**SC2002:**

***A logo with a lion on it

Description automatically generated*Object-Oriented Design Analysis & Programming**

Github:

Etc:

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Course** | **Lab Group** | **Signature / Date** |
| LIEW WEI JIE | SC2002 | SCED | / 15-11-2024 |
| LIM ZU LIANG | SC2002 | SCED | / 15-11-2024 |
| JACOB ONG JIA CHUN | SC2002 | SCED | / 15-11-2024 |
| TAN YI XIANG, DYLAN | SC2002 | SCED | / 15-11-2024 |

**Table of Contents**

[**Chapter 1. Design Considerations** 3](#_Toc182480886)

[**Chapter 1.1. subheading1** 3](#_Toc182480887)

[**Chapter 1.2. subheading2** 3](#_Toc182480888)

[**Chapter 2. Unified Modelling Language Diagram** 4](#_Toc182480889)

[**Chapter 3. Additional Features** 5](#_Toc182480890)

[**Chapter 4. Functional Tests** 6](#_Toc182480891)

[**Chapter 4.1. Patient Actions** 6](#_Toc182480892)

[**Chapter 4.2. Doctor Actions** 7](#_Toc182480893)

[**Chapter 4.3. Pharmacist Actions** 8](#_Toc182480894)

[**Chapter 4.4. Administrator Actions** 9](#_Toc182480895)

[**Chapter 4.5. Login System & Password Management** 9](#_Toc182480896)

[**Chapter 5. Reflecting on our Voyage** 10](#_Toc182480897)

[**Chapter 5.1. Polymorphism of Challenges** 10](#_Toc182480898)

[**Chapter 5.2. Refactoring Challenges in Real-Time** 11](#_Toc182480899)

[**Chapter 5.3. Committing Knowledge Gained to Brain Repository** 13](#_Toc182480900)

## **Chapter 1. Design Considerations**

### **Chapter 1.1. subheading1**

Ijij

### **Chapter 1.2. subheading2**

Ijij

## **Chapter 2. Unified Modelling Language Diagram**

## **Chapter 3. Additional Features & Enhancements**

**General**

|  |
| --- |
| 1. **Forget Password Request:** We added an option for users to request a password reset to default, pending Administrator approval. 2. **Password Decoupling and Hashing:** User passwords are securely hashed and stored separately from other personal information to enhance data protection. 3. **User-Friendly Interface:** Information is sorted and displayed neatly (typically in tables), and we allow users to make selections by entering an index (e.g., 1 or 2), simplifying navigation. 4. **Aesthetically Pleasing Interface:** Our team enhanced the program's appearance with features like styled headings, role-specific colours, and more for an improved user experience. 5. **Stringent Validation:** Ensures accurate input (e.g., valid data formats) and enforces logic (e.g., prevents dispensing medicine if stock is insufficient). |

**Patient**

|  |
| --- |
| 1. **Registration:** First-time users can easily register themselves as patients within our program. 2. **Rate Doctors:** After their appointments, patients can rate and provide feedback on doctors, enhancing care quality and sharing valuable experiences with others. 3. **View and Pay Bills:** Patients can view and pay their medical bills directly through the system. |

**Doctor**

|  |
| --- |
| **View Patient Ratings:** Doctors can view patient ratings, including an average score, to assess their performance and gauge patient satisfaction, helping them identify areas for improvement. |

**Administrator**

|  |
| --- |
| 1. **Manage Forgot Password Requests:** Administrators can approve or deny password reset requests for users who have forgotten their credentials. 2. **Bill Patients:** After each appointment, Administrators can bill patients accordingly for services rendered and medicine prescribed. |

## **Chapter 4. Functional Tests**

### **Chapter 4.1. Patient Actions**

Patient Main Menu Test Case 1: View Medical Record

|  |  |
| --- | --- |
|  |  |

Test Case 2: Update Personal Information Test Case 3: View Available Appointment Slots

|  |  |
| --- | --- |
|  |  |

Test Case 4: Schedule an Appointment

|  |
| --- |
|  |

Test Case 5: Reschedule an Appointment

|  |
| --- |
|  |

Test Case 6: View Available Appointment Slots

|  |
| --- |
|  |

Test Case 7: View Scheduled Appointments Test Case 8: View Past Appointment Outcome Records

|  |  |
| --- | --- |
|  |  |

### **Chapter 4.2. Doctor Actions**

Doctor Main Menu Test Case 9: View Patient Medical Records

|  |  |
| --- | --- |
|  |  |

Test Case 10: Update Patient Medical Records

|  |
| --- |
|  |

Test Case 11: View Personal Schedule Test Case 12: Set Availability for Appointments

|  |  |
| --- | --- |
|  |  |

Test Case 13: Accept or Decline Appointment Requests

|  |
| --- |
|  |

Test Case 14: View Upcoming Appointments

|  |
| --- |
| A black screen with white text  Description automatically generated |

Test Case 15: Record Appointment Outcome

|  |
| --- |
|  |

### **Chapter 4.3. Pharmacist Actions**

Pharmacist Main Menu Test Case 17: Update Prescription Status

|  |  |
| --- | --- |
|  |  |

Test Case 16: View Appointment Outcome Record

|  |
| --- |
|  |

Test Case 18: View Medication Inventory

|  |
| --- |
|  |

Test Case 19: Submit Replenishment Request

|  |
| --- |
| A screenshot of a computer screen  Description automatically generated |

### **Chapter 4.4. Administrator Actions**

Administrator Main Menu Test Case 20: View and Manage Hospital Staff

|  |  |
| --- | --- |
|  |  |

Test Case 21: View Appointments Details

|  |
| --- |
|  |

Test Case 22: View and Manage Medication Inventory

|  |
| --- |
| A screenshot of a computer program  Description automatically generatedA screenshot of a computer program  Description automatically generated |

Test Case 23: Approve Replenishment Requests

|  |
| --- |
|  |

### **Chapter 4.5. Login System & Password Management**

Starting the Program Exiting the Program

|  |  |
| --- | --- |
|  |  |

Test Case 24: First-Time Login and Password Change Test Case 25: Login with Incorrect Credentials

|  |  |
| --- | --- |
|  |  |

## **Chapter 5. Reflecting on our Voyage**

### **Chapter 5.1. Polymorphism of Challenges**

#### Chapter 5.1.1. Upcasting Theory into Practice

Now that we’re reaching the end of our journey in SC2002 Object-Oriented Design & Programming, it's clear this project has been nothing short of a wild and bumpy ride. One of the largest hurdles our team faced was translating the course content—concepts like Polymorphism, Decoupling, and the SOLID principles—into functional elements of our project.

On paper, these ideas seemed logical and manageable but applying them in practice revealed layers of unexpected complexity. What initially appeared to be straightforward connections quickly unravelled, presenting roadblocks at every turn, and pushing us to troubleshoot and rethink our approach.

The theories that had seemed clear during lectures demanded a far greater level of precision and foresight than we anticipated. Implementing design patterns in real-time was not merely a matter of “fitting the pieces together.” Each piece was crafted in relation to the whole, carefully balanced to prevent unneeded dependencies and preserve the architecture of the code.

#### Chapter 5.1.2. Concurrency Collisions & Member Overloading

Taking on a project of this scale—with our team’s limited experience—turned out to be more of a juggling act than we anticipated. Coordinating all the different components was tough, especially for such an interconnected project with multiple roles like “Patient” and “Doctor”, along with classes that were heavily associated on one another. Our decision to push for quality did not make things any easier; we were determined to follow the course principles by decoupling classes as much as possible and applying the Boundary-Control-Entity (BCE) model. Again, in theory, it sounded great; in practice, it was excruciatingly challenging.

To add to the mix, our team members are all overloading this semester (taking on more credits than usual) with SC2006 Software Engineering, another project-heavy course where we have to develop a website. On top of managing these, we were juggling tests, presentations, studying for other modules, and our own personal commitments.

The sheer scope of everything, along with our conflicting schedules, pushed us to our limits. Most nights ended with us still typing away in the early morning hours, trying to keep our progress aligned, troubleshoot issues, and wrap up tasks. The constant pressure of deadlines forced us to carefully manage our time and communication just to keep our heads above water.

#### Chapter 5.1.3. Null Pointer Teammate

Amidst all the stress and project chaos, we also found ourselves dealing with a “Null Pointer” teammate—someone who essentially returned nothing (read: unresponsive and zero contribution). With limited project experience, our team had no idea how to approach this issue. It felt like we were navigating a storm on a makeshift raft, with this teammate acting as a hole in the hull, constantly dragging us down. To make matters worse, we knew we are graded as a team of five, which only added to our frustration as we tried to figure out a resolution.

### **Chapter 5.2. Refactoring Challenges in Real-Time**

#### Chapter 5.2.1. Debugging the Group Dynamics

To keep our group running smoothly, we knew we had to address our teamwork issues directly—and as early as possible. We began by establishing clearer communication channels and setting a structured schedule for regular updates, including bi-weekly labs and weekly tutorials. These meetings helped us synchronize our progress and set more realistic goals. Each session became an opportunity to realign tasks, clarify misunderstandings, and ensure everyone felt included and understood the plan moving forward.

With a mutual understanding of the heavy workload each of us was facing, our team agreed to keep communication open. This meant we would seek assistance early when needed, be upfront about our availability, and rotate who took the lead on different tasks. By sharing the load this way, we managed (for the most part) to stay on track and support each other when things got especially hectic. While not perfect, these adjustments helped us find a working rhythm, making it easier to tackle tasks and reducing the friction in our teamwork and schedules.

#### Chapter 5.2.2. Iterating towards the Solution

Throughout the project, we made a conscious effort to stay fully engaged, avoiding the trap of simply going through the motions. Completing this project required more than just attendance; we needed to actively contribute and stay focused on the bigger picture. Embracing an Agile mindset, we kept everyone in the loop through regular check-ins, where we updated each other, tackled hurdles, and shared ideas openly.

This approach allowed us to iterate on our solutions more effectively. When issues arose, we brainstormed as a team, learning from each attempt and refining our approach until we found workable solutions. Agile principles helped us stay flexible and adapt quickly, adjust priorities as needed, and remain aligned as a team. By keeping communication open and focusing on continuous improvement, we managed to push forward with each iteration, even when the challenges and workload seemed overwhelming.

#### Chapter 5.2.3. Utilizing Super Methods

To address the “Null Pointer” teammate issue, we knew we had to make use of all available resources—our T.A. and professor became our “super methods.” First, we consulted our T.A., who recommended escalating the matter to the professor. With the professor’s guidance and support on the administrative side, we were able to resolve the situation and ultimately remove the teammate from our group.

With this adjustment, our team of four became a more efficient, well-oiled machine. This experience showed us the value of reaching out for support when handling challenging situations, allowing us to focus our energy on moving forward as a stronger, more cohesive team.

### **Chapter 5.3. Committing Knowledge Gained to Brain Repository**

#### Chapter 5.3.1. Inheriting Technical Expertise

This project took our technical knowledge from theory to practice, embedding skills in our “brain repository” that we will carry forward. Concepts like polymorphism, SOLID principles, and the Boundary-Control-Entity model transformed from abstract ideas into real, functional tools. Applying these principles taught us the nuances of design patterns, debugging, and structuring code thoughtfully. The hands-on experience revealed complexities we hadn’t anticipated, but each challenge sharpened our problem-solving skills, reinforcing our foundation in a way that classroom learning alone couldn’t achieve.

#### Chapter 5.3.2. Forking Project Management and Collaboration Skills

On the collaborative side, we gained practical project management skills that will serve us well beyond this course. Working as a team meant learning to communicate effectively, respect different schedules, and distribute tasks according to each member’s strengths. Each step of the way, we honed skills like setting clear goals, adjusting to setbacks, and maintaining accountability. These experiences taught us how to “fork” project management techniques into our workflow, making collaboration smoother and reinforcing our ability to navigate the demands of team projects.

#### Chapter 5.3.3. Refactoring Adaptability and Resilience

The most valuable takeaway was in refactoring our mindset to be more adaptable and resilient. This project was filled with unexpected challenges, and each one demanded a fresh perspective and a willingness to adjust. Adopting a flexible approach allowed us to stay focused despite the constant demands and shifting priorities. Building resilience was essential; we learned to handle setbacks with a problem-solving mindset, seeing each obstacle as an opportunity for growth. By the end, we had grown more resilient, prepared to tackle future projects with a steady approach and a readiness to adapt as needed.

**End**