CPP104: Linked List Operations Activity: Patient Management System

System Overview:

The primary purpose of the patient management system is to streamline the administrative and medical operations within a healthcare facility. This system is designed to efficiently handle patient information, track medical histories, manage appointments, and ensure that healthcare providers have immediate access to crucial patient data. By automating and organizing patient records, the system aims to enhance the quality of care, reduce administrative burdens, and improve the overall patient experience.

The functionality of the system revolves around maintaining comprehensive and up-to-date records of all patients. This includes personal information, medical history, appointment schedules, treatment plans, and billing information. The system ensures that all data is stored securely and can be accessed and updated in real-time, providing healthcare providers with the most current information. Additionally, the system is designed to be user-friendly, allowing medical staff to navigate and utilize its features with ease.

The patient management system offers an intuitive interface for users to add new patient records, update treatment details, schedule appointments, and record new medical information. It also allows quick retrieval of records, search functionality for quick access, and updates to existing records to ensure accurate and current patient data, facilitating informed medical decisions.

The patient management system is a unique system that synchronizes patient data in real-time, protects sensitive information with robust security protocols, and is designed for ease of use. It features linked lists for easy scaling and powerful reporting tools for data-driven decision-making, improving healthcare outcomes and ensuring consistent performance despite large data volumes.

To sum up, the patient management system is an all-inclusive instrument intended to improve the efficacy and efficiency of healthcare management. Healthcare professionals can experience a seamless workflow thanks to the system's extensive features and user-friendly interface, which guarantees flexible and efficient management of patient records through linked list operations.

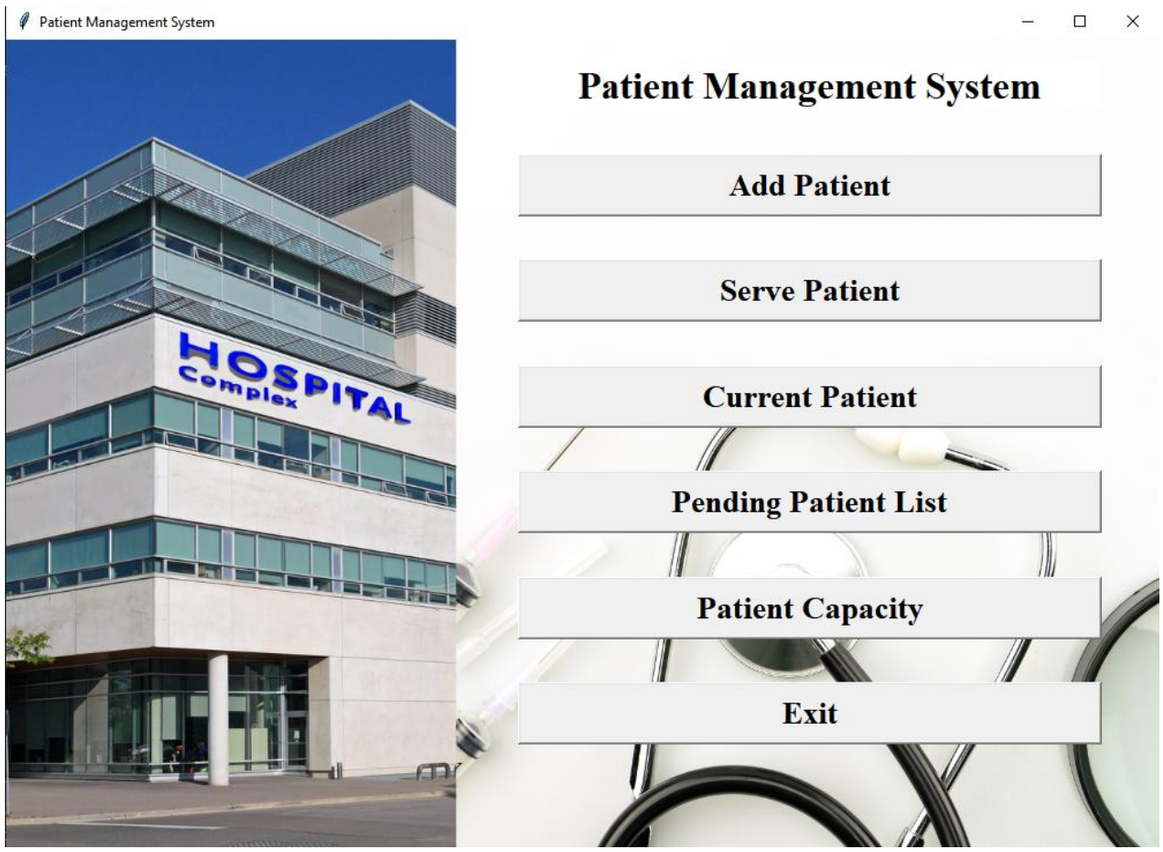
Order Management Operations:

- **Add Patient:** Adds a new patient to the linked list.
- **Serve Patient:** Removes the first patient from the linked list and marks them as served.
- **Current Patient:** Displays the patient currently being served.
- **Pending Patient List:** Displays all patients in the linked list.
- **Patient Capacity:** Checks if the list has reached its maximum capacity. The maximum number of patient for this program is 10 patients.

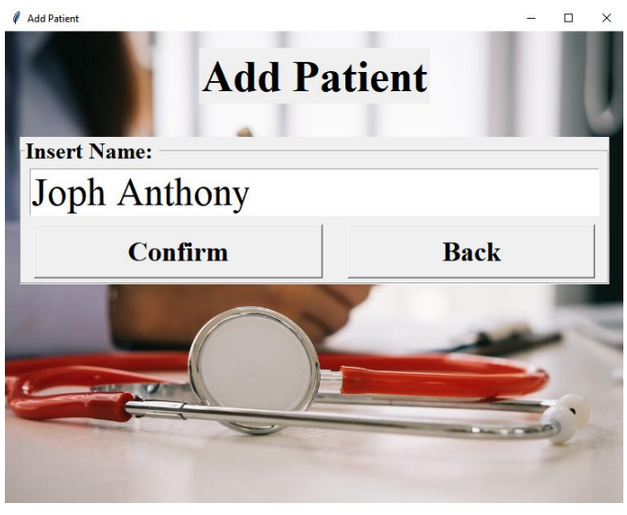
User Manual:

Step by Step Guide

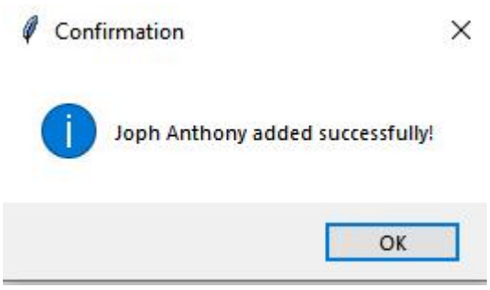
Upon execution of the program, an interface will appear where the user can choose an option by pressing the buttons.



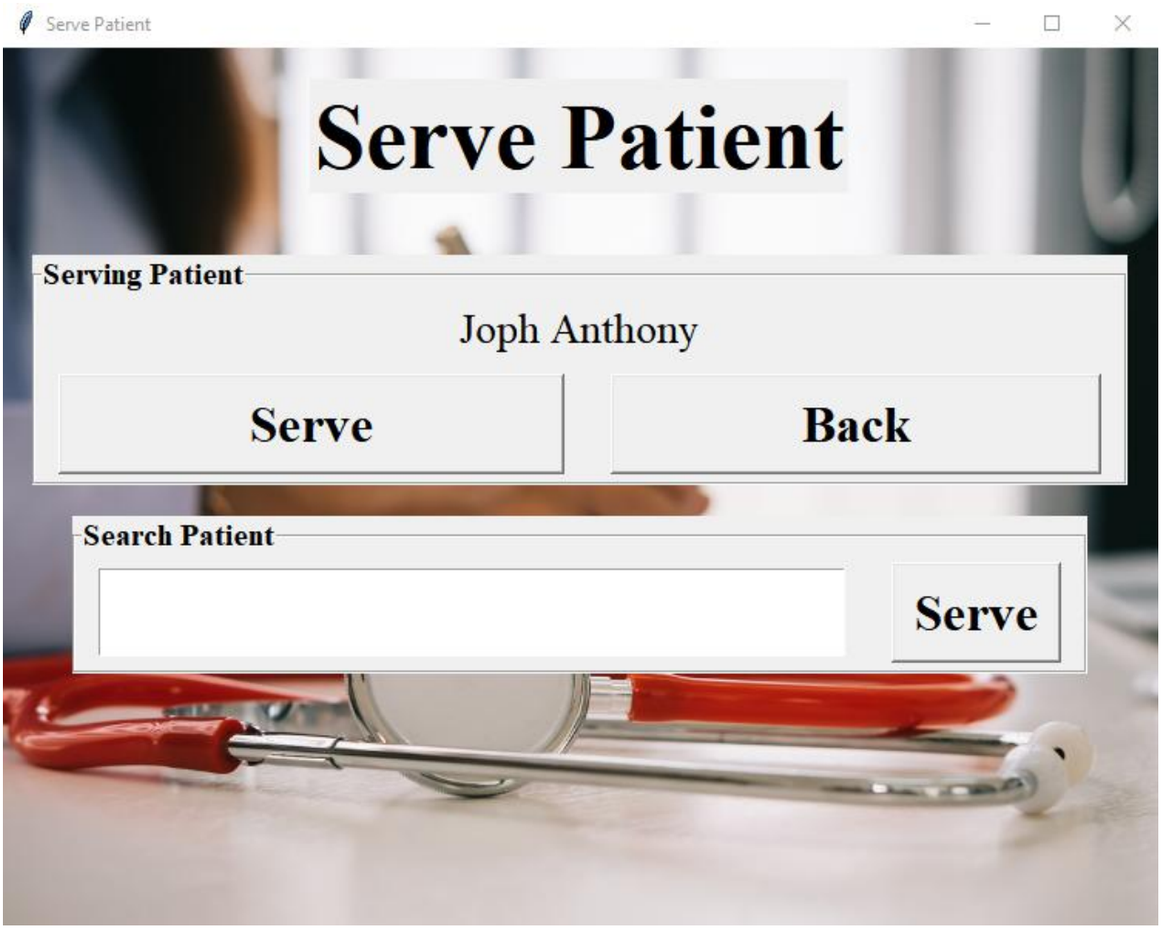
If "Add Patient" is pressed, a new window will appear where the user can input the patient's name.



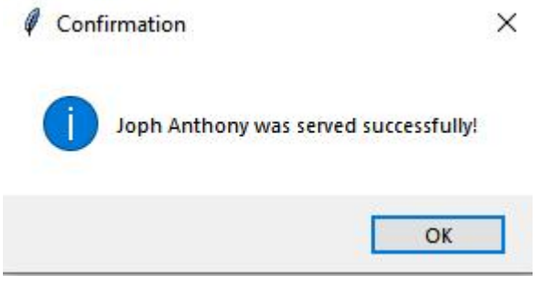
Pressing the "Confirm" button will provide a message box indicating that the patient's name has been added.



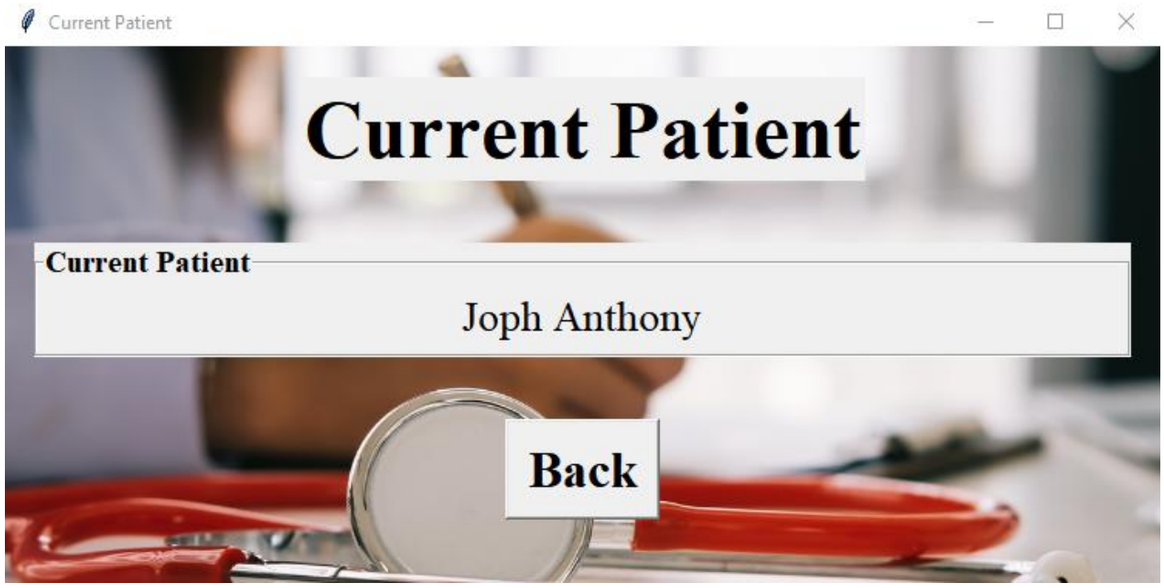
If "Serve Patient" is pressed, a new window will appear with the patient's name that is added first.



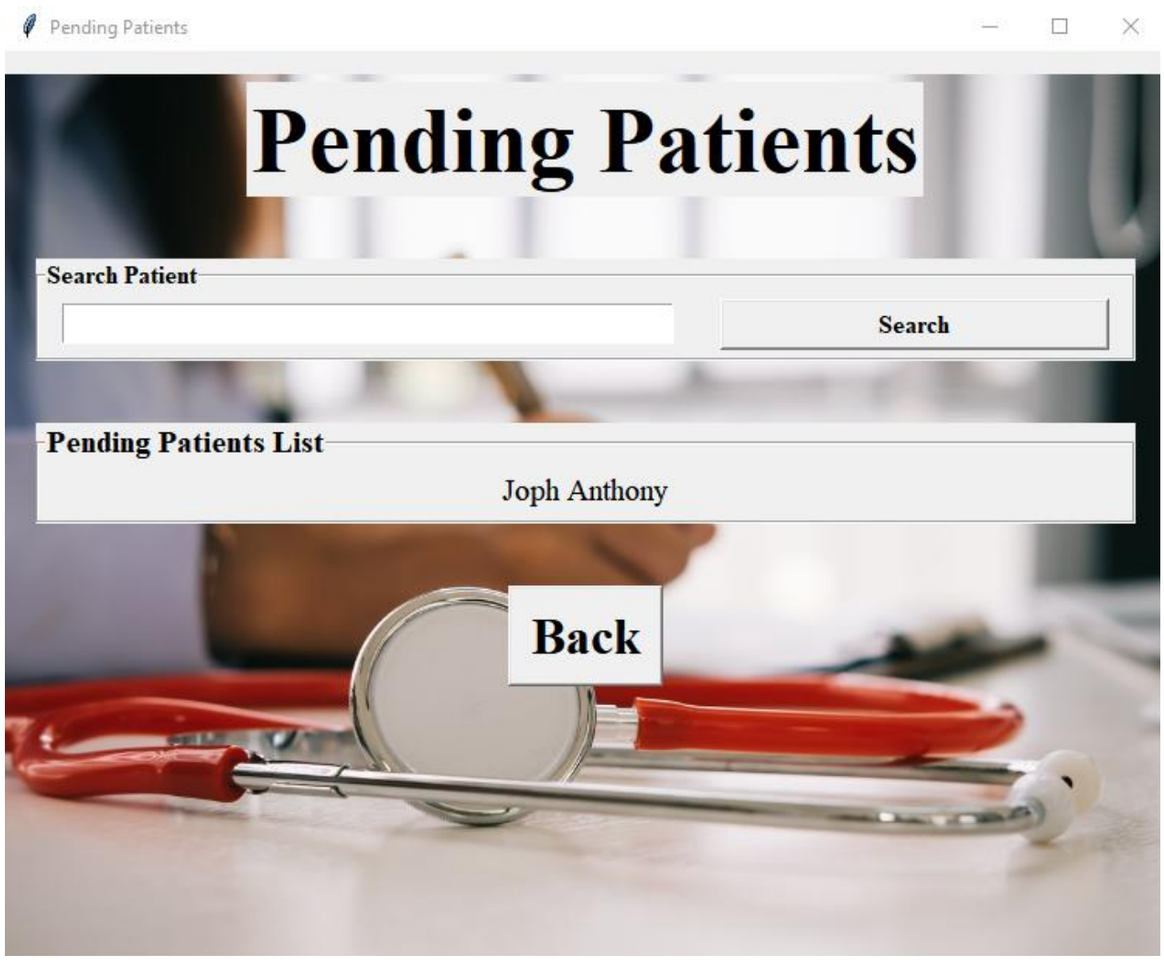
Pressing the "Serve Patient" button will provide a message box indicating that the patient has been served whereas it removes the patient's name from the list.



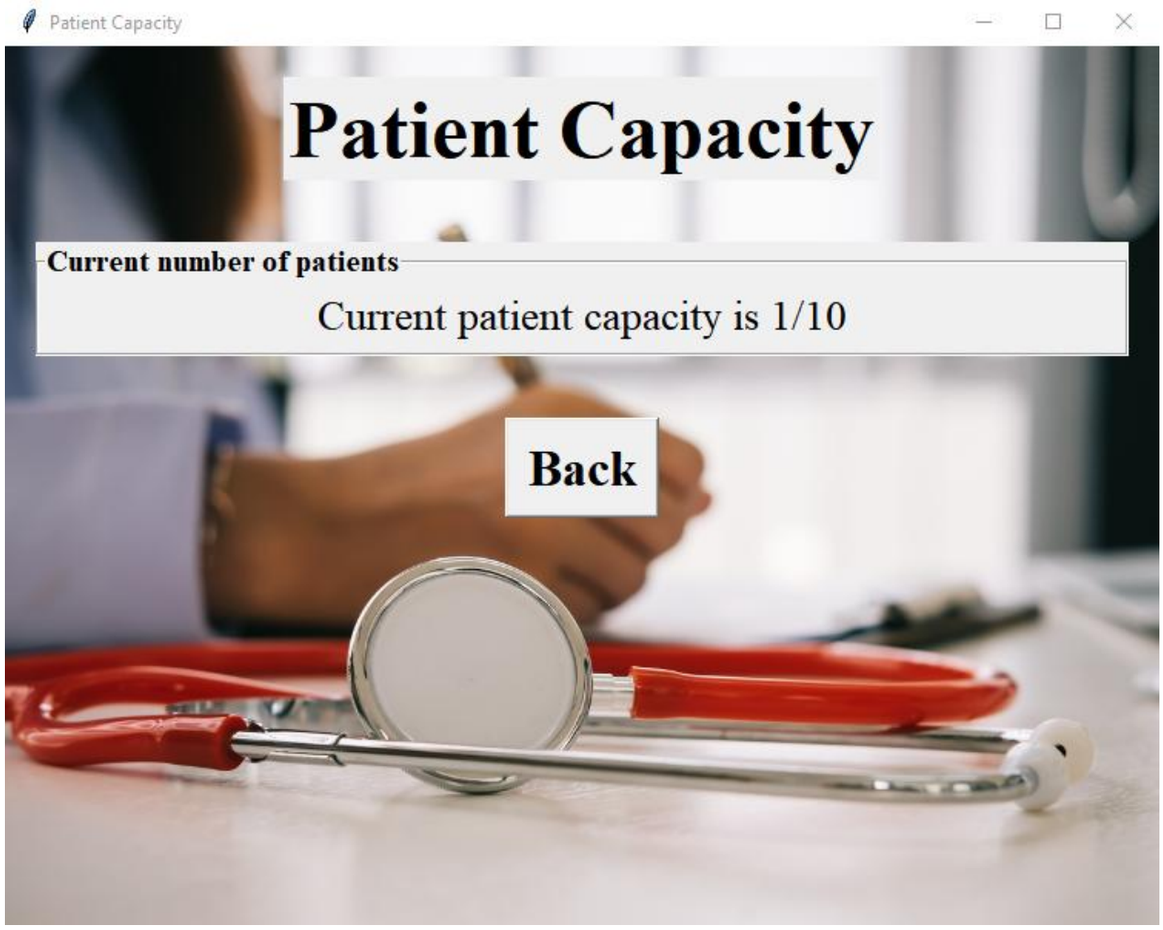
If “Check Current Patient” is pressed, a new window will appear displaying all patients name in the list



If “Check Pending Patient” is pressed, a new window will appear displaying any patients left to serve.



If “Check Patient List is Full” is pressed, a new window will appear displaying whether the list is full or not.



Pressing the “EXIT” button will close the main window.

Python Libraries

This system utilizes the Tkinter GUI library, which allows students to create a visual user interface that functions as a software application. This application empowers users to perform various tasks such as adding a patient, serving a patient, checking the current patient, and more. By leveraging Tkinter, the user gains full control over the entire system, enhancing its accessibility and user-friendliness.

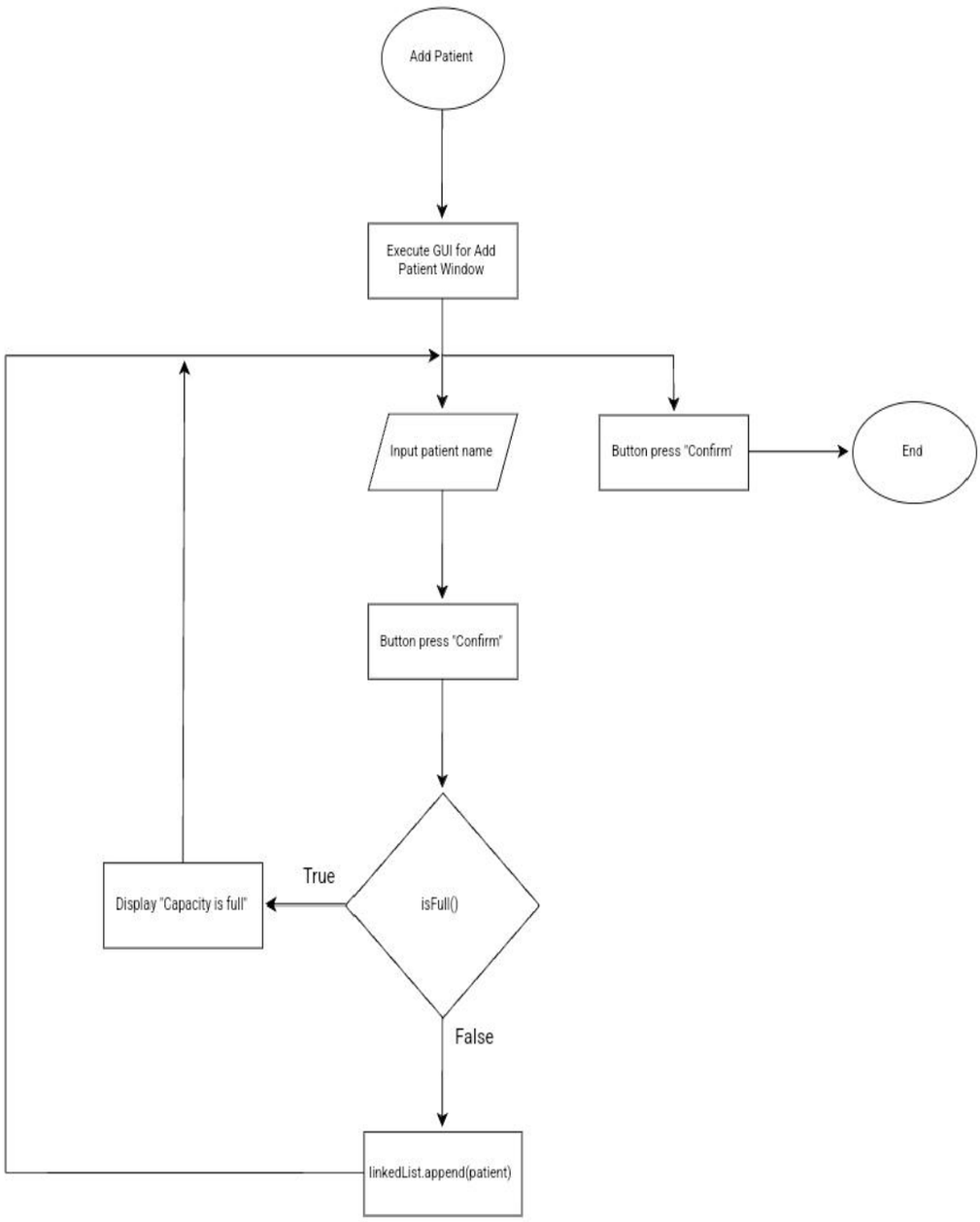
The main functionalities of the LinkedList data structure are encapsulated within a dedicated class. This class begins by initializing an empty array and a variable with a specific value. Several critical functions are included to manage the LinkedList efficiently:

1. **is_empty**: This function checks if the list contains any elements. It returns True if the list is empty and False otherwise, ensuring that the application can handle cases where operations are attempted on an empty list.
2. **append**: This function adds an element to the end of the list. It allows for dynamic growth of the list by enabling the addition of new patient records or data entries at the end of the list.
3. **delete_node_at_position**: This function removes an element from the list. It ensures that the application can maintain an accurate and up-to-date list by allowing the deletion of specific entries as needed.
4. **length**: This function determines the number of elements in the list. It provides a way to gauge the size of the list, which can be crucial for managing system resources and ensuring efficient operations.
5. **Display_patients**: This function prints all the elements in the list. It offers a way to visualize the entire list, which can be helpful for debugging and for users who need to review the current state of the list.

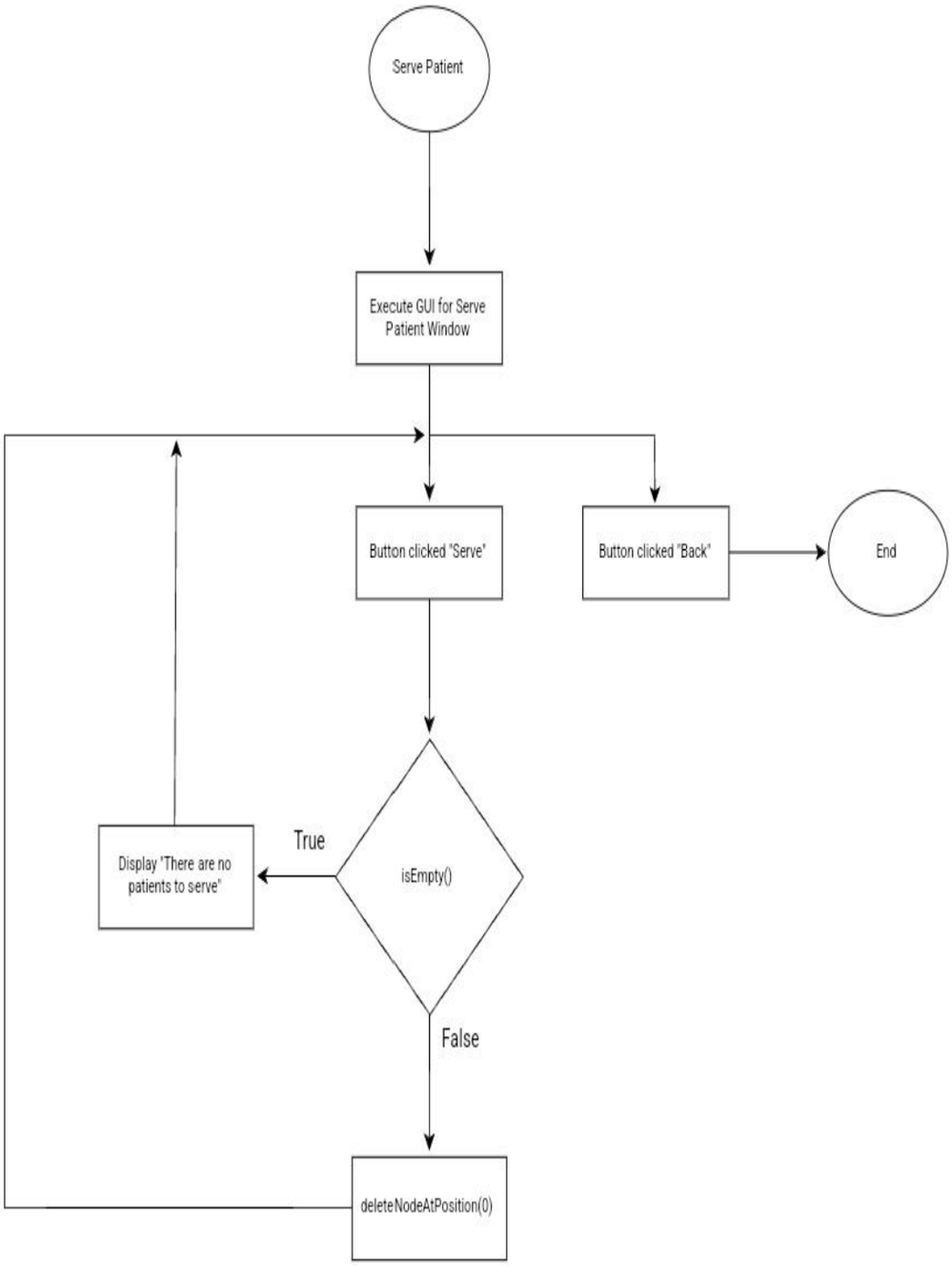
By incorporating these functions into the LinkedList class, the system gains strong and adaptable data management capabilities. Users can seamlessly interact with the list, performing a variety of operations required for effective patient management. The combination of Tkinter's user interface and LinkedList's data handling ensures that the application is both powerful and simple to use, meeting the needs of users who want an intuitive and responsive system.

Furthermore, using Tkinter in conjunction with LinkedList offers students an excellent learning opportunity. It teaches them about the practical application of data structures in real-world scenarios and emphasizes the importance of efficient data management in software development.

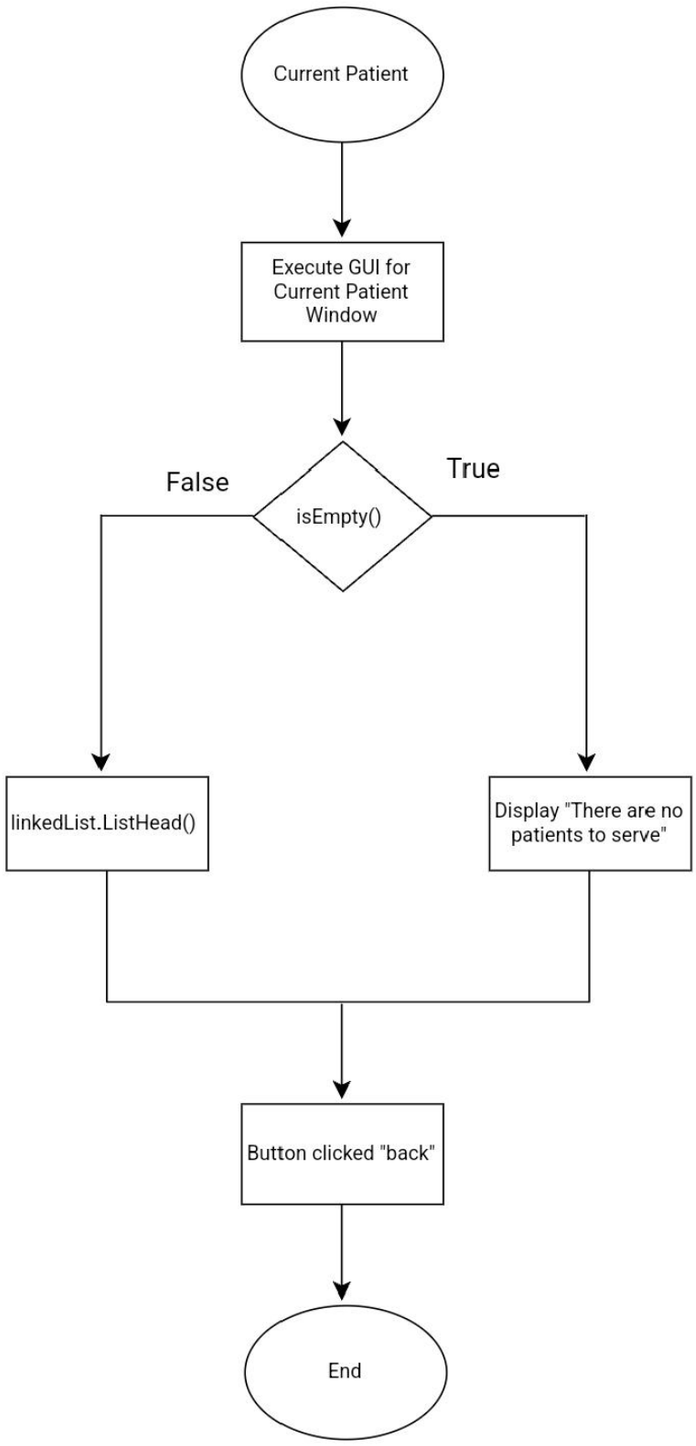
Process Details and Flowcharts



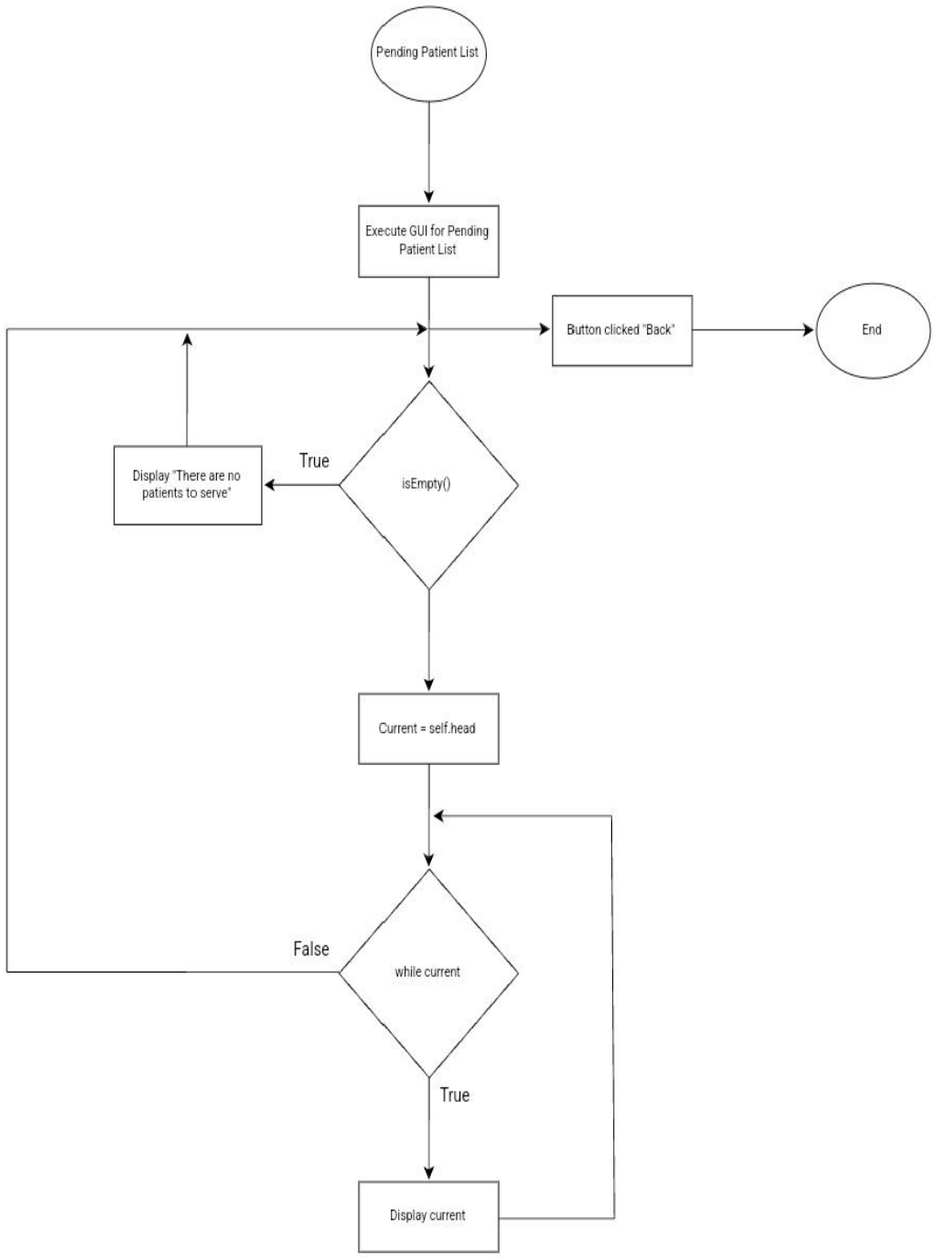
This flowchart shows the process of adding patients in the linked list. The program will ask for the patient's name. When the user hits the confirm button, the program will check if the list if full. If the list is not full, then it will add the patient name in the list. Otherwise, the program will display that the capacity is full.



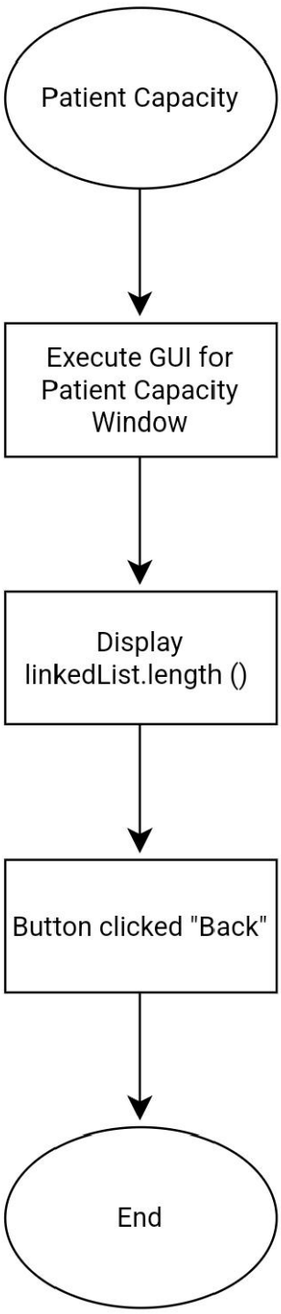
This flowchart shows the process of serving the patients. Upon clicking the serve button, the program will check if the list is empty. If the list is empty, the program will display that there are no patients to serve. Otherwise, it will delete the current head of the linked list, indicating that the patient was served successfully.



This flowchart shows the process of displaying the current patient being served. The program will check if the list is empty. If it is not empty, it will display the head of the list or the patient currently being served. Otherwise, it will display a message that “There are no patients to serve”.



This flowchart shows the process of displaying the list of pending patients. The program will check is the list is empty. If it is not empty, the program will loop through every node and display every patients' name in the list on the window. Otherwise, the program will display a message "There are no patients to serve".



This flowchart shows the the process of displaying the patient capacity. The program displays the number of patients currently admitted in the facility.

Code Snippets

```
class LinkedList:
    def __init__(self):
        self.head = None

    def is_empty(self): #checks if the Linked list is empty
        return self.head is None
```

is_empty() method checks if the linked list is empty.

```
    def append(self, data): #insert at end
        new_node = Node(data)
        if self.is_empty():
            self.head = new_node
            return
        last_node = self.head
        while last_node.next:
            last_node = last_node.next
        last_node.next = new_node
```

Append() method inserts the patient's data at the end of the linked list

```
    def delete_node_at_position(self, position): #deletes a node at a certain
index or position
        if self.head is None:
            return
        temp = self.head
        if position == 0:
            self.head = temp.next
            temp = None
            return
        for _ in range(position - 1):
            temp = temp.next
            if temp is None:
                return
        if temp is None or temp.next is None:
            return
```

```
        next_node = temp.next.next
        temp.next = None
        temp.next = next_node
```

Delete_node_at_position() method deletes a node at a certain position. The position will be indicated at its callout argument.

```
    def length(self): #returns the current lenght of the list
        count = 0
        current = self.head
        while current:
            count += 1
            current = current.next
        return count
```

Length() method returns the number of nodes in the linked list.

```
    def listHead(self): #returns the head of the linked list
        if self.head:
            return self.head.data
        else:
            errorText = "There are no patients to serve yet"
            return errorText
```

listHead() method returns the head node of the linked list.

```
    def isFull(self): #checks if the list is at max capacity (max at 10)
        return self.length() >= 10
```

isFull() method checks if the list is greater than or equal to the max number of patients that the facility can accommodate.

```
    def displayPatients(self): #returns all the data inside the linked list
        current = self.head
        nodes = []
        while current:
            nodes.append(current.data)
            current = current.next
        return nodes
```

displayPatients() method appends every node in a local array then each element is returned.

Technical Documentation:

Linked list operations are crucial for efficient patient record management. These data structures consist of nodes, each containing data and a reference to the next. They handle dynamic data, update without reorganization, and simplify inserting or deleting operations. They allocate memory as needed for each new patient record, optimizing memory usage. Traversing a linked list is simple and efficient, allowing quick access to patient information.

Pseudocode

Main Program

```
Begin
  Function execute_main_window_GUI()
    Execute Main Window GUI

    While True
      If Add Patient button pressed Then
        Call add_patient()
      Else If Serve Patient button pressed Then
        Call serve_patient()
      Else If Current Patient button pressed Then
        Call current_patient()
      Else If Pending Patient List button pressed Then
        Call pending_patient_list()
      Else If Patient Capacity button pressed Then
        Call patient_capacity()
      Else If Exit button pressed Then
        Break
      End If
    End While
End
```

Add Patient

```
Begin
   Function add_patient()
      Execute GUI for Add Patient Window

      Input patient name

      Button press "Confirm"

      If isFull() is True
         Display "Capacity is full"
      Else False
         linkedList.append(patient)
      EndIf
End
```

Serve Patient

```
Begin
   Function serve_patient()
      Execute GUI for Serve Patient Window

      Button clicked "Serve"

      If isEmpty() is True
         Display "There are no patients to serve"
      Else False
         deleteNodeAtPosition(0)
      EndIf

      Button clicked "Back"

End
```

Pending Patient List

```
Begin
   Function pending_patient_list()
      Execute GUI for Pending Patient List

      Button press "Back"
         End

      If isEmpty() is True
         Display "There are no patients to serve"
      Else
         Current = self.head
         While current is not None
            Display current
            Current = Current.next
```

```
        EndWhile
    EndIf
End
```

Patient Capacity

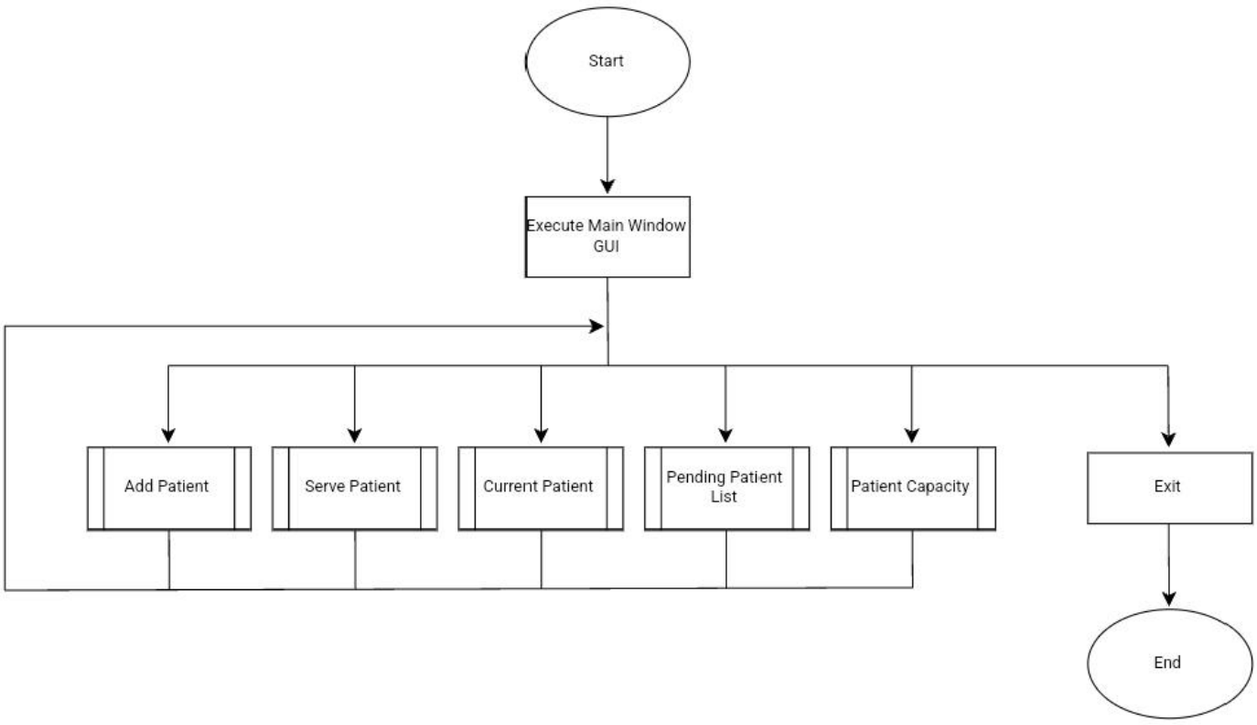
```
Begin
  Function patient_capacity()
    Execute GUI for Patient Capacity Window

    Display linkedList.length()

    Button press "Back"

End
```

Current Patient

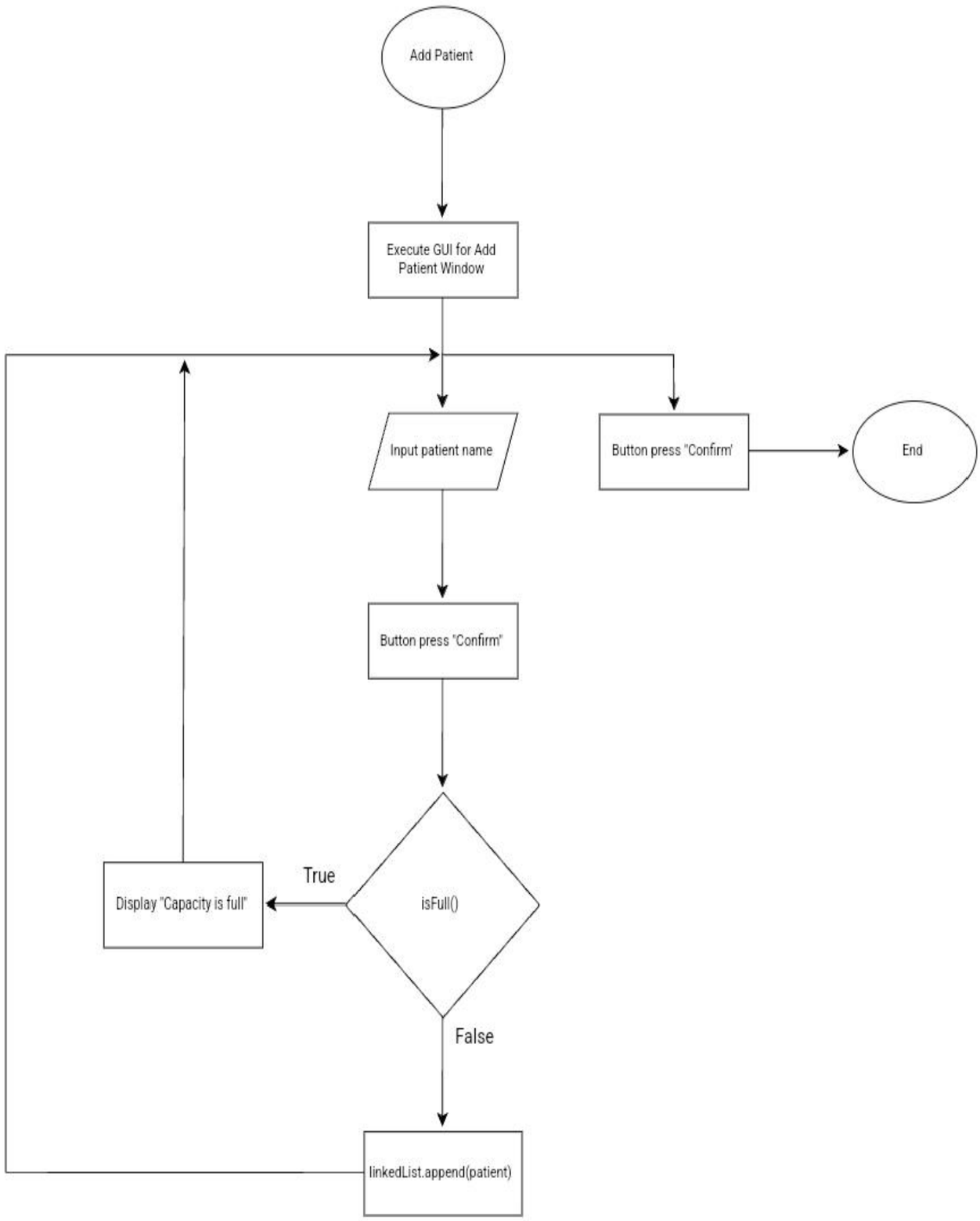
```
Begin
  Function current_patient()
    Execute GUI for Current Patient Window

    If isEmpty() is True
      Display "There are no patients to serve"
    Else
      linkedList.ListHead()
    EndIf

    Wait for button click "back"
End
```

Flowcharts

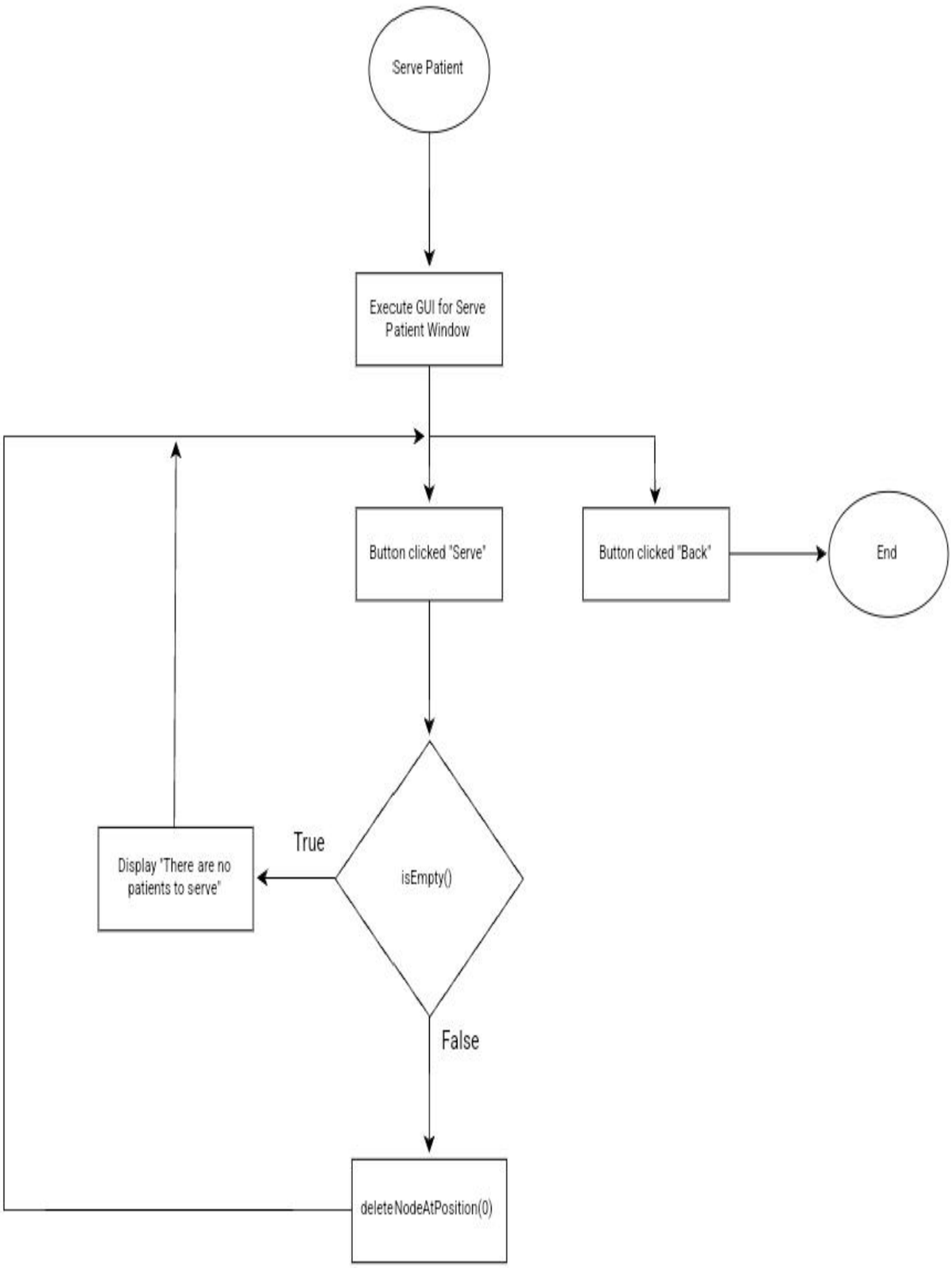
Main Program



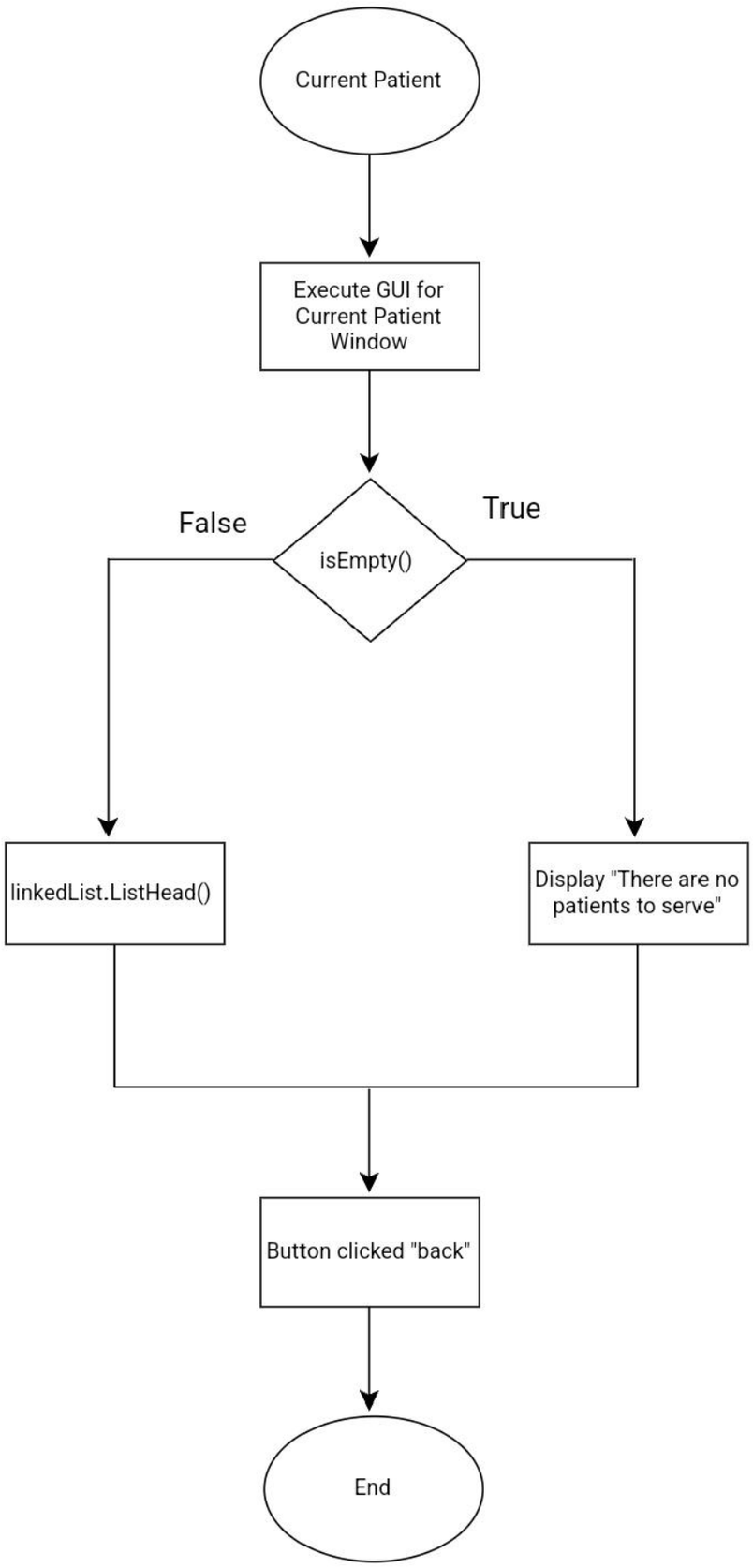
Add Patient


Add Patient
Execute GUI for Add Patient Window
Input patient name
Button press "Confirm"
Button press "Confirm"
End
isFull()
True
Display "Capacity is full"
False
linkedList.append(patient)

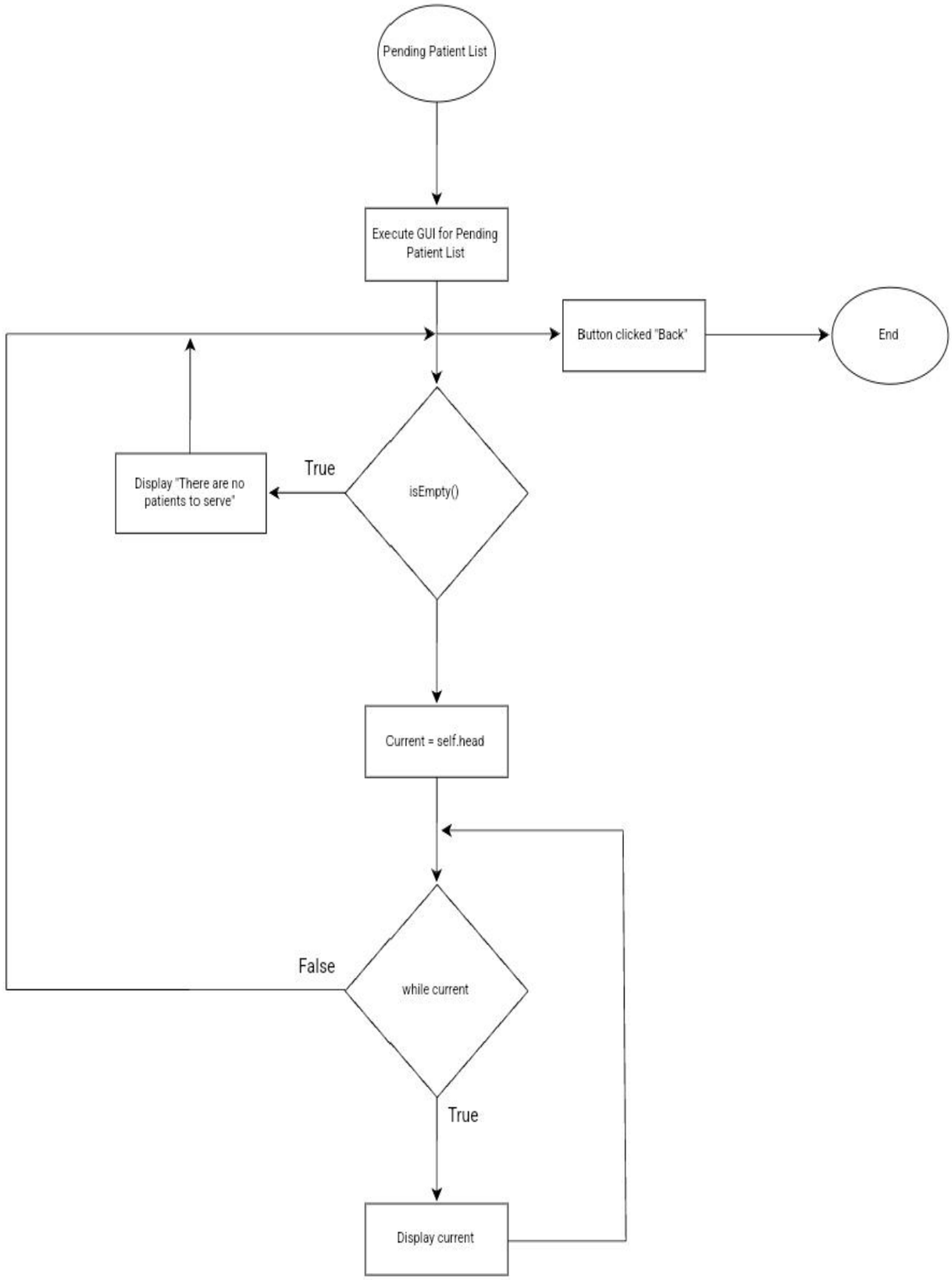
Serve Patient



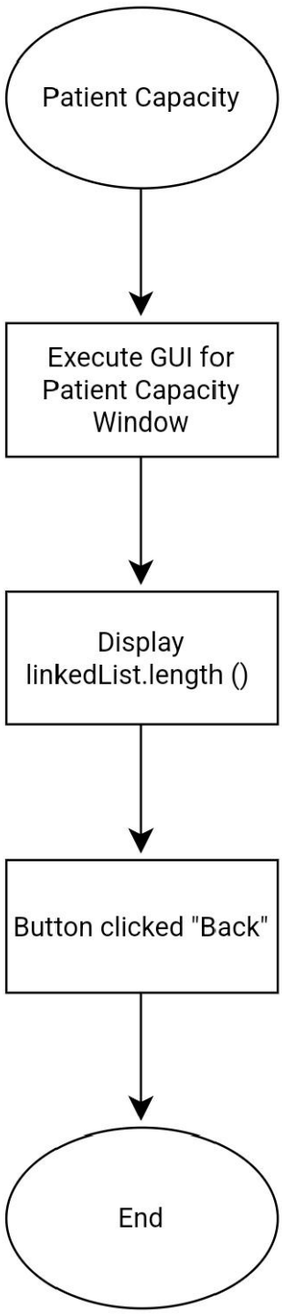
Current Patient



Pending Patient List



Patient Capacity



Known Issues and Limitations:

This program is can only store limited information from the patient. Specifically, the program can only store the name of the patient inside the linked list. The developers recommend to add more entry boxes so that the patient can fill out more information about them so that management can monitor their patients efficiently. The developers also suggest that the program should have filter options that sorts out some information such as the age, sex, illness, etc.

Future Enhancements:

The programmer can improve the user experience of the patient management system in a variety of ways, starting with changing its visual appearance. One of the most basic but effective enhancements is to provide a more appropriate background image for the main window. A carefully selected background image can significantly improve the interface's visual appeal and user engagement. For example, a healthcare-themed background, such as an image of a calm, clean medical environment, can make users feel more at ease and reassured when interacting with the system. This visual enhancement not only makes the system more appealing, but also contributes to the creation of a professional and trustworthy environment, which is essential in a healthcare setting.

In addition to visual improvements, the system's functionality can be greatly improved. One way to accomplish this is to give users more options and tools for better patient data management. For example, adding functions like reverse, prepend, and insert_after_node to fully utilize the LinkedList data structure can significantly improve the system's efficiency and flexibility.

By incorporating these advanced linked list operations, the patient management system can offer more robust data handling capabilities, making it easier for healthcare professionals to manage and access patient information efficiently. This can lead to improved patient care, as healthcare providers can quickly retrieve and update records, ensuring that they have the most current information at their fingertips.

Appendices:

We use draw io to make our flowchart, and we also searched in the internet to find resources that would help us understand the correct way to proceed. We also watched YouTube video to better understand the topic and to gain more techniques especially in Tkinter GUI and images.

Additionally, we use ChatGPT to fix some programs, gather more information, and make corrections. Quillbot is also used to verify that the grammar is correct.

https://app.diagrams.net/

https://www.programiz.com/article/flowchart-programming

https://youtu.be/cTu74xMkolg?si=Ae_RP4Qq90GUPdUr

https://youtu.be/KhN5knbHa7o?si=e0_MMT7aZgsR0E-5

https://chatgpt.com/

https://quillbot.com/

Members:

Name	Contribution
Donasco, Jomer John L.	Back-End Development and Documentation
Inay, Christian B.	Front-End Development and Documentation
Mañosca, Joph Anthony G.	Flowchart, Pseudocode, and Documentation

JOMER JOHN L. DONASCO

Blk 2 Lot 4 San Isidro Heights Phase 2 Banlic, City of Cabuyao, Laguna
pjaydonasco@gmail.com
09152223913

CAREER OBJECTIVES

I am an aspiring successful computer engineering student. I am committed to develop my skills in terms of the hardware and software aspects of computers. I am eager to learn more about the theories and concepts in computer engineering and apply them in the professional field

EXPERIENCES, SEMINARS, AND AFFILIATIONS

WAREHOUSE SORTER
Ninja Van Express Tech. Philippines
Lot B, Pulo-Diezmo Road Cabuyao Laguna, Philippines

MEMBER
AWS Cloud Club - University of Cabuyao
https://www.facebook.com/awscepnc

CHARACTER REFERENCES

ENGR. JIM MARCO M. PABELLON
Pamantasan ng Cabuyao
Cabuyao, Laguna
pabellonjimmarco@gmail.com

FRANCIS DELOS SANTOS
Ninja Van Express Tech. Philippines
Cabuyao, Laguna
09561223174

PROFILE

I am a dedicated computer engineering student that is eager to learn more about the hardware and software aspects of technology. I am proficient in programming and other computer related skills

TECHNICAL SKILLS

- Programming skills
- Computer Literate
- Office Productivity Utilization Skills

EDUCATIONAL BACKGROUND

PAMANTASAN NG CABUYAO
Bachelor in Computer Engineering

LAGUNA BELAIR SCIENCE SCHOOL
Senior High School

LAGUNA BELAIR SCIENCE SCHOOL
Junior High School

PERSONAL DATA

GENDER:	Male
HEIGHT:	164cm
WEIGHT:	50kg
CITIZENSHIP:	Filipino
CIVIL STATUS:	Single
RELIGION:	Roman Catholic
LANGUAGE:	Filipino
BIRTH DATE:	June 7, 2000
PARENTS:	Mercedita L. Donasco Joselito V. Donasco

I hereby certify that the above information is true and correct based on my knowledge.

DONASCO, JOMER JOHN L.

Christian B. Inay

Block 9 Lot 6 Centennial Townhomes 2, Brgy. Pulo, Cabuyao, Laguna
0907 – 529 – 6948
Christianinay98@gmail.com

OBJECTIVE

To strengthen a solid foundation in computer engineering principles, obtain practical experience through internships and projects, improve problem-solving abilities, and stay current with technology changes. Aim for academic excellence while actively participating in team collaborations and networking. Develop a strong foundation in computer engineering principles, gain practical experience through internships and projects, improve problem-solving skills, and stay up to date on the latest technological advancements. Aim for academic excellence by actively participating in team collaborations.

TECHNICAL SKILLS

Microsoft Office
Basic Computer Programming
Java
Python

PERSONAL SKILLS

Time Management
Adaptability
Teamwork
Communication Skill
Problem-Solving Skill

EDUCATION

COLLEGE
University of Cabuyao
Katapatan Homes, Brgy. Banay-banay, Cabuyao, Laguna

SENIOR HIGH SCHOOL
University of Cabuyao
Katapatan Homes, Brgy. Banay-banay, Cabuyao, Laguna

SECONDARY
Balibago Integrated High School
Balibago, Sta. Rosa, Laguna

PRIMARY
Fourth Estate Elementary School
Fourth Estate Subdivision, Sucat, Parañaque City

INFORMATION

Civil Status: Single
Religion: Roman Catholic
Birthday: December 14, 1998
Mother's Name: Arlene Inay
Father's Name: Alex Inay Sr.
Nationality: Filipino
Height: 170 cm
Weight: 75 Kg

ACHIEVEMENTS

Senior High School
With Honor - Grade 11
With Honor - Grade 12
Top 2 - Grade 12

Work Experience/ Affiliation

N/A

Character Reference

Ms. Maricar Cruz
Senior High School Adviser

CHRISTIAN B. INAY
APPLICANT'S NAME

JOPH ANTHONY G. MAÑOSCA

BLK 17 L 23 PH 1 SJV 7, MARINIG, CABUYAO, LAGUNA
Akosijopjop123@gmail.com
09855482659

PROFILE

Dedicated and cautious student applying to the University of Cabuyao's Bachelor of Computer Engineering program. aiming to build a solid foundation in the topic by obtaining practical experience via school projects, internships, and volunteer activity. keen to utilize technical skills in a fast-paced, collaborative environment and add to team initiatives.

TECHNICAL SKILLS

- Management Skills
- Creativity
- Negotiation
- Critical Thinking

EDUCATIONAL BACKGROUND

Pamantasan ng Cabuyao
Bachelor of Science in Computer Engineering

Westbridge Institute of Technology, Inc
2021 – 2023

Cabuyao Integrated National School
2016 - 2020

PERSONAL DATA

GENDER:	MALE
HEIGHT:	175.26cm
WEIGHT:	77kg
CITIZENSHIP:	FILIPINO
CIVIL STATUS:	SINGLE
RELIGION:	CATHOLIC
LANGUAGE:	FILIPINO
BIRTHDATE:	JULY12,2005
PARENTS:	ANALISAGARAGARA JOPERTC.MAÑOSCA

I hereby certify that the above information is true and correct based on my knowledge.

MAÑOSCA , JOPH ANTHONY

CAREER OBJECTIVES

Career objectives a dedicated and meticulous student seeking an entry-level position in Computer Engineering where I can use my skills and advance my career. I can't wait to grow in a progressive company and add value to a vibrant team..

EXPERIENCES, SEMINARS, AND AFFILIATIONS

CHARACTER REFERENCES

Mark Aldrin Ramos
Mabuhay phase 1 Blk 20 L 3
macmacramos08@gmail.com