

##### Jaypee University of Information Technology

**PROJECT-REPORT**

Course Code: 20B1WCI571

Course Name: DATA SCIENCE AND VISUALIZATION LAB

Submitted by: Submitted to:

Name: Aganay Sambyal & Vaibhav Gupta Name: Dr. Diksha Hooda

Roll no: 211175 & 211183

Batch: CS-53

# ACKNOWLEDGEMENT

We would like to express our sincere gratitude towards all the people who have supported us during this project. Without their valuable contributions, this project would not have been possible. In addition to this, We take this occasion to thank God, almighty for blessing us with his grace and taking our endeavor to a successful culmination. We extend our sincere and heartfelt thanks to our esteemed guide, **DR. DIKSHA HOODA**, for providing us with the right guidance and advice at crucial junctures and for showing me the right way. Furthermore, We would like to express Our gratitude towards my family and friends for their unwavering support and encouragement. Their constant motivation and support have helped Us to stay focused and determined throughout the project. Finally, We would like to thank all the people who have contributed to this project in any way. Your contributions have been invaluable and greatly appreciated.

Thank you all for your support and guidance throughout this project.

## TABLE OF CONTENTS

[ABSTRACT](#_heading=h.gjdgxs)

1. **INTRODUCTION** 
   1. [PROJECT AIMS AND OBJECTIVES](#_heading=h.30j0zll)
   2. [BACKGROUND OF PROJECT](#_heading=h.1fob9te)
2. **SYSTEM ANALYSIS** 
   1. [SOFTWARE REQUIREMENT SPECIFICATION](#_heading=h.3znysh7)
   2. SOFTWARE TOOL USED
3. **SYSTEM DESIGN** 
   1. PROJECT CODE
4. **CONCLUSION & FUTURE SCOPE**

## ABSTRACT:

The PulseHub project presents a comprehensive data science solution for managing patient records efficiently. The system, implemented in Python, leverages CSV file handling and visualization libraries to provide a user-friendly interface for healthcare professionals. The Patient Records System enables the addition, retrieval, update, and deletion of patient information. Additionally, it offers functionalities such as searching for patients within a specified age range and visualizing age and health problem distributions.

The core features include data loading and saving, patient manipulation methods, and insightful data visualizations. The system utilizes CSV files to store patient records, ensuring data consistency. Interactive user input allows seamless interaction with the system, enabling healthcare practitioners to manage patient data effortlessly. The code employs exception handling to address file-not-found scenarios, ensuring the robustness of the application.

The data visualization aspect of PulseHub adds value to healthcare management by providing graphical representations of age distribution and health problem prevalence among patients. The age distribution histogram offers a quick overview of the patient demographic, while the health problem distribution pie chart provides insights into prevalent health issues.

PulseHub serves as a versatile tool for healthcare administrators and practitioners, offering a unified platform for managing patient records and gaining insights from the data. The system's modular structure and interactive design make it adaptable for integration with larger healthcare management systems, contributing to the advancement of patient care and data-driven decision-making in the medical field.

**CHAPTER 1 INTRODUCTION**

This chapter gives an overview of the aim, objectives, background and operation environment of the system.

* 1. **PROJECT AIMS AND OBJECTIVES**

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter. The aims and objectives are as follows:

* + - Develop a robust Patient Records System for efficient healthcare data management.
    - Ensure data consistency through CSV file handling.
    - Create an interactive interface for seamless addition, retrieval, and updating of patient records.
    - Implement data visualization for insights into patient demographics and prevalent health issues.
    - Design the system for adaptability, capable of standalone use or integration into larger healthcare systems.

##### BACKGROUND OF PROJECT

In the evolving landscape of healthcare, the PulseHub project emerges as a response to the pressing need for a more efficient and streamlined patient record management system. Traditional methods of record-keeping within healthcare settings often prove cumbersome and error-prone, underscoring the necessity for a digital solution. PulseHub draws inspiration from the increasing importance of data-driven decision-making in healthcare, seeking to empower healthcare professionals with a Python-based system that incorporates data science principles. This system is designed to enhance the management of patient records, offering not only a reliable storage and retrieval mechanism but also a means of extracting valuable insights through data visualization. By leveraging these principles, PulseHub aims to contribute to the ongoing improvement of data management practices in the healthcare sector.

## PROJECT REQUIREMENTS:

* Use CSV file handling for reliable data storage and retrieval.
* Develop an intuitive user interface for healthcare practitioners.
* Integrate matplotlib for visualizations of age distribution and health problem prevalence.
* Implement robust exception handling, especially for file-not-found scenarios.
* Design the system with modularity for potential integration into larger healthcare management systems.

**CHAPTER 2 SYSTEM ANALYSIS**

The PulseHub system is meticulously designed to optimize healthcare data management. Through a robust data file handling mechanism, it ensures the consistency of patient records using CSV files. The interactive user interface caters to the specific needs of healthcare practitioners, allowing seamless addition, retrieval, and updating of patient information. The integration of matplotlib facilitates insightful data visualizations, offering a quick overview of patient demographics and prevalent health issues. Exception handling, notably addressing file-not-found scenarios, ensures the reliability of the system. The modular design enhances adaptability, making PulseHub suitable for standalone use or potential integration into larger healthcare management systems.

#### SOFTWARE REQUIREMENT SPECIFICATION

* + 1. **GENERAL DESCRIPTION**

**PRODUCT DESCRIPTION:**

PulseHub is a Python-based Patient Records System designed for efficient healthcare data management. It ensures data consistency through CSV file handling, features an intuitive user interface for healthcare practitioners, and allows seamless addition, retrieval, updating, and deletion of patient records. The system incorporates matplotlib for insightful data visualizations, and its robust exception handling ensures reliability. With a modular design, PulseHub is adaptable for standalone use or integration into larger healthcare management systems.

##### PROBLEM STATEMENT:

The problem occurred before having computerized system includes:

##### Manual Record-Keeping Challenges:

##### Time-Consuming: Manual entry, retrieval, and update of patient records are labor-intensive and time-consuming processes, potentially leading to inefficiencies.

##### Error-Prone: The manual nature of record-keeping increases the likelihood of errors in data entry, jeopardizing the accuracy and integrity of patient information.

##### Limited Accessibility:

##### Restricted Information Access: Healthcare practitioners face limitations in accessing patient data promptly, hindering the swift delivery of care and decision-making.

##### Data Inconsistency:

##### Lack of Standardization: Without a computerized system, maintaining a standardized format for patient records is challenging, leading to inconsistencies in data representation.

##### Inefficient Reporting:

##### Cumbersome Reporting Processes: Generating comprehensive reports on patient demographics, age distribution, and health issues without a computerized system is a time-consuming and challenging task.

##### Security Concerns:

##### Vulnerability to Data Breaches: Manual record-keeping systems may lack the security measures required to safeguard sensitive patient information, posing a risk of data breaches.

#### SYSTEM REQUIREMENTS

* + - 1. **NON-FUNCTIONAL REQUIREMENTS**

#### Product Requirements

#### Performance and Reliability:

#### Ensure high performance with swift response times and minimal downtime, supported by robust exception handling.

#### Security and Usability:

#### Prioritize data security through encryption and access controls while maintaining an intuitive and user-friendly interface for healthcare practitioners.

#### Scalability and Compatibility:

#### Design the system for scalability to efficiently manage growing data volumes and ensure compatibility with common operating systems.

#### Documentation and Maintainability:

#### Provide comprehensive documentation for system usage, maintenance, and troubleshooting, with a design that facilitates easy updates and improvements.

* + - 1. **FUNCTIONAL REQUIREMENTS**

**Data Management:**

Enable the addition, retrieval, updating, and deletion of patient records.

**Data Visualization:**

Implement visualizations using matplotlib for age distribution and prevalent health issues among patients.

**Search Functionality:**

Allow healthcare practitioners to search for patients by ID and within specified age ranges.

**User Authentication:**

Implement user authentication mechanisms to ensure secure access to patient records.

#### SOFTWARE AND HARDWARE REQUIREMENTS

**Software Requirements:**

1. **Operating System:**Compatible with Windows, macOS, and Linux operating systems.
2. **Python:**Requires Python 3.x for the execution of the PulseHub system.
3. **Libraries:**Dependencies include matplotlib for data visualization.
4. **Text Editor or Integrated Development Environment (IDE):**Recommend using a text editor (e.g., Visual Studio Code) or IDE (e.g., PyCharm) for code development.

**Hardware Requirements:**

1. **Processor**:  
   Minimum dual-core processor for optimal performance.
2. **Memory (RAM):**Recommended 4 GB or higher for efficient data handling.
3. **Storage:**Adequate storage space for patient records and system files.
4. **Display:**Monitor with a minimum resolution of 1280x800 pixels for optimal interface display.
5. **Input Devices:**Keyboard and mouse or equivalent input devices for user interaction.
6. **Network Connection:**Internet connectivity for library installations and potential system updates.

These requirements ensure the smooth functioning of the PulseHub system, offering compatibility with common operating systems and sufficient resources for efficient performance.

### SOFTWARE TOOLS USED

For the development of PulseHub, the following software tools are employed:

1. **Visual Studio Code (VS Code):** Utilized as the primary code editor for efficient development and debugging.
2. **Python:** The system is developed using the Python programming language, leveraging its versatility and readability.
3. **Matplotlib Library:** Incorporated for data visualization, aiding in the creation of insightful charts and graphs.

These tools collectively contribute to the creation, testing, and optimization of the PulseHub Patient Records System.

## CHAPTER 3 SYSTEM DESIGN

**CODE**

**FUNCTIONS**

def add\_patient(self, patient\_id, name, address, phone\_number, age, health\_problem):

# Convert PatientID to string to ensure consistency

patient\_id = str(patient\_id)

if patient\_id not in self.records:

self.records[patient\_id] = {

"PatientID": patient\_id,

"Name": name,

"Address": address,

"Phone Number": phone\_number,

"Age": age,

"Health Problem": health\_problem

}

self.save\_data()

print(f"Patient {name} added successfully.")

else:

print(f"Patient with ID {patient\_id} already exists.")

def get\_patient\_info(self, patient\_id):

# Convert PatientID to string to ensure consistency

patient\_id = str(patient\_id)

return self.records.get(patient\_id)

def display\_all\_patients(self):

print("\nPatient Records:")

for info in self.records.values():

print(f"ID: {info.get('PatientID', 'N/A')}, Name: {info.get('Name', 'N/A')}, Address: {info.get('Address', 'N/A')}, Phone Number: {info.get('Phone Number', 'N/A')}, Age: {info.get('Age', 'N/A')}, Health Problem: {info.get('Health Problem', 'N/A')}")

def update\_patient\_info(self, patient\_id, name, address, phone\_number, age, health\_problem):

# Convert PatientID to string to ensure consistency

patient\_id = str(patient\_id)

if patient\_id in self.records:

self.records[patient\_id] = {

"PatientID": patient\_id,

"Name": name,

"Address": address,

"Phone Number": phone\_number,

"Age": age,

"Health Problem": health\_problem

}

self.save\_data()

print(f"Patient {name} information updated successfully.")

else:

print(f"Patient with ID {patient\_id} not found.")

def delete\_patient(self, patient\_id):

# Convert PatientID to string to ensure consistency

patient\_id = str(patient\_id)

if patient\_id in self.records:

del self.records[patient\_id]

self.save\_data()

print(f"Patient with ID {patient\_id} deleted successfully.")

else:

print(f"Patient with ID {patient\_id} not found.")

def search\_by\_age\_range(self, min\_age, max\_age):

results = [info for info in self.records.values() if min\_age <= int(info.get('Age', 0)) <= max\_age]

return results

def plot\_age\_distribution(self):

ages = [int(info.get('Age', 0)) for info in self.records.values() if info.get('Age').isdigit()]

plt.hist(ages, bins=range(min(ages), max(ages) + 5, 5), edgecolor='black')

plt.title('Age Distribution of Patients')

plt.xlabel('Age')

plt.ylabel('Number of Patients')

plt.show()

def plot\_health\_problem\_distribution(self):

health\_problems = [info.get('Health Problem', 'N/A') for info in self.records.values()]

health\_problem\_counts = {problem: health\_problems.count(problem) for problem in set(health\_problems) if problem != 'N/A'}

labels = list(health\_problem\_counts.keys())

counts = list(health\_problem\_counts.values())

plt.pie(counts, labels=labels, autopct='%1.1f%%', startangle=140)

plt.title('Health Problems Distribution of Patients')

plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()

# CHAPTER 5

**CONCLUSION & FUTURE SCOPE**

In conclusion, the PulseHub Patient Records System presents a viable solution to the challenges associated with manual patient data management in healthcare. The system, developed using Python and supported by libraries such as Matplotlib, offers an intuitive user interface for healthcare practitioners. Through rigorous testing, including functionality, performance, security, and usability assessments, PulseHub demonstrates its reliability and efficiency in streamlining data processes and providing insightful visualizations. The system's modular design allows for adaptability, making it suitable for standalone use or integration into larger healthcare management frameworks.

**4. Future Scope:**

The PulseHub project lays the foundation for future enhancements and integrations in healthcare data management. The following areas offer potential avenues for further development:

1. **Machine Learning Integration:**
   * Implement machine learning algorithms for predictive analytics, enabling the system to anticipate and address healthcare trends.
2. **Mobile Application:**
   * Develop a mobile application to extend accessibility, allowing healthcare practitioners to interact with patient records on the go.
3. **Enhanced Security Measures:**
   * Integrate advanced security protocols to safeguard patient data against evolving cyber threats and ensure compliance with healthcare data protection regulations.