



SC2002 Object Oriented Design and Programming

Hospital Management System

SCSI Assignment Group 2

Name	Matriculation Number
CHONG KWANG CHEN	U2321826L
HNG CHERNG KHAI	U2320773F
LIU YANZHI	U2320837H
TAN SHI QI	U2320951L
YAP MEI YEE	U2321703A

1. Design Considerations

1.1 Design Approach

HMS (Hospital Management System) is a Command Line Interface (CLI) application designed for managing hospital operations, including staff, patient, appointment, and medicine management. It provides a streamlined way for administrative processes.

Our system utilises boundary, controller, and entity classes. The repositories are entity classes, where data is fetched and stored. Controllers such as AdminController and MedicineController implement logic to ensure operations can be carried out to meet the system's functional requirements. The UI classes act as boundary classes, where users can interact with the system to perform required actions.

1.2 Highlights

In our project, session cookies are a highlight. It provides efficient user session management by storing the user ID, the user's role within the system, and the session's start time. This improves the security of the system and eases user interactions.

1.3 SOLID Design Principles

1.3.1 Single Responsibility Principle

The Single Responsibility Principle states that a class should have only one responsibility and thus only one reason for the class to change. One example of the use of this principle in our project is the MedicineController Class. This class only manages operations related to medicine, such as adding, removing and updating medicines. It focuses on one task which is medicine management.

1.3.2 Open-Closed Principle

The Open-Closed Principle states that classes should be open for extension but closed for modification. In our HMS system, the MainUI class is an abstract class that serves as the superclass of all other UI classes. It defines the general methods shared by all UI classes (eg. getUserChoice(), printBreadCrumbs()) and leaves other implementations to subclasses (eg.

start(), printChoice()). New UI classes can be added by extending MainUI class without modifying MainUI class.

1.3.3 Liskov Substitution Principle

The Liskov Substitution Principle states that subtypes must be substitutable for their base types. The classes that show this principle include Doctor and Pharmacist classes. They are subclasses of HMSPersonnel class, which provides a template for common behaviour. Doctor and Pharmacist can be substituted for HMSPersonnel, for instance, AdminController.printPersonnelDetails (HMSPersonnel personnel) when required. Although Doctor and Pharmacist have extra attributes and methods, they can be treated as HMSPersonnel, adhering to the Liskov Substitution Principle.

1.3.4 Interface Segregation Principle

The Interface Segregation Principle states that a client should never be forced to implement an interface that does not use or depends on the methods they do not use. In our system design, the role-specific UI classes demonstrate this principle. For example, the DoctorUI and AdminUI. The Doctor UI handles the user interface for doctors and focuses on actions like adding appointment outcomes, setting availability, and viewing personal schedules. The AdminUI would focus on actions like viewing staff, managing medicine inventory, and viewing all appointments.

Although the UI classes are subclasses of MainUI, each of them has only the methods that are relevant to its specific role. The DoctorUI does not have a function for updating patient personal details, and the AdminUI does not have a function to schedule an appointment. They are not implemented with the methods that are not required by their respective roles.

1.3.5 Dependency Inversion Principle

The Dependency Inversion Principle states that high-level modules should not depend on low-level modules, and both should depend on abstraction. In our codes, the repositories act as abstracted data access layers. Repositories like PersonnelRepository and RecordsRepository abstract the details of data access such as storing and fetching the data. The high-level modules like controllers and UIs interact with them through defined methods. If the ways to fetch or store

data are to be changed, the modification can be done in repositories without modifying the UI classes and controller classes.

1.4 Object-Oriented Concepts

1.4.1 Abstraction

Abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects and provide defined conceptual boundaries, relative to the perspective of the viewer. For our project, there are abstract classes like Repository and MainUI. They serve as the templates that define the common properties and methods, but they do not implement the full functionalities. Their subclasses will implement the methods related to their functionalities, simplifying the system.

1.4.2 Encapsulation and Information Hiding

Encapsulation builds a barrier to protect objects' private data, while information hiding hides the details or implementation of class from users.

Classes such as HMSPersonnel encapsulate their attributes and provide getters and setters for accessing and modifying these attributes. The use of private data attributes in the model classes ensures that data cannot be accessed outside the class, which maintains data integrity.

1.4.3 Inheritance

Inheritance is a mechanism that defines a new class that inherits the properties and methods of a parent class. One example of this concept in our code is the HMSPersonnel class which serves as the base class for Doctors, Pharmacists, Patients and Admins. They share common attributes like UID, Gender and Email and methods such as getUID(), setEmail() and getGender(). This helps to reduce code duplication.

1.4.4 Polymorphism

Polymorphism means the ability of an object reference to be referred to different types, knowing which method to apply depends on its position in the inheritance hierarchy, which is realized through method overriding.

Polymorphism allows methods to have different implementations based on the object that is invoking them. It enables a single method name to act differently for different subclasses.

Method Overriding:

```
public abstract class MainUI {
    protected abstract void printChoice();
    public abstract void start();
    ...
}

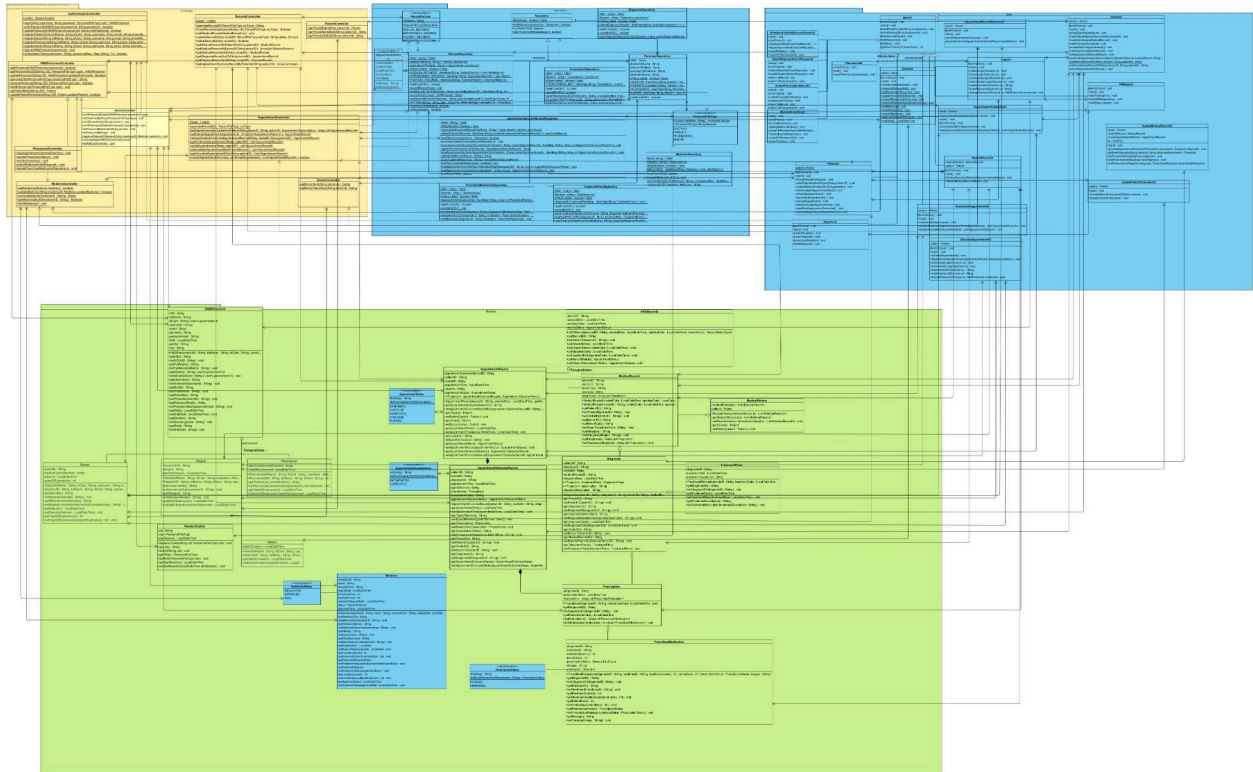
public class PharmacistUI extends MainUI{ {
    @Override
    protected void printChoice() {
        System.out.printf("Welcome! Pharmacist --- %s ---\n")
        ...
    }
    @Override
    public void start() {
        showPharmacistDashboard();
    }
}
```

MainUI is an abstract superclass for all the UI classes. It defines two abstract methods such as printChoice() and start(). All subclasses must implement these abstract methods. This ensures that each subclass provides its specific implementation for these methods, which shows method overriding.

1.5 Design Architectural Pattern

Our system adopts the Model-View-Controller Pattern. The Model includes entity classes that define the data structure and behaviour of the system, such as Doctor, Pharmacist, Patient, and Admin in the package model. The View represents the user interface, displaying data to the user and receiving user input. For example, UI classes in the package view such as LoginUI and AdminUI represent the View. each tailored to a specific functionality. The Controller acts as an intermediary between the Model and the View. It contains the logic that drives the functionalities of the system. For instance, the MedicineController in the package controller manages the medicine-related functionalities.

2. UML Class Diagram



Please refer to the attached class diagrams file for a clearer view.

2.1 Controllers

Controllers are used to manage the logic that handles interactions between different parts of our HMSApp. They act as intermediaries between the Model, View (user interface) and Repository (where the data is stored, in our case, the csv files). In our project, we use controllers to contain methods other than the basic accessors and mutators as well as methods related to repositories, to act as a bridge between View (UI) and Repository, they are used to handle specific actions related to each object and passing data from one object to another. For example, medicineController contains methods related to medicine such as addMedicine, removeMedicine. By keeping business logic in controllers, we avoid cluttering the model with specific application logic and keep the view focused on displaying data. Hence making the code more reusable and flexible as the change in application logic will not affect core data structure (model) or the user interface (view).

2.2 Repositories

The Repository layer in HMSApp is designed to manage data access and storage for key objects, such as Medicine and HMSPersonnel. It encapsulates data operations, allowing our application to load data from and save data to persistent storage (CSV files) efficiently and consistently. For example, the MedicineRepository handles all data management tasks for Medicine objects, including loading existing records from CSV, saving updates, and converting objects to and from CSV format. By centralizing data handling within the Repository layer, we ensure a clear separation of concerns, making the application more modular, maintainable, and adaptable to future storage changes.

2.3 Model Classes

The Model layer in HMSApp represents the core data structures for key entities in the application, encapsulating all relevant properties and behaviors of these entities. Key model classes include HMSPersonnel and HMSRecords, which define the structure and attributes of personnel and record data, respectively.

HMSPersonnel: This class serves as an abstract base class for all personnel in the system and contains common attributes. All other personnel extend this base class and can have their own separate attributes.

HMSRecords: This abstract class serves as a blueprint for various types of records in the system. Concrete subclasses can extend HMSRecords to represent specific record types, inheriting its fundamental properties and adding custom behavior as needed.

The Model classes encapsulate the essential data properties and provide a structured way to interact with data entities within the HMSApp system.

2.4 View (User Interface)

The View layer in HMSApp provides the user interface components that facilitate user interaction with the application. It is responsible for displaying information and collecting user input, serving as the bridge between the user and the underlying application logic. A key

component of the View layer is the abstract class MainUI, which defines common methods for all user interface classes within HMSApp.

Each view UI in HMSApp functions as a Dashboard or Menu specific for the user based on the user's role. These dashboards provide a set of options based on the specific role of the user within the application.

2.5 Relationships and Dependencies

Associations: Associations between controllers, repositories, and model classes. Unidirectional association logic: Controllers know models and repositories, but models and repositories do not know controllers. A controller can access data in the model and repositories, but the model doesn't have any direct knowledge or dependency on the controller. For example, PharmacistController interacts with MedicineRepository to manage medicine inventory. This relation enables the controllers to act as bridges in between other packages, we improve our codes' maintainability and scalability.

Dependencies: Views depend on controllers/ repositories to perform data access and manipulation. For example, UI will call controllers and use methods in controllers to access repositories to obtain the data needed. This dependency is essential, as it ensures that controllers and UIs remain focused on application logic and presentation, while repositories handle data persistence. This separation supports the MVC pattern, improving modularity and maintainability.

Aggregation/Composition: Aggregation indicates that one class contains references to another. e.g., MedicalRecord aggregate Diagnosis (Diagnosis still exists even after MedicalRecord is destroyed) and AppointmentRecord composite AppointmentOutcomeRecord, which composite Prescription. (AppointmentOutcomeRecord no longer exists when AppointmentRecord is destroyed).

3. Reflection

3.1 Difficulties Encountered and Ways to Conquer

During the early stages of designing the system, we encountered many problems, particularly in determining which classes to be implemented and how to set up the repository. The interconnected large number of classes made the design process more complicated. We realised

that a solid design is crucial for our system to have good maintainability, extensibility and readability. We spent a lot of time deciding on the structure of our system.

After careful consideration, we decided to adopt the MVC architectural pattern. It helps organise the codebase such that it has good scalability, maintainability and flexibility. For data storage, we chose to use CSV files as the database and used HashMap to store the data loaded from these files. HashMap offers constant time complexity for get() and put() operations on average, decreasing system waiting time. In addition, HashMap has the advantage of dynamic sizing, allowing it to handle data efficiently.

3.2 Knowledge Learnt

Through this project, we learned how to build a system that applies the SOLID and OOP principles. These principles are important in system design as they ensure reusability, maintainability, and extensibility of the code. In addition, we learned to document our code using Javadoc, which helps others understand the implemented functions and facilitates the extension or modification of codes. Lastly, we learned to import and export files and store the data in HashMap for data access during system runtime. This allows us to manage data without relying on databases, thus offering a lightweight alternative.

3.3 Further Improvements

In the future, we aim to extend the system to support multiple users accessing it simultaneously, and with real-time updates after each user performs an operation. Another improvement will be to create a graphical user interface using HTML, CSS and JavaScript. This will improve user experience by providing a more interactive and user-friendly platform for users. Additionally, we plan to use database for storing the data as it provides advanced querying capabilities and features like data consistency which allows efficient storage and retrieval of data.






Appendix:

Declaration of Original Work for SC2002/CE2002/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 honored 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 awarded for the assessed work. In addition, disciplinary actions may be taken.

Name	Course (SC2002/CE2002 CZ2002)	Lab Group	Signature /Date
CHONG KWANG CHEN	SC2002	SCSI	 15/11/2024
HNG CHERNG KHAI	SC2002	SCSI	 15/11/2024
LIU YANZHI	SC2002	SCSI	 15/11/2024
TAN SHI QI	SC2002	SCSI	 15/11/2024
YAP MEI YEE	SC2002	SCSI	 15/11/2024

Appendix I: Declaration on Use of GAI (Generative Artificial Intelligence) Assistance in relation to Assignment/Project (to be submitted individually even for group projects)

I Tan Shi Qi (student name),
C230057 @e.ntu.edu.sg (NTU email) honestly and sincerely make the following declaration in relation to the following course submission:

1. Name of course: Object-Oriented Design and Programming
2. Course Code: SC2002
3. Instructor: Dr. Zhang Jie, Dr. Li Fang
4. Title of Assignment/Project Submission: Hospital Management System

In relation to the foregoing I hereby declare that, fully and properly in accordance with the Assignment/Project Instructions I have (check where appropriate):

- i. Used GAI as permitted to assist in generating key ideas only. ☐
- ii. Used GAI as permitted to assist in generating a first text only. ☐
- And/or
- iii. Used GAI to refine syntax and grammar for correct language submission only. ☒

Or

- iv. As it is not permitted: Not used GAI assistance in any way in the development or generation of this assignment or project. ☐

I also declare that I have :

- a. Fully and honestly submitted the digital paper trail required under the assignment/project instructions; and that
- b. Wherever GAI assistance has been employed in the submission in word or paraphrase or inclusion of a significant idea or fact suggested by the GAI assistant, I have acknowledged this by a footnote; and that,
- c. Apart from the foregoing notices, the submission is wholly my own work.

TAN SHI QI 
Student Name & Signature

15/11/2024
Date

Appendix I: Declaration on Use of GAI (Generative Artificial Intelligence) Assistance in relation to Assignment/Project (to be submitted individually even for group projects)

I Chong Kwang Chen (student name),
KCHONG044 @e.ntu.edu.sg (NTU email) honestly and sincerely make the following declaration in relation to the following course submission:

1. Name of course: Object-Oriented Design Programming
2. Course Code: SC2002
3. Instructor: Dr. Zhang Jie, Dr Li Fang
4. Title of Assignment/Project Submission: Hospital Management System

In relation to the foregoing I hereby declare that, fully and properly in accordance with the Assignment/Project Instructions I have (check where appropriate):

- i. Used GAI as permitted to assist in generating key ideas only. ☐
 - ii. Used GAI as permitted to assist in generating a first text only. ☐
- And/or
- iii. Used GAI to refine syntax and grammar for correct language submission only. ☒

Or

- iv. As it is not permitted: Not used GAI assistance in any way in the development or generation of this assignment or project. ☐

I also declare that I have :

- a. Fully and honestly submitted the digital paper trail required under the assignment/project instructions; and that
- b. Wherever GAI assistance has been employed in the submission in word or paraphrase or inclusion of a significant idea or fact suggested by the GAI assistant, I have acknowledged this by a footnote; and that,
- c. Apart from the foregoing notices, the submission is wholly my own work.



Chong Kwang Chen

Student Name & Signature

15/11/2024

Date

Appendix I: Declaration on Use of GAI (Generative Artificial Intelligence) Assistance in relation to Assignment/Project (to be submitted individually even for group projects)

I Liu Yanzhi (student name),
liuy0251@e.ntu.edu.sg (NTU email) honestly and sincerely make the following declaration in relation to the following course submission:

1. Name of course: Object-Oriented Design Programming
2. Course Code: SC2002
3. Instructor: Dr. Zhang Jie, Dr. Li Fang
4. Title of Assignment/Project Submission: Hospital Management System

In relation to the foregoing I hereby declare that, fully and properly in accordance with the Assignment/Project Instructions I have (check where appropriate):

- i. Used GAI as permitted to assist in generating key ideas only. ☐
- ii. Used GAI as permitted to assist in generating a first text only. ☐
- And/or
- iii. Used GAI to refine syntax and grammar for correct language submission only. ☒

Or

- iv. As it is not permitted: Not used GAI assistance in any way in the development or generation of this assignment or project. ☐

I also declare that I have :

- a. Fully and honestly submitted the digital paper trail required under the assignment/project instructions; and that
- b. Wherever GAI assistance has been employed in the submission in word or paraphrase or inclusion of a significant idea or fact suggested by the GAI assistant, I have acknowledged this by a footnote; and that,
- c. Apart from the foregoing notices, the submission is wholly my own work.

Liu Yanzhi 
Student Name & Signature

15/11/2024
Date

Appendix I: Declaration on Use of GAI (Generative Artificial Intelligence) Assistance in relation to Assignment/Project (to be submitted individually even for group projects)

I Hng Cherng Khai (student name),
C230077@e.ntu.edu.sg (NTU email) honestly and sincerely make the following declaration in relation to the following course submission:

1. Name of course: Object-Oriented Programming
2. Course Code: SC2002
3. Instructor: Dr Zhang Jie, Dr Li Fang
4. Title of Assignment/Project Submission: Hospital Management System

In relation to the foregoing I hereby declare that, fully and properly in accordance with the Assignment/Project Instructions I have (check where appropriate):

- i. Used GAI as permitted to assist in generating key ideas only. ☐
- ii. Used GAI as permitted to assist in generating a first text only. ☐

And/or

- iii. Used GAI to refine syntax and grammar for correct language submission only. ☒

Or

- iv. As it is not permitted: Not used GAI assistance in any way in the development or generation of this assignment or project. ☐

I also declare that I have :

- a. Fully and honestly submitted the digital paper trail required under the assignment/project instructions; and that
- b. Wherever GAI assistance has been employed in the submission in word or paraphrase or inclusion of a significant idea or fact suggested by the GAI assistant, I have acknowledged this by a footnote; and that,
- c. Apart from the foregoing notices, the submission is wholly my own work.



Hng Cherng Khai

Student Name & Signature

15/11/2024

Date

Appendix I: Declaration on Use of GAI (Generative Artificial Intelligence) Assistance in relation to Assignment/Project (to be submitted individually even for group projects)

I Yap Mei Yee (student name),
E230150 @e.ntu.edu.sg (NTU email) honestly and sincerely make the following declaration in relation to the following course submission:

1. Name of course: Object Oriented Programming
2. Course Code: SC2002
3. Instructor: Dr Zhang Jie, Dr Li Fang
4. Title of Assignment/Project Submission: Hospital Management System

In relation to the foregoing I hereby declare that, fully and properly in accordance with the Assignment/Project Instructions I have (check where appropriate):

- i. Used GAI as permitted to assist in generating key ideas only. ☐
- ii. Used GAI as permitted to assist in generating a first text only. ☐
- And/or
- iii. Used GAI to refine syntax and grammar for correct language submission only. ☒

Or

- iv. As it is not permitted: Not used GAI assistance in any way in the development or generation of this assignment or project. ☐

I also declare that I have :

- a. Fully and honestly submitted the digital paper trail required under the assignment/project instructions; and that
- b. Wherever GAI assistance has been employed in the submission in word or paraphrase or inclusion of a significant idea or fact suggested by the GAI assistant, I have acknowledged this by a footnote; and that,
- c. Apart from the foregoing notices, the submission is wholly my own work.

Yap Mei Yee



Student Name & Signature

15/11/2024

Date

Appendix: Member's work contribution and distribution breakdown

1. Tan Shi Qi

- Admin Functionalities
- Report
- UML Class Diagram
- Live Demo (Admin Part)

2. Liu Yanzhi

- Pharmacist Functionalities
- Report
- UML Class Diagram
- Live Demo (Pharmacist Part)

3. Chong Kwang Chen

- Repository Functionalities
- Login and Registration Functionalities
- GitHub Branch Management
- UML Class Diagram
- Live Demo (MVC Code Structure)

4. Hng Cherng Khai

- Repository Functionalities
- Doctor Functionalities
- Patient Functionalities
- UML Class Diagram
- Model Design
- Live Demo (Doctor/Patient Part)

5. Yap Mei Yee

- Doctor Functionalities
- Patient Functionalities

- Repository Functionalies (appointmentOutcomeRecord)
- UML Class Diagram
- Live Demo (Doctor/Patient Part)