**Object Oriented Programing Project Report**



**Hospital Management System**

**Prepared by**

**at**

**Seneca Polytechnic**

**March 2024**

**Table of Contents**

[Contributions 3](#_Toc162188099)

[I Presentation YouTube link 3](#_Toc162188100)

[II Project Description 3](#_Toc162188101)

[1 Project Overview 3](#_Toc162188102)

[2 The Purpose of the Project 3](#_Toc162188103)

[The User Business or Background of the Project Effort 3](#_Toc162188104)

[3 The Scope of the Work 4](#_Toc162188105)

[III Requirements 4](#_Toc162188106)

[4 Product Use Cases 4](#_Toc162188107)

[IV Design 5](#_Toc162188108)

[5 UML Class Diagram 5](#_Toc162188109)

[6 Data Flow Diagram 6](#_Toc162188110)

[V Testing and Evaluation 6](#_Toc162188111)

[7 Features to be tested. 6](#_Toc162188112)

[8 Pass/Fail Criteria 8](#_Toc162188113)

[VI Project Issues 9](#_Toc162188114)

[9 Lessons Learned 9](#_Toc162188115)

[VII Conclusion 9](#_Toc162188116)

[VIII References 9](#_Toc162188117)

### ****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.

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

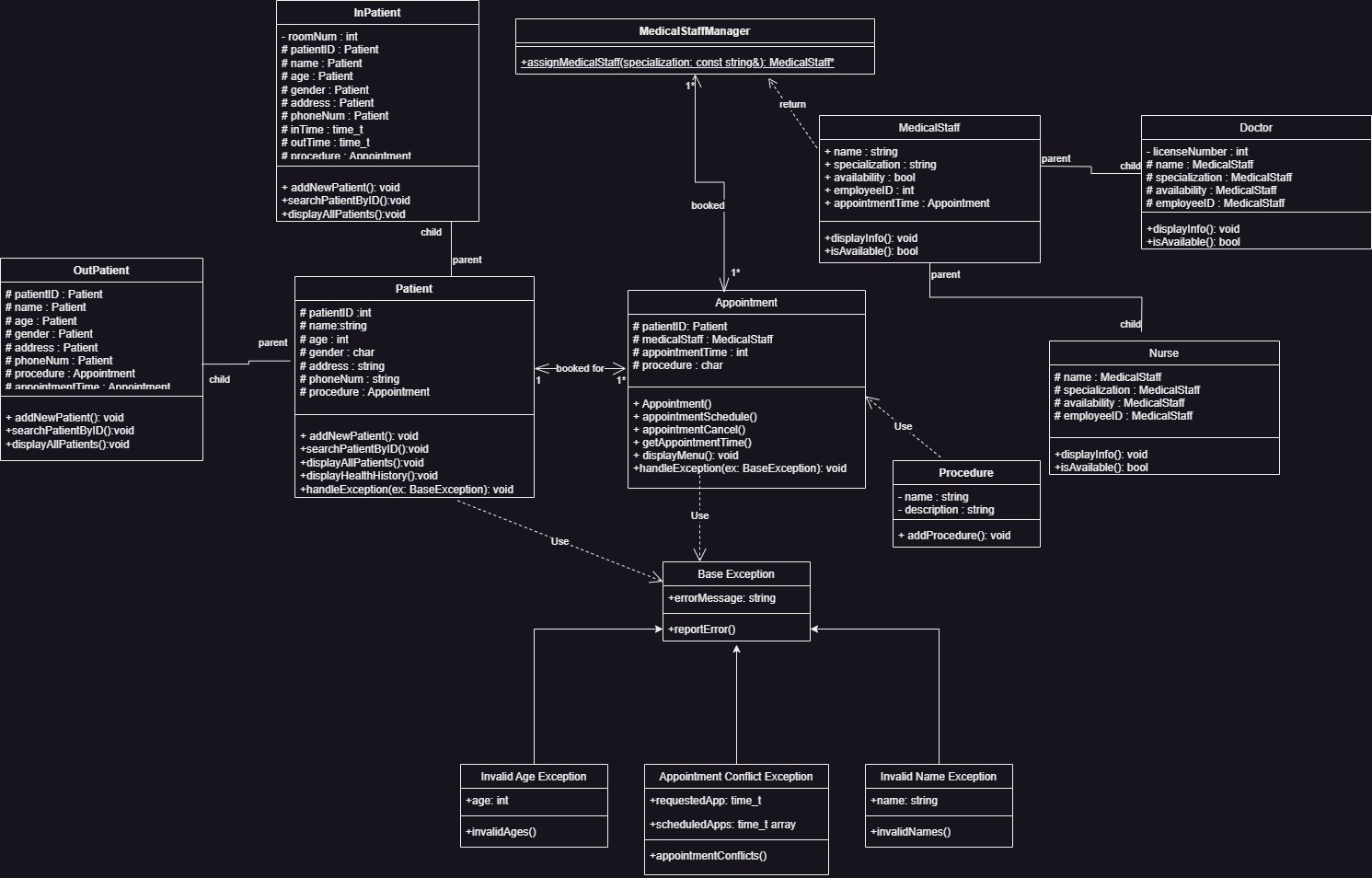
7. Display all available appointments

8. Display booked appointments

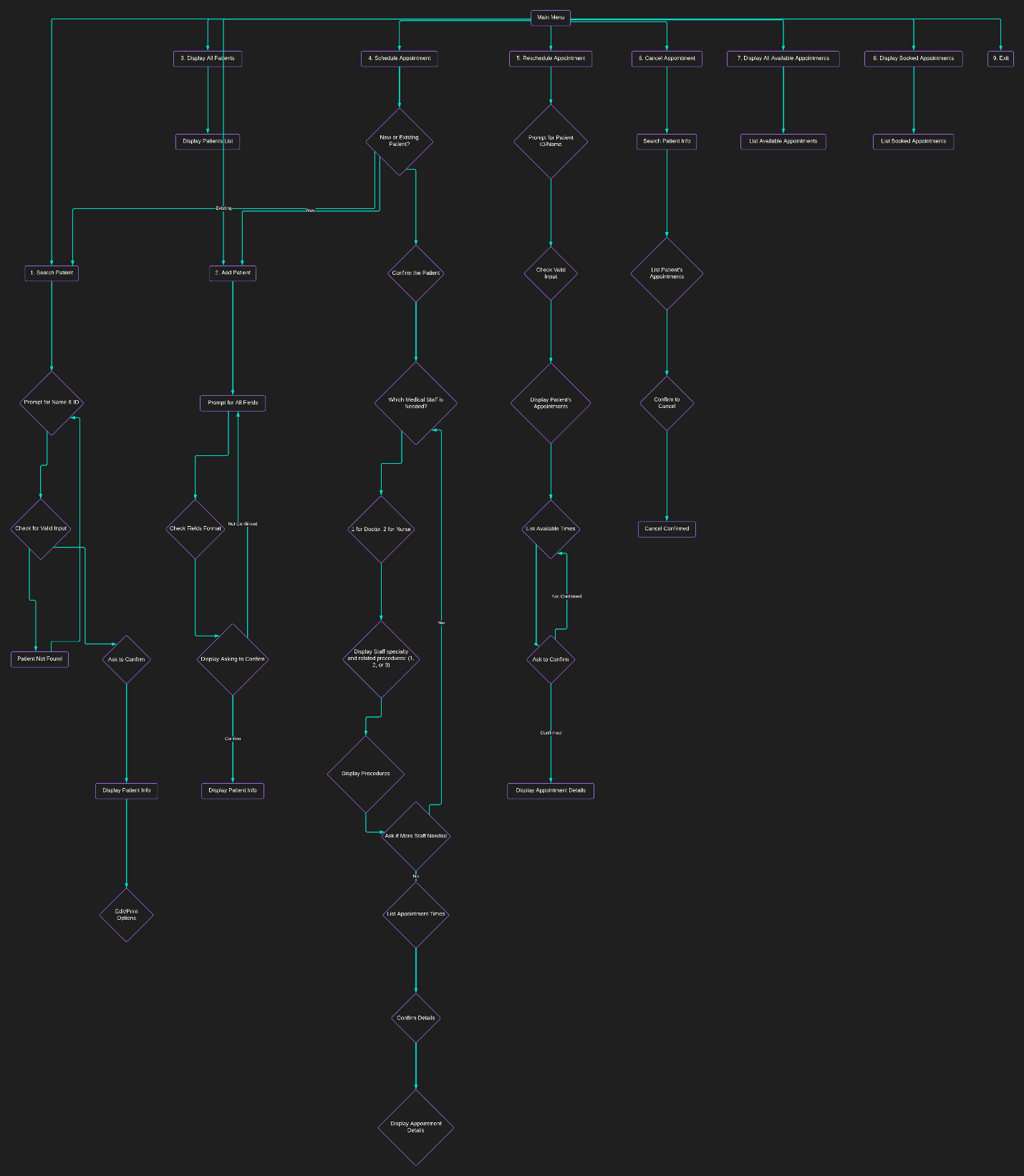
9. Exit

# Design

## UML Class Diagram



## Data Flow Diagram



# Testing and Evaluation

## Features to be tested.

* Add patient:
* Search patient:
* Display all patients
* Schedule appointment
* Reschedule appointment
* Cancel appointment
* Display all available appointments
* Display booked appointments
* Exit

## 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

During the development of the patient management system, several key considerations emerged, shedding light on best practices and challenges encountered in the project.

The utilization of a container of pointers posed a notable challenge. By combining STL containers with pointers, complexities arose regarding ownership and manipulation of the underlying objects. For instance, employing a smart pointer, such as unique\_ptr, to manage objects of the Patient class and its derivatives, led to the transfer of ownership upon insertion into the container. Consequently, the pointer would become null, akin to a deep copy scenario. To access stored values thereafter, the implementation of iterators or search functions became necessary.

Another significant aspect pertained to the selection of appropriate STL containers aligned with business requirements. Matching container types with the characteristics of the data entities proved crucial for operational efficiency. For instance, assigning medical staff to a vector container suited their stable nature, minimizing real-time updates. Conversely, patients were assigned a list container due to dynamic characteristics necessitating real-time updates. Employing suitable container types based on business contexts enhanced operational robustness and streamlined future development efforts.

Maintaining data integrity through comprehensive validation procedures constituted another focal point. Defining reasonable ranges for variables, such as patient ages, and implementing exception handling for out-of-range inputs ensured data reliability. Additionally, validating temporal data and resolving scheduling conflicts contributed to the overall integrity of the system.

An intriguing challenge encountered during development related to user input handling within encapsulated function bodies. The iterative nature of function calls and loops sometimes resulted in repetitive usage of user inputs. Research revealed that the input stream object, cin, retained excess input, affecting subsequent executions. Mitigation strategies involving cin.clear() and cin.ignore() functions were implemented to manage input stream memory and facilitate user interactions.

Reflection on project management practices highlighted areas for improvement, particularly in work distribution and time management. Regular check-ins and reassessment of roles could have ensured alignment with initial commitments, preventing disruptions, delays, imbalance of work, and enhancing team efficiency.

In summary, addressing these challenges and implementing best practices contributed to the successful development of the patient management system, fostering operational reliability, and facilitating future enhancements.

# Conclusion

In conclusion, the development of the Hospital Management System presented challenges in managing pointers within STL containers, aligning container selection with business needs, and ensuring data integrity. Mitigation strategies and best practices were employed to navigate these challenges successfully. Reflection on project management highlighted areas for improvement, emphasizing the importance of effective communication and time management. Ultimately, the project culminated in a robust solution poised to enhance operational efficiency and patient care within healthcare institutions.

# References

Charming Data. (2023, May 29). Push code to your GitHub account – Under 3 minutes [Video]. YouTube. https://www.youtube.com/watch?v=vpRkAoCqX3o

Fergusson, K. (2018, March 1). UML class diagrams in Draw.io. https://drawio-app.com/blog/uml-class-diagrams-in-draw-io/

Lucidchart. (n.d.). Lucidchart. https://www.lucidchart.com/pages/

Pixabay. (n.d.). Health care medicine healthy. [Illustration]. Pixabay. https://pixabay.com/illustrations/health-care-medicine-healthy-2082630/

Portfolio C. (2023, March 26). Exception handling | C++ tutorial [Video]. YouTube. https://youtube.com/watch?v=5nCXSDv6e4I

Simplilearn. (2021, December 2). Exceptions handling in C++ | What is exception handling in C++ | C++ programming | Simplilearn [Video]. YouTube. https://www.youtube.com/watch?v=7hcQQEHZPiQ

Spaceo. (n.d.). Hospital management system. Spaceo. https://www.spaceo.ca/blog/hospital-management-system/

The Cherno. (2017, September 3). Operators and operator overloading in C++ [Video]. YouTube. https://www.youtube.com/watch?v=mS9755gF66w