Object Oriented

**Hospital Management System Project**



Prepared by

**Jianfeng Lin**

**Baoling Zhang**

**Chandler Mckelvie**

**Steven Lam**

at the

**Seneca College**

November 2023

**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)

[2a 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** |
| 120949227 | **Jianfeng Lin** | Code |  | 10 |
| 121280226 | **Baoling Zhang** | Code and report |  |  |
| 186118212 | **Chandler Mckelvie** | Code |  |  |
| 104236153 | **Steven Lam** | Code |  |  |

# Presentation YouTube link

Run through of application: <https://www.youtube.com/watch?v=A_nUMly9DkU>

Github repo: https://github.com/StvnLm/OOP\_Assignment1\_Backend/commits/develop?after=6bf254943b6fadcf2f880f3f51dd140dacce3793+34&branch=develop&qualified\_name=refs%2Fheads%2Fdevelop

# Project Description

## Project Overview

The Hospital Management System aims to streamline the administrative and operational tasks within a hospital, ultimately enhancing patient care and improving the efficiency of healthcare providers. The system will provide a user-friendly interface for hospital staff to manage patient information, appointments, and medical staff schedules.

## The Purpose of the Project

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

## Motivation:

## The motivation behind this project is rooted in the need for modern healthcare facilities to improve their management processes. Manual record-keeping and appointment scheduling can lead to errors, scheduling conflicts, and inefficiencies. By automating these tasks and utilizing STL for data management, the Hospital Management System aims to:

## - Enhance patient care by reducing the likelihood of errors and delays.

## - Improve the overall efficiency of hospital operations.

## - Provide a centralized and easily accessible repository for patient data.

## - Ensure accurate and timely scheduling of medical appointments.

## - Enhance data security and patient privacy through proper exception handling and data validation.

## Considerations:

## Considerations for this project include:

## - The potential impact of improved hospital management on patient outcomes and satisfaction.

## - The need for a robust and reliable system to handle sensitive patient data.

## - The efficiency gains and cost savings that can be achieved through automation.

## - The importance of adhering to best practices in object-oriented programming, exception handling, and STL usage.

## - The need for thorough testing to validate the correctness and reliability of the system.

## The Scope of the Work

# Requirements

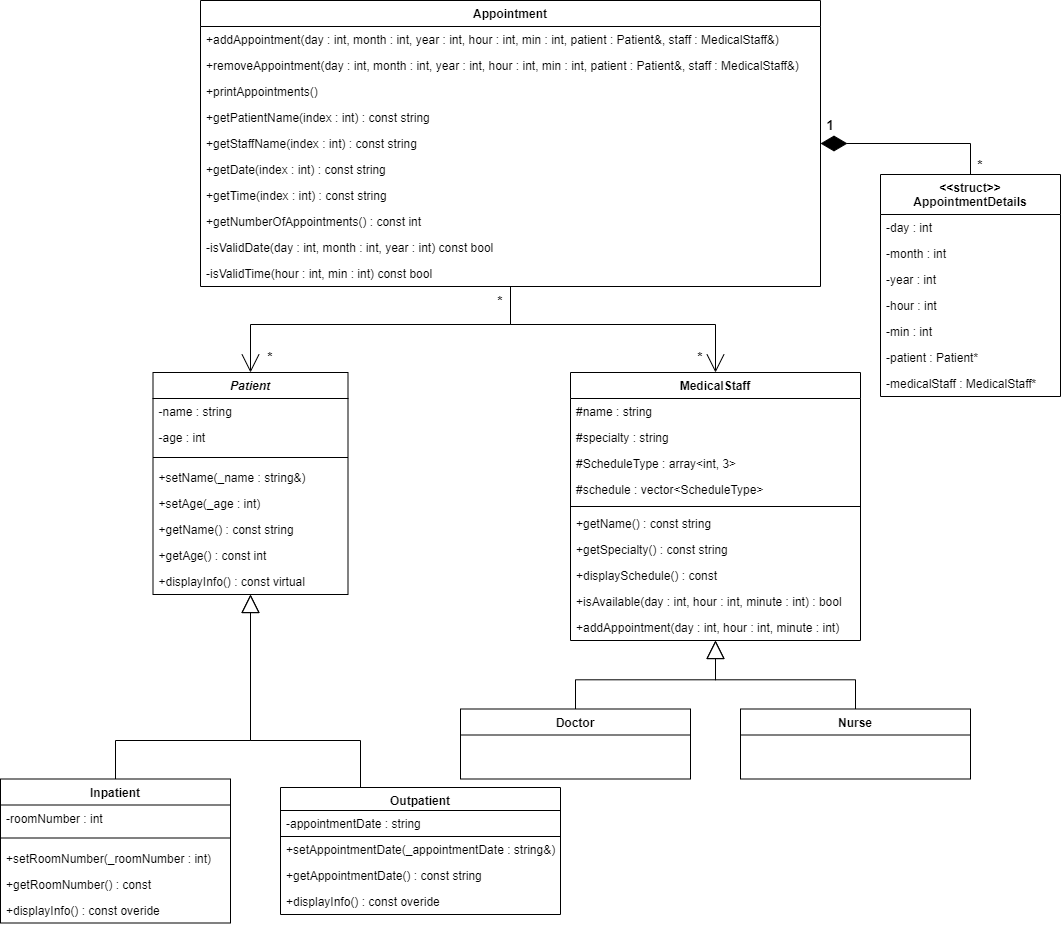
## Product Use Cases

1. **Main Menu Display:** The system starts by displaying a menu for a hospital management system with options like adding patients, scheduling appointments, managing medical staff, displaying patient records, saving and loading data, and exiting the system.
2. **Adding Patients:** The user can add inpatients or outpatients by entering their names, ages, and relevant information. Before adding a patient, the system checks if the entered age is within a valid range (0-120). If it's not, an "InvalidAgeException" error is shown.
3. **Scheduling Appointments:** To schedule an appointment, the system lists available patients and medical staff. The user selects a patient and a staff member, then inputs a date and time for the appointment. It checks the availability of the selected staff member and schedules the appointment if they are available. Otherwise, it notifies that the staff member is not available at that time.
4. **Managing Medical Staff:** Users can add doctors or nurses to the system. They input the name and specialty of the staff member, and it gets added to the system's records. The system also allows displaying the existing medical staff.
5. **Displaying Patient Records:** This option allows the user to view the details of all the patients currently in the system.
6. **Saving and Loading Data:** The system provides functionality to save the entered patient and medical staff data to a file named "hospital\_data.txt". It saves each patient's name, age, medical staff's name, specialty, appointment date, and time. Additionally, it allows loading data from this file when the "Load Data from File" option is chosen.
7. **Exiting the System:** Finally, the user can choose to exit the system, which deallocates the memory of dynamically allocated objects (patients), clears the patient records, and exits the program.

Throughout these steps, the system incorporates error handling through custom exception classes like "InvalidAgeException" and uses a try-catch block to catch and display any exceptions that might occur during the execution of the program.

# Design

## UML Class Diagram



# Testing and Evaluation

## Features to be tested.

**Patient Record Management:**

* Adding, updating, and deleting patient records.
* Searching for patient records by name, ID, or other criteria.

**Appointment Scheduling**:

* Creating new appointments.
* Handling appointment conflicts.

**Medical Staff Management:**

* Assigning staff to appointments.
* Managing staff schedules and availability.

**Exception Handling:**

* Handling invalid data inputs (e.g., incorrect age, invalid appointment times).
* Custom exceptions such as InvalidAgeException or AppointmentConflictException.

**Data Storage and Retrieval:**

* Persistence and retrieval of data (if applicable).
* Integrity and consistency of data after operations like add, delete, or update.

**User Interface:**

* Usability and responsiveness of the UI.
* Error handling and user feedback in the UI.

**System Performance:**

* Response time for various operations.
* Handling of large volumes of data.

## Pass/Fail Criteria

**Patient Record Management:**

* Pass: Records are accurately added, updated, and deleted. Search functions return correct results.
* Fail: Incorrect handling of patient data or failure in searching records.

**Appointment Scheduling:**

* Pass: Appointments are correctly scheduled without conflicts.
* Fail: Appointment conflicts are not handled

**Medical Staff Management:**

* Pass: Staff are correctly assigned and their schedules are accurately managed.
* Fail: Incorrect assignment of staff or errors in schedule management.

**Exception Handling:**

* Pass: All exceptions are caught and handled gracefully with appropriate user feedback.
* Fail: Exceptions crash the program or are not correctly identified.

**Data Storage and Retrieval:**

* Pass: Data is consistently and accurately stored and retrieved.
* Fail: Loss of data integrity or failure in data persistence.

**User Interface:**

* Pass: UI is intuitive, responsive, and provides correct feedback.
* Fail: UI is unresponsive, confusing to navigate, or provides incorrect feedback.

**System Performance:**

* Pass: System responds quickly to operations and handles large data efficiently.
* Fail: Slow response times or inability to manage large data sets.

# Project Issues

## Complexity in Implementing STL: The integration of STL in managing patient data and appointment scheduling posed a significant challenge due to the complexity and variety of STL containers and algorithms.

## Exception Handling: Crafting robust exception handling mechanisms, especially for custom exceptions like `InvalidAgeException` and `AppointmentConflictException`, required careful consideration to ensure system stability.

## Appointment Scheduling Conflicts: Managing appointment scheduling without conflicts was a complex task, necessitating efficient data structures and algorithms.

## Lessons Learnt

**Importance of STL in C++:** The project underscored the versatility and power of the STL in managing complex data and algorithms efficiently.

**Robustness through Exception Handling:** Exception handling emerged as a cornerstone for building a stable and reliable system.

**User-Centric Design:** The project highlighted the significance of a user-centric approach in software design, especially in a complex domain like healthcare.

# Conclusion

In concluding the Hospital Management System project, several key aspects stand out, encompassing the issues encountered, solutions implemented, and the overall outcome of the project.

# References

McConnell, S. (2013). *Code Complete: A Practical Handbook of Software.*

Sommerville. (2015). *Software Engineering. Pearson.*

(Sommerville, 2015) (McConnell, 2013)