**Object Oriented Programing Project Report**

Figure 1



**Hospital Management System**

**Prepared by**

**at**

**Seneca Polytechnic**

**March 2024**

**Table of Contents**

[Contributions 3](#_Toc140089285)

[I Presentation YouTube link 3](#_Toc140089286)

[II Project Description 3](#_Toc140089287)

[1 Project Overview 3](#_Toc140089288)

[2 The Purpose of the Project 3](#_Toc140089289)

[The User Business or Background of the Project Effort 3](#_Toc140089290)

[3 The Scope of the Work 4](#_Toc140089291)

[III Requirements 4](#_Toc140089292)

[4 Product Use Cases 4](#_Toc140089293)

[IV Design 4](#_Toc140089294)

[5 UML Class Diagram 4](#_Toc140089295)

[V Testing and Evaluation 4](#_Toc140089296)

[6 Features to be tested. 4](#_Toc140089297)

[7 Pass/Fail Criteria 4](#_Toc140089298)

[VI Project Issues 4](#_Toc140089299)

[8 Lessons Learnt 4](#_Toc140089300)

[VII Conclusion 4](#_Toc140089301)

[VIII References 4](#_Toc140089302)

### ****Contributions****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Student ID** | **Name** | **Components** | **Details** | **Development hours** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Presentation YouTube link

# Project Description

## Project Overview

For this project, we have created a Hospital Management System that aims to facilitate efficient management of patient records, appointment scheduling, managing medical staff and procedures within a hospital. It provides a comprehensive solution for organizing patient information, managing appointments, and ensuring seamless communication between medical staff and patients.

## The Purpose of the Project

### The User Business or Background of the Project Effort

Content

In this project, we endeavor to develop a comprehensive hospital management system tailored to the healthcare sector. This system will facilitate the seamless management of patient records, appointment scheduling, medical staff tracking, and medical procedures. By incorporating various STL containers, algorithms, and iterators, we aim to optimize data storage and processing, ensuring efficient operation of the system. The need for such a sophisticated and user-friendly platform arises from the increasing demands and complexities within healthcare administration. Our objective is to deliver a robust solution that empowers healthcare professionals to efficiently manage patient information, streamline administrative tasks, and enhance overall operational efficiency within healthcare facilities.

图示

描述已自动生成

Figure: A generalized flow chart of patient visit (Chen & Wang, 2016)

Motivation

Our motivation for embarking on this project stems from the desire to hone our C++ programming skills while tackling real-world challenges in healthcare management. By undertaking the development of a hospital management system, we seek to gain practical experience in utilizing STL containers, algorithms, and iterators for efficient data handling in a critical domain. Furthermore, we recognize the importance of developing intuitive and reliable software solutions to address the complex needs of healthcare institutions. Through this project, we aim to contribute to the improvement of healthcare services by providing a robust, user-friendly, and scalable hospital management system.

Considerations

It's essential to acknowledge that while our hospital management system aims to address key functionalities such as patient record management, appointment scheduling, staff tracking, and medical procedure handling, it is intentionally simplified and may not encompass all aspects of a comprehensive hospital management system. This simplified approach allows us to focus on core functionalities and optimize the system's performance and usability within the scope of this project. However, it's important to recognize that real-world hospital management systems may involve additional complexities and features, such as billing, inventory management, electronic health records (EHR), and regulatory compliance. Future iterations or expansions of this system could explore incorporating these elements to offer a more comprehensive solution for healthcare institutions.

## The Scope of the Work

The scope of the project encompasses the design, development, and implementation of a Hospital Management System that meets the specified requirements outlined in the project guidelines. This includes the creation of patient class hierarchies, implementation of exception handling mechanisms, integration of STL containers for data storage and manipulation, appointment scheduling functionalities, management of medical staff, and user interface development.

# Requirements

## Product Use Cases

1. Search patient

2. Add patient

3. Display all patients

4. Schedule appointment

5. Reschedule appointment

6. Cancel appointment

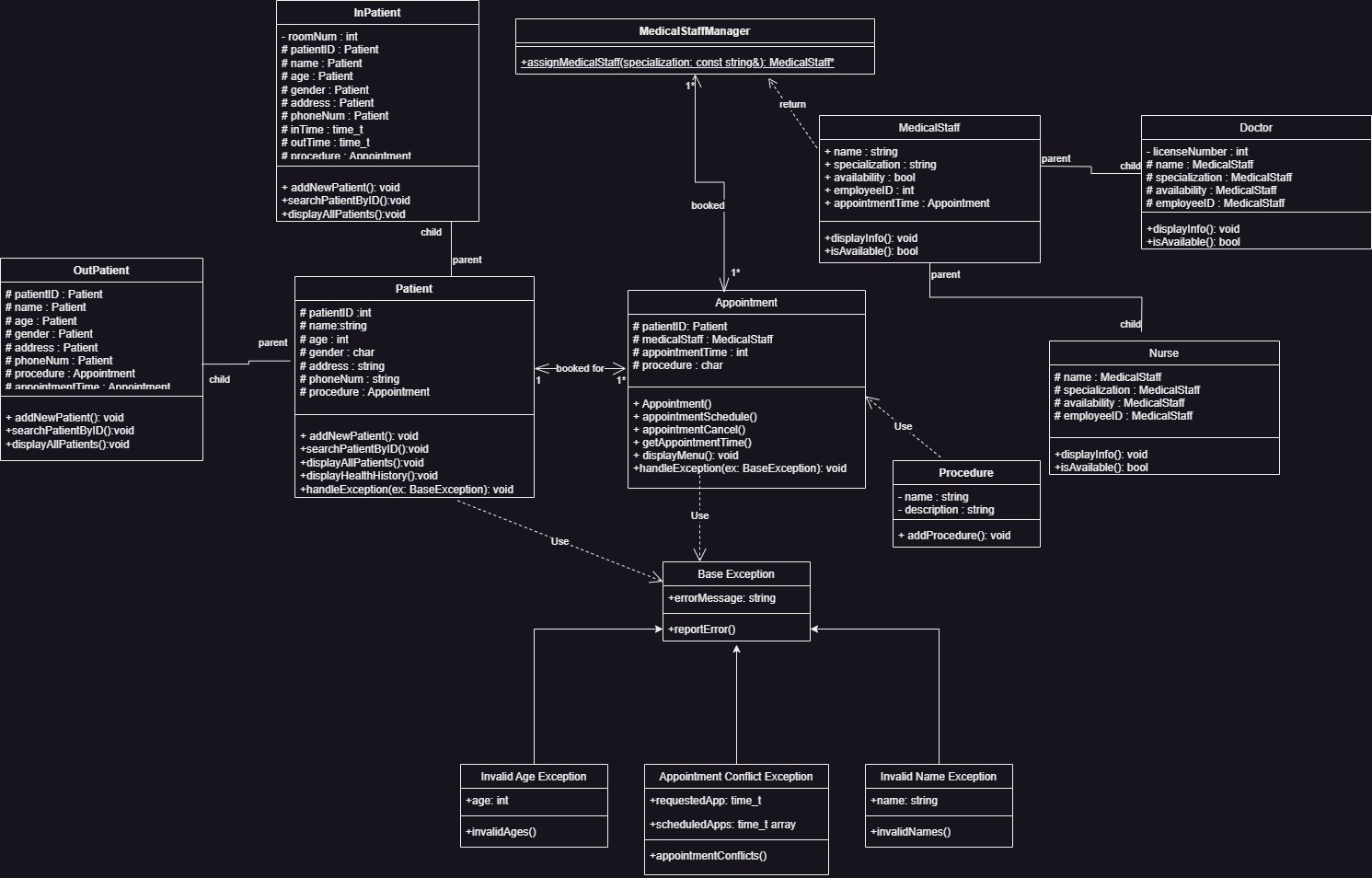
7. Display all available appointments

8. Display booked appointments

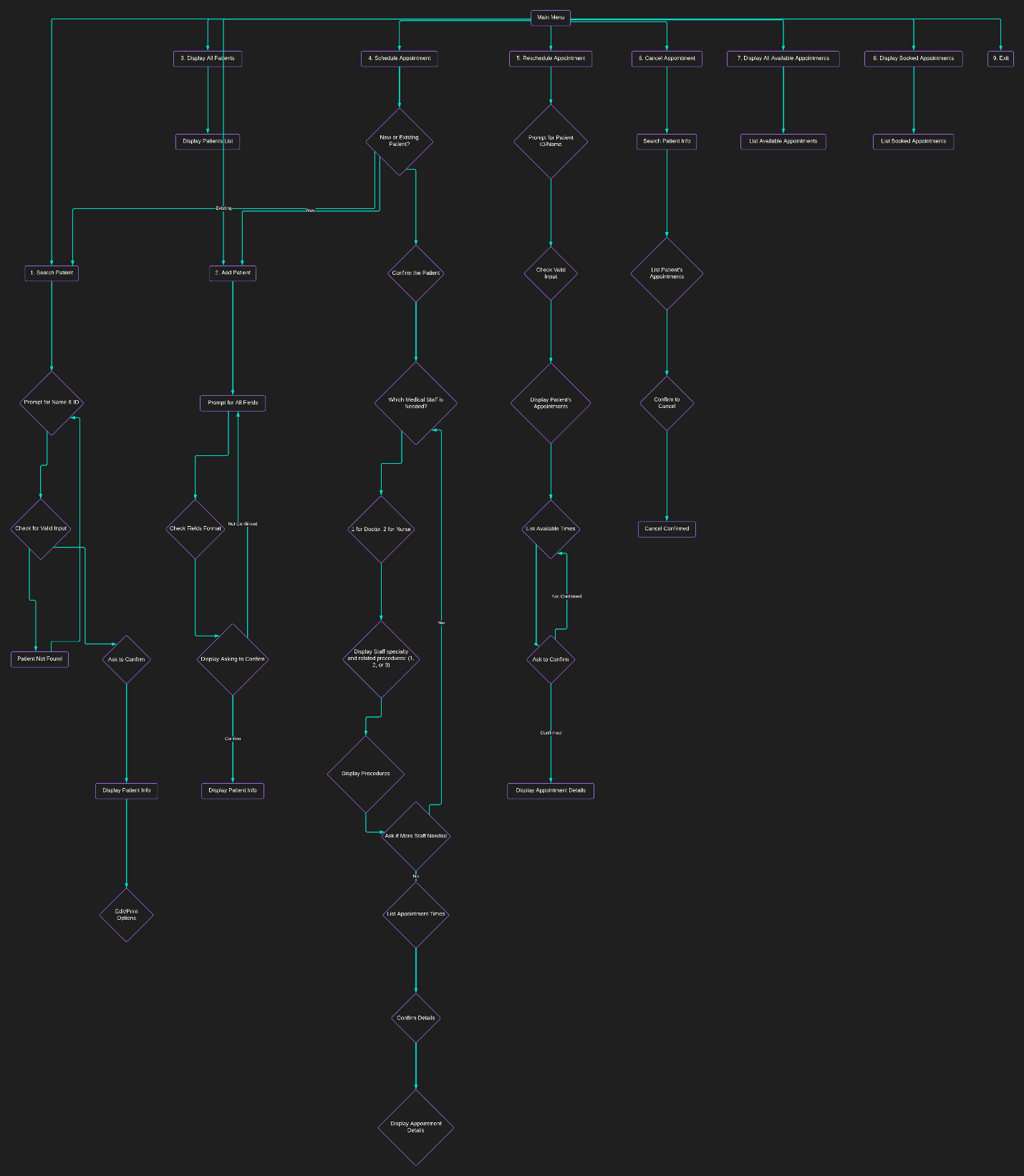
9. Exit

# Design

## UML Class Diagram



## Data Flow Diagram



# Testing and Evaluation

## Features to be tested.

* Search patient:

文本

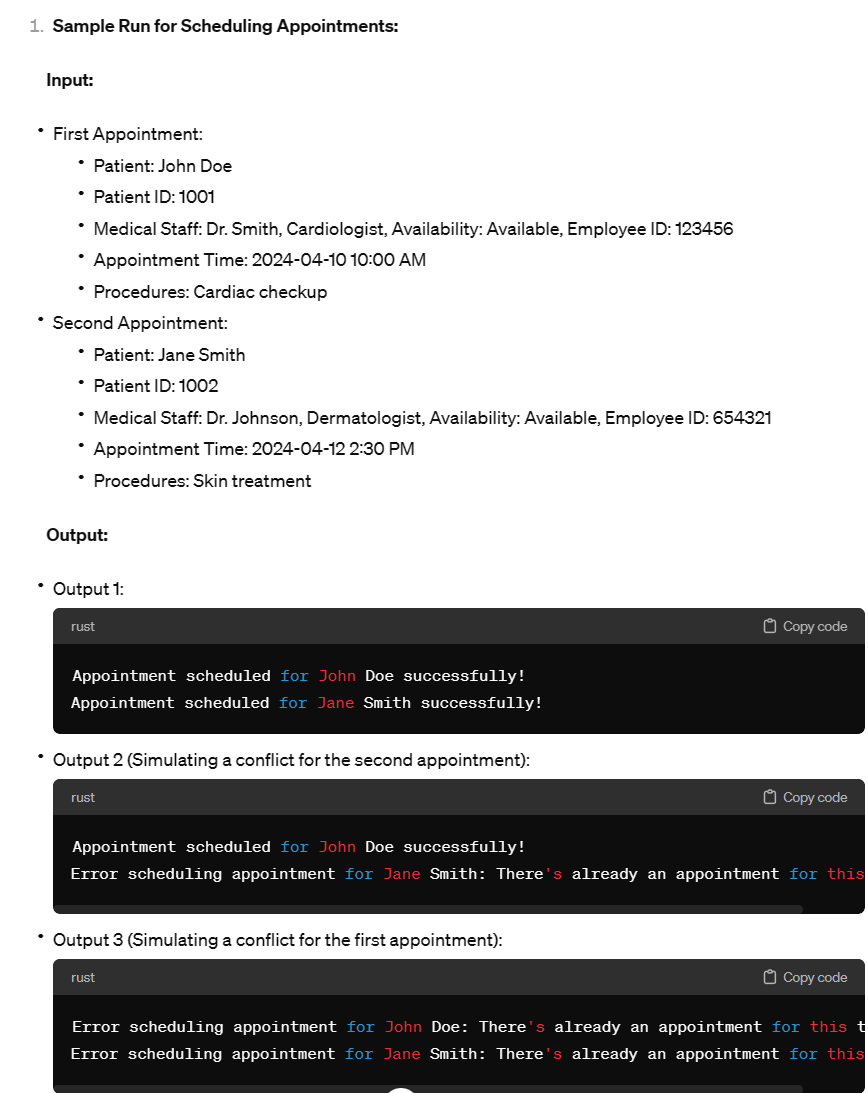
描述已自动生成

* Add patient:  
  文本

  描述已自动生成文本

  描述已自动生成  
  文本

  描述已自动生成
* Display all patients  
  文本

  描述已自动生成
* Schedule appointment  
  
* Reschedule appointment  
  文本

  描述已自动生成
* Cancel appointment  
  文本

  描述已自动生成
* Display all available appointments  
  图形用户界面

  低可信度描述已自动生成
* Display booked appointments  
  文本

  描述已自动生成
* Exit  
  图形用户界面, 文本, 网站

  描述已自动生成

## Pass/Fail Criteria

1. **Search Patient:**

**Pass:** Successfully prompts for name and ID, validates input, displays patient info if found, and allows editing/printing options.

**Fail:** Fails to prompt for necessary input, does not validate input, or does not display patient info.

2. **Add Patient:**

**Pass:** Prompts for all required fields with correct formats, confirms input, displays patient info upon confirmation, and allows editing if needed.

**Fail:** Does not prompt for required fields, lacks input validation, or does not display patient info.

3. **Display All Patients:**

**Pass:** Displays all patients with names and IDs.

**Fail:** Does not display all patients or displays incorrect information.

4. **Schedule Appointment:**

**Pass:** Successfully schedules an appointment for both new and existing patients, prompts for necessary information, confirms details, and lists available appointment times.

**Fail:** Fails to schedule appointments, does not prompt for required information, or lists incorrect appointment times.

5. **Reschedule Appointment:**

**Pass:** Allows rescheduling of appointments, prompts for patient ID/name, displays appointments, lists available times, confirms changes, and updates the appointment accordingly.

**Fail:** Unable to reschedule appointments, does not prompt for required information, or does not confirm changes.

6. **Cancel Appointment:**

**Pass:** Successfully cancels appointments, prompts for patient info, lists appointments, confirms cancellation, and updates the appointment list accordingly.

**Fail:** Unable to cancel appointments, does not prompt for required information, or does not confirm cancellation.

7. **Display All Available Appointments:**

**Pass:** Lists all available appointments for each doctor/nurse accurately.

**Fail:** Does not display available appointments or displays incorrect information.

8. **Display Booked Appointments:**

**Pass:** Lists all booked appointments showing patient ID and name, appointment time, and medical staff assigned.

**Fail:** Does not display booked appointments or displays incorrect information.

9. **Exit:**

**Pass:** Allows the user to exit the program gracefully.

**Fail:** Does not provide an option to exit or does not exit properly.

# Project Issues

## Lessons Learned

* Container of pointer:  
  The combination of usage of STL container and pointers can be confusing. In the definition of patient management system, we used a smart pointer (unique\_ptr) to point to Patient class objects and its derived class objects, and then created a list to store the pointer using push\_back function. Upon being pushed back, the ownership of the underlying linkage is actually transferred into the container, and the pointer becomes a null pointer. This is similar to a deep copy in that the activity of the pointer after this will not change the value of the one being stored into the container. To access the stored value afterwards, an iterator or a search function must be created.
* STL container vs. business needs:  
  The selection of STL container should closely match the needs of the objects in business cases. In this project, medical staff are assigned a <vector> container as staff tend to be stable and stay unchanged in most of the time, and they do not need to be updated in real-time. However, patients are assigned a <list> container as they need to be iterated according to different urgency levels, and they are reasonably expected to be updated real-time during operational hours. Deployment of the proper types of the STL containers according to different business situations can increase efficiency and robustness in daily operation, and it can reduce efforts in the extensive developments to happen in the future.
* Data validation and integrity:  
  Throughout the development of the project, we have defined reasonable ranges for variables. For example, age of the patients are set to 0-150. If the input age is detected to be out of the range, the exception handling function will be called and ask the user to input a valid data. Coverage of data validation in this project are mainly time-related, including checking for a valid date and checking for conflicts of appointment times. Based on the results from the sample runs, the developers have achieved data integrity in this project.
* User input with cin within encapsuled function bodies  
  Along the process of development, we found that, due to the stacking nature of member functions when they call each other, sometimes functions and iteration loops take user inputs and repetitively use the same user input for subsequent executions. We have researched on our own and found the reason to be that cin, as an object of input stream, stores excessive inputs for later usage. The solution is to use cin.clear() and cin.ignore() functions to clear out the memory and force the shell to wait for subsequent user inputs.

# Conclusion

Write the conclusion of the project by discussing the issues, solutions and the outcome.

# References

Image link: <https://pixabay.com/illustrations/health-care-medicine-healthy-2082630/>

UML - <https://drawio-app.com/blog/uml-class-diagrams-in-draw-io/>

HMS - <https://www.spaceo.ca/blog/hospital-management-system/>

Lucid chart: <https://www.lucidchart.com/pages/>

[1] Deitel, P. J., & Deitel, H. M. (2014). C++ How to Program. How to program series. *Prentice Hall*, *3*, 42..

[2] Anonymous et al (2015). Vector of pointer points to NULL when using push\_back(). *Stack Overflow*. <https://stackoverflow.com/questions/56637297/vector-of-pointer-points-to-null-when-using-push-back> (Retrieved Mar. 23, 2024)

[3] Chen, X., Wang, L., Ding, J., & Thomas, N. (2016). Patient Flow Scheduling and Capacity Planning in a Smart Hospital Environment. IEEE Access, 4, 135-148.

[4] Anonymous et al (2015). C++ Program not waiting for cin. *Stack Overflow*. https://stackoverflow.com/questions/12272277/program-not-waiting-for-cin/ *(*Retrieved Nov. 24 2023)