

OBJECT ORIENTED DES & PROG(SC2002)

**Health Management System**

**Tutorial Group: SCE4**

**Team Number: 6**

|  |  |  |
| --- | --- | --- |
| **Members** | **Matrix No.** | **Github ID** |
| Soo Qin Loong Marcus | U2322313C | marcussoo |
| Heng Guan Xiang | U2322018L | HengGuanXiang |
| Tay Yuan Geng | U2322020D | TayYuanGeng |
| Tan Yong Quan Nicholas | U2321573D | nichtyq |
| Yeoh Zhe Wei | U2321990D | zw1010 |

Font: Times new Roman 12 pt font size, 1.5 spacing

Github Link: <https://github.com/TayYuanGeng/SC2002-Assignment>

**Declaration of Original Work for CE/CZ2002 Assignment**

We hereby declare that the attached group assignment has been researched, undertaken, completed, and submitted as a collective effort by the group members listed below.

We have honoured the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work.

We understand that if plagiarism is found in the assignment, then lower marks or no marks will be rewards for the assessed work. In addition, disciplinary actions may be taken.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Course | Lab Group | Signature / Date |
| Soo Qin Loong Marcus | SC2002 | SCE4 | Marcus 12/11/2024 |
| Heng Guan Xiang | SC2002 | SCE4 | Guan Xiang 12/11/2024 |
| Tay Yuan Geng | SC2002 | SCE4 | Yuan Geng 12/11/2024 |
| Tan Yong Quan Nicholas | SC2002 | SCE4 | Nicholas 12/11/2024 |
| Yeoh Zhe Wei | SC2002 | SCE4 | Zhe Wei 12/11/2024 |

Important Notes:

1. Name must **EXACTLY MATCH** the one printed on your Matriculation Card.
2. Student Code of Academic Conduct includes the latest guidelines on usage of Generative AI and any other guidelines as released by NTU.

[**1. Introduction 4**](#_k9hxuta24pxg)

[**2. Design Consideration 4**](#_lj4fb6suw8j)

[i. SOLID Principles 4](#_ywzpv8cx3rgw)

[ii. OOP Concepts 6](#_ut3fa11xvqk)

[**3. Assumptions Made 6**](#_4jh9g9tclep4)

[**4. Detailed UML class diagram 7**](#_sitne4orgrq0)

[**5. Additional Features 8**](#_yo8r1cdp6wlu)

[**6. Testing (Test cases and results) 9**](#_bu3gdji9196n)

[i. Patient Actions 9](#_8ntjcbchuqku)

[ii. Doctor Actions 10](#_wypkh9p48pd3)

[iii. Pharmacist Actions 10](#_b4tfr0mf3emb)

[iv. Administrator Actions 11](#_w5jv7xq5d8nk)

[v. Login System and Password Management 11](#_b0es1as1ennc)

[**7. Reflection 12**](#_voofq4dxhx25)

[i. Marcus 12](#_icspkh5yz3gx)

[ii. Guan Xiang 12](#_yb6wdiblzkoq)

[iii. Yuan Geng 12](#_a142l1hy74ed)

[iv. Nicholas 12](#_b7dj83uaku4w)

[v. Zhe Wei 12](#_3qfc67sl5gtv)

# Introduction

In this project, our group developed a Hospital Management System, applying object-oriented concepts and design principles. This report covers our design considerations, presents the UML diagram, demonstrates various test cases, and includes our reflections on the development process.

# Design Consideration

## i. SOLID Principles

1. **Single Responsibility Principle (SRP)**

The Single Responsibility Principle (SRP) states that each class should have only one responsibility and only one reason to change. In our application, some of the classes that follows SRP includes

1. **CSVUtilsController**

The CSVUtilsController class only has a single responsibility which is to read or write any data to and from the csv files which contains information that is required from the application.

1. **Models**

All of our models follow the SRP.

1. **Open-Closed Principle (OCP)**

The Open-Closed Principle states that a class should be open for extension but close for modification. If there is a need to add a new function there is not a need to change the existing implementation of the main class. An example of this principle in our application would be

1. **Account**

The Account class is an abstract class that would be inherited by the different subclasses for example, Patient, Doctor or Administrator. This allows new account types to be created without changing any implementation in the Account class.

1. **Liskov Substitution Principle (LSP)**

The Liskov Substitution Principle states that when an instance of a class is extended to another class, the inheriting class should preserve the expected behaviours and guarantees of the superclass.

1. **Staff**

For all subclasses under Staff, such as Doctor, Administrator and Pharmacist, they preserve the expected behaviours when inheriting the superclass.

1. **Interface Segregation Principle (ISP)**

The Interface Segregation Principle states that it is better to have several small interfaces rather than a big general purpose interface. Such that a class would not need to have the implementation for a method it does not require.

1. **CSVUtilsInterface**

Further changes to CSV related means that we could change the implementation without changing the rest of the code that uses USVUtilsInterface.

1. **PasswordUtilsInterface**

Further changes to password related means that we could change the implementation without changing the rest of the code that uses PasswordUtilsInterface.

1. **Dependency Inversion Principle (DIP)**

Dependency Inversion Principle states that higher level modules should not depend on low level modules, both should depend on abstractions. An example of this in our application would be

1. **IMedicalRecordService**

The PatientController depends on the IMedicalRecordService rather than the concrete implementation of MedicalRecordService. The IMedicalRecordService serves as an abstraction between Patient and MedicalRecordService

## ii. OOP Concepts

1. **Abstraction**

Through abstraction by breaking down complex things by creating objects, classes and variables which can also help to avoid repeating codes. For example, we created abstract class “Account” and interfaces such as CSVUtilsInterface and PasswordUtilsInterface.

1. **Encapsulation**

Our model classes include private attributes, and are provided access to these attributes through public methods. This concept ensures that data is kept safe.

1. **Polymorphism**

We implemented both method overloading and method overriding which are forms of polymorphism.

1. **Inheritance**

Create new subclasses that share some of the attributes of existing superclasses. Etc, Superclass: Staff and Subclass: Doctor, Pharmacist, Administrator

# Assumptions Made

|  |  |
| --- | --- |
| All Users | 1. Users must change their password after their initial login. |
| Appointments | 1. Appointments are booked in a per hour basis  2. Working hours are from 8AM to 5PM |
| Medical Records | Certain parts of Medical Records can only be updated by doctor one of the requirements   * Update records method also take in user as argument and check user role |

# Detailed UML class diagram

Placed under github uml diagram folder.

[UML diagram](https://github.com/TayYuanGeng/SC2002-Assignment/blob/main/HospitalManagementSystem/UML%20diagram/Class_Diagramof_Hospital_Management_System_Vertical.jpg)

# Additional Features

* 1. **Password Hashing**
     1. Password when updated onto the csv files are hashed to ensure confidentiality
     2. Password hashing algorithm used is SHA-256
  2. **Password Requirements**
     1. Passwords must be 8 to 16 characters long.
     2. Contains at least one uppercase letter.
     3. Contains at least one special character (such as !@#$%^&\*, etc.)
     4. Does not contain the default password “Password”.
  3. **Viewing of doctors and/or date availability**
     1. Flexibility added into the original requirement, where patients are able to:
        1. View specific doctor’s unavailability
        2. View specific date where doctors are unavailable
        3. View specific date and doctor’s unavailability
        4. View all unavailable date and time slots
        5. View available slots of a doctor and date
  4. **Viewing of Medicine by Administrator**
     1. Ability to sort the Medicine List by:
        1. Name
        2. Stock Level
        3. Low Level Alert

# Testing (Test cases and results)

### i. Patient Actions

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Test Description** | **Expected Result** | **Pass/Fail** |
| 1 | **View Medical Record** | Medical record details displayed | Pass |
| 2 | **Update Personal Information**  Enter new email and contact number | “Contact Details Updated!” | Pass |
| 3 | **View Available Appointment Slots**  Choose different filter methods for appointments | Unavailable/available time slots displayed | Pass |
| 4 | **Schedule an Appointment**  Enter date and time of appointment | “Appointment Scheduled” | Pass |
| 5 | **Reschedule an Appointment**  Enter date and time of appointment to reschedule  Enter date and time of appointment to reschedule to | “Appointment Rescheduled successfully!” | Pass |
| 6 | **Cancel an Appointment**  Enter date and time of appointment to cancel | “Appointment cancelled successfully! The time slot has been made available.” | Pass |
| 7 | **View Scheduled Appointments** | Scheduled Appointment details displayed | Pass |
| 8 | **View Past Appointment Outcome Records** | Past Appointment Outcome displayed | Pass |

### ii. Doctor Actions

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Test Description** | **Expected Result** | **Pass/Fail** |
| 9 | **View Patient Medical Record (Assigned Patient)** | Medical record details displayed | Pass |
| 10 | **Update Patient Medical Record (Assigned Patient)** | “Diagnosis and Treatment updated” | Pass |
| 11 | **View Personal Schedule** | Unavailable time slots displayed | Pass |
| 12 | **Set Availability for Appointments (New DateTime)** | - | Pass |
| 13 | **Accept or Decline Appointment Requests (Reject Accepted Appointment)** | “Appointment declined!” | Pass |
| 14 | **View Upcoming Appointments** | Upcoming appointments displayed | Pass |
| 15 | **Record Appointment Outcome (Existing Appointment)** | Appointment outcome details displayed | Pass |
| 16 | **Logout** | Login menu displayed | Pass |

### iii. Pharmacist Actions

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Test Description** | **Expected Result** | **Pass/Fail** |
| 17 | **View Appointment Outcome Record** | Appointment outcome record displayed | Pass |
| 18 | **Update Prescription Status (Pending)** | “Medications dispensed!” | Pass |
| 19 | **View Medication Inventory** | Medicine inventory displayed | Pass |
| 20 | **Submit Replenishment Request** | “Replenish request submitted!” | Pass |
| 21 | **Logout** | Login menu displayed | Pass |

### iv. Administrator Actions

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Test Description** | **Expected Result** | **Pass/Fail** |
| 22 | **View Staff Records** | Staff Details displayed | Pass |
| 23 | **Sort Staff List** | Staff list sorted based on ID/Name/Age/Role | Pass |
| 24 | **Add Staff** | Staff added successfully | Pass |
| 25 | **Remove Staff** | Staff Removed Successfully | Pass |
| 26 | **Edit Staff** | Details edited successfully | Pass |

### v. Login System and Password Management

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Test Description** | **Expected Result** | **Pass/Fail** |
| 25 | **First-Time Login and Password Change** | User logged in successfully | Pass |
| 26 | **Login with Incorrect Credentials** | "Incorrect login. (1) Try again or (2) Exit:" | Pass |

# 

# Reflection

## Marcus

In this project, coordinating and collaboration was one of the most challenging parts in this group project. However, by communicating the problems we faced, we are able to help one another and tackle these challenges together. I have learnt a lot of things in OODP like encapsulation, inheritance, polymorphism, and abstraction, which are the backbone of scalable and maintainable software design. All of which will be applicable in the future when I enter the workforce. One improvement would be to create a separate dedicated views class as well as implementing more interfaces. Overall, this module was enjoyable and the things learnt is a fresh take compared to the other modules which are mathematics heavy.

## Guan Xiang

You can see everyone’s different style of writing or development of code, clashing of ideals on how the program should be structured is not unusual but with clear communication and direction it can be resolved easily. Although this is the first time I have interacted with Java, I have learnt Inheritance, Polymorphism before but Exception Handling was very helpful. Class diagrams and design principles were all very good refreshers for me. I think the module can be improved by implementing harder weekly group assignments with a shorter timeframe and at least 20 group members.

## Yuan Geng

It was harder than I thought to create a UML diagram before starting the coding process. I wasn't able to fully comprehend the standards by simply reading them without any real-world experience. We therefore made the decision to alter our approach, starting with coding and letting the UML diagram evolve gradually rather than striving for a perfect UML diagram from the beginning. This approach was incredibly helpful since it allowed me to work on the management system while immediately applying the topics I had learnt in lectures.

## Nicholas

From this project, I was able to learn how the SOLID principles come together along with the other OOP concepts such as abstraction, encapsulation, polymorphism and inheritance. It was quite challenging at the start to come up with the class diagram that fits the SOLID principles, but after some discussion and research we were able to come up with it ensuring that our design is scalable and easy to understand. Something we can improve on is to ensure that all aspects of our design are aligned with the SOLID principles.

## Zhe Wei

This assignment allowed me to apply the concepts of object oriented programming into an application that simulates real world problems. By having such a problem, it forces me to think of the design process starting from the class diagram, which translates into code. The challenges faced during coding evidently showed that good code design reduced the amount of problems that programmers have to deal with when collaborating with each other. For example, if the code is not modular and many parts depend on each other, the development process will be gruelling as much more communication and waiting will be required, hence a longer project completion time.