((char\*)&class-object, sizeof(class-object));

Files syntax:

Stream -object.write((char\*)&class-object,sizeof(class -object)); Of.close():- It closes the stream. All buffers associated with the stream are flushed before closing. System allocates buffers are freed upon closing. Buffers are assigned with set buf are not automatically freed.

## IMPLEMENTATION & METHODLOGY

**Implementation Methodology:**

Model View Controller or MVC as it is popularly called, is a software design pattern for developing web applications. A Model View Controller pattern is made up of the following three parts:

* + - **Model** - The lowest level of the pattern which is responsible for maintaining data.

* + - **View** - This is responsible for displaying all or a portion of the data to the user.

* + - **Controller** - Software Code that controls the interactions between the Model and View.

MVC is popular as it isolates the application logic from the user interface layer and supports separationof concerns. Here the Controller receives all requests for the application and then works with the Modelto prepare any data needed by the View. The View then uses the data prepared by the Controller to generate a final presentable response. The MVC abstraction can be graphically represented as follows.

# CHAPTER 6

**SNAPSHOTS**

## HOME PAGE

|  |
| --- |
|  |
|  |  |

Figure 6.1: Home Page

.

## PATIENT DATA PAGE

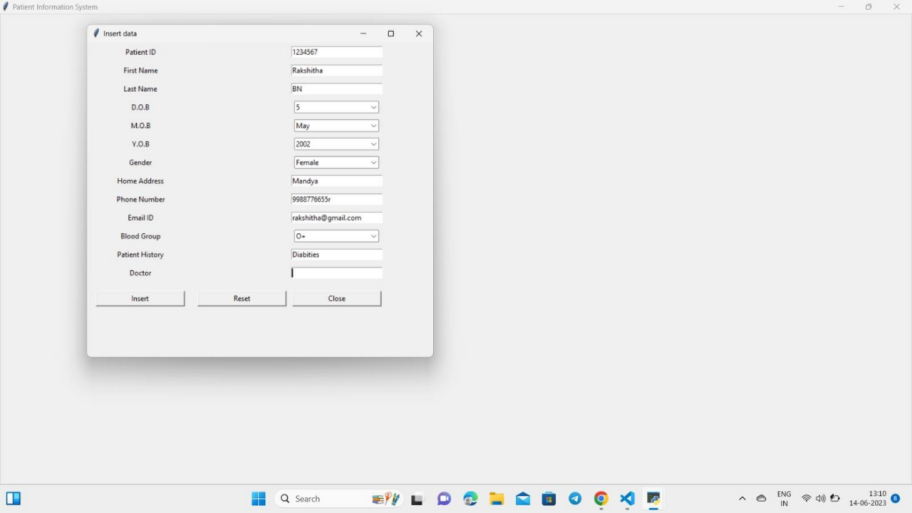


Figure 6.2: Patient Data Page

* 1. **RESET PAGE**

Figure 6.2: Patient Data page

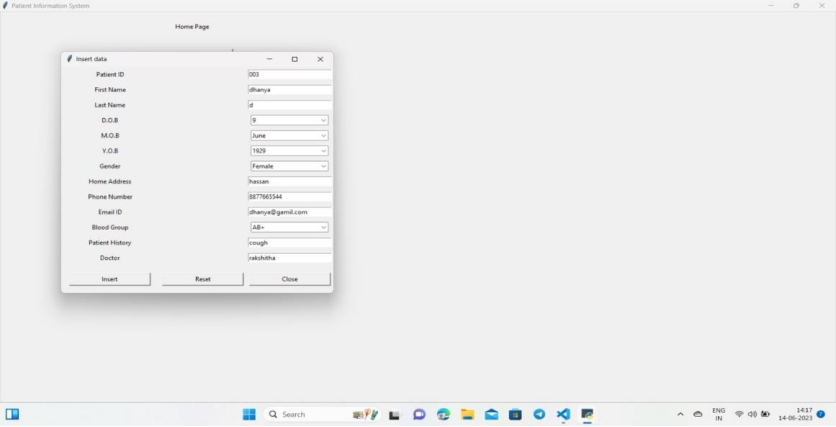


Figure 6.3: Reset page

## UDPADTE PAGE

|  |
| --- |
|  |
|  |  |

Figure 6.4 Update Page

* 1. **UPDATE DATA PAGE**

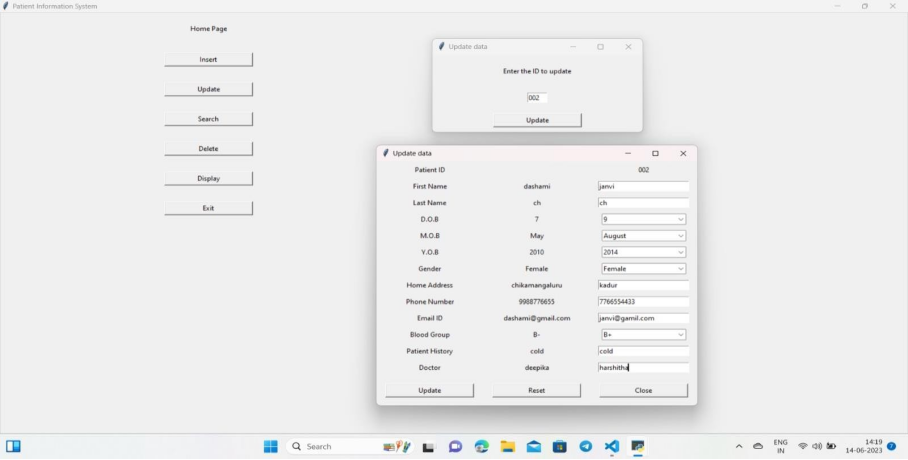


Figure 6.5 Update Data Page

## SEARCH DATA

|  |
| --- |
|  |
|  |  |

Figure 6.6 Search Data Page

## VIEW WINDOW

|  |
| --- |
|  |
|  |  |

Figure 6.7 View Window Page

## DELETE DATA

|  |
| --- |
|  |
|  |  |

Figure 6.8 Delete Data Page

## DISPLAY DATA

|  |
| --- |
|  |
|  |  |

Figure 6.9 Display Data Page

# CHAPTER6

# CONCLUSION

In conclusion, Patient Information System play a vital role in accurately recording, managing, and securing information . These systems provide organizations withthe ability to maintain an auditable and transparent record of their activities . Without question, patient-powered patient registries and networks are a rapidly evolving contributor to research, and particularly to research that focuses on direct improvements in practice. These entities blur traditional boundaries, breaking the barrier of patient, family, and advocate involvement and control in research, translation, and dissemination. A clear movement has emerged to connect individual patient organizations and single-condition patient registries into broader networks that unify, standardize, and optimize the data collection and research generation process. Yet, incentives remain for organizations that wish to pursue their own agendas and create independent single-condition patient registries for their constituencies, which may or may not be possible when joining a PPRN.

A number of factors in the current health care environment may influence further evolution of patient- powered research, including the increased attention and interest in patient-centered research evidence, the expanding role of patients in institutional research efforts, and new technological advancements that enable new forms of patient outreach and engagement. It is likely that the options and choices for patient advocacy and support organizations wishing to contribute to research will expand. Yet, it is also likely that best practices and standards will emerge that guide both researcher-powered and patient-powered registries and research networks, providing additional knowledge to organizations seeking the most efficient and effective approaches to generating knowledge about conditions and treatments.

# REFERENCES

[1] MichaelJ. Folk, Bill Zoellick, Greg Ricchardi: File Structures -An Object-Oriented

Approach with C++

[2] https://www.google.com

[3] https://www.youtube.com

[4] https://github.com

[5] https://www.geeksforgeeks.org/file-handling-c-classes/?ref=lbp

[6] https://www.python.com

[7] "Python GUI Programming with Tkinter" by Alan D. Moore

[8] "Python Crash Course" by Eric Matthes.