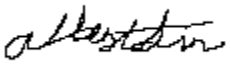
Declaration of Original Work

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
Lin Lihong Albert	SC2002	SCE1	 07/11/2025
Terashima Shuma	SC2002	SCE1	 18/11/2025
Chaware Tanvi Pankaj	SC2002	SCE1	 18/11/2025
Ng Jean Tzi, Edric	SC2002	SCE1	 18/11/25

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC2002 OBJECT ORIENTED DESIGN & PROGRAMMING

Lab Assignment Deliverables

Lab Group	SCE1
Assignment Group	1
Members	Lin Lihong Albert
	Terashima Shuma
	Chaware Tanvi Pankaj
	Ng Jean Tzi, Edric

Table of contents:

1. Design Considerations....4

1.1 Approach Taken....4

1.2 OO Principles Used....4

Inheritance & Polymorphism....4

1.3 Design Patterns / Extensibility....5

2. UML Class diagram....6

3. UML Sequence Diagram....7

4. Test Case Demonstration....7

5. Reflection....10

5.1 Difficulties Encountered....10

5.2 Key Insights from the Development Process (and the knowledge learnt from this course)....10

5.3 Future Improvements....11

1. Design Considerations

1.1 Approach Taken

The code was written with minimal repetition in mind (Don't Repeat Yourself). Following the SOLID framework, the goal was to minimize repeated code, and reduce the size of each method, especially if its functionality could be split into more sub-methods.

1.2 OO Principles Used

Inheritance & Polymorphism

All user types (Student, CompanyRep, CareerCenterStaff) inherit from a common User superclass.

This allows:

- Shared authentication and credential handling
- Specialized behaviours (e.g., CompanyRep requires approval to log in)

Polymorphism allows future system controllers to interact with users independently of their concrete type (e.g., User loggedInUser = login(...)).

Solid Principles:

S - Even for more complex methods, the code is split into the roles that each actor (Company / CompanyRep etc.) would play. For example, CompanyRep has a function createInternship(). Instead of taking in the parameters on top of doing the Company's role of checking if the number of internships exceed the max number, CompanyRep only does the preliminary check to make sure all parameters are valid, before passing it onto Company, which will handle the actual creation.

O - The User class is used as a template for all derived child classes. It is closed to modification, and child classes are to implement their own version of functions.

L - Each User subclass inherits from User, hence casting these classes to a generic User type would allow for the same behaviour as interacting directly with a User class.

I - Each class has a specific function, there is a separation of specific methods. For example, the CompanyRep and Student classes both derive from the same User class, but are split as they serve different functions.

D - The User class acts as an interface for all relevant “Persons” in this code (i.e. CompanyRep, Student, CareerCenterStaff). Hence, it does not depend on any of these classes that inherit the methods and attributes. The CompanyRep also does not handle the modification of Internship, as that is the role of the Company. This dependency is only uni-directional.

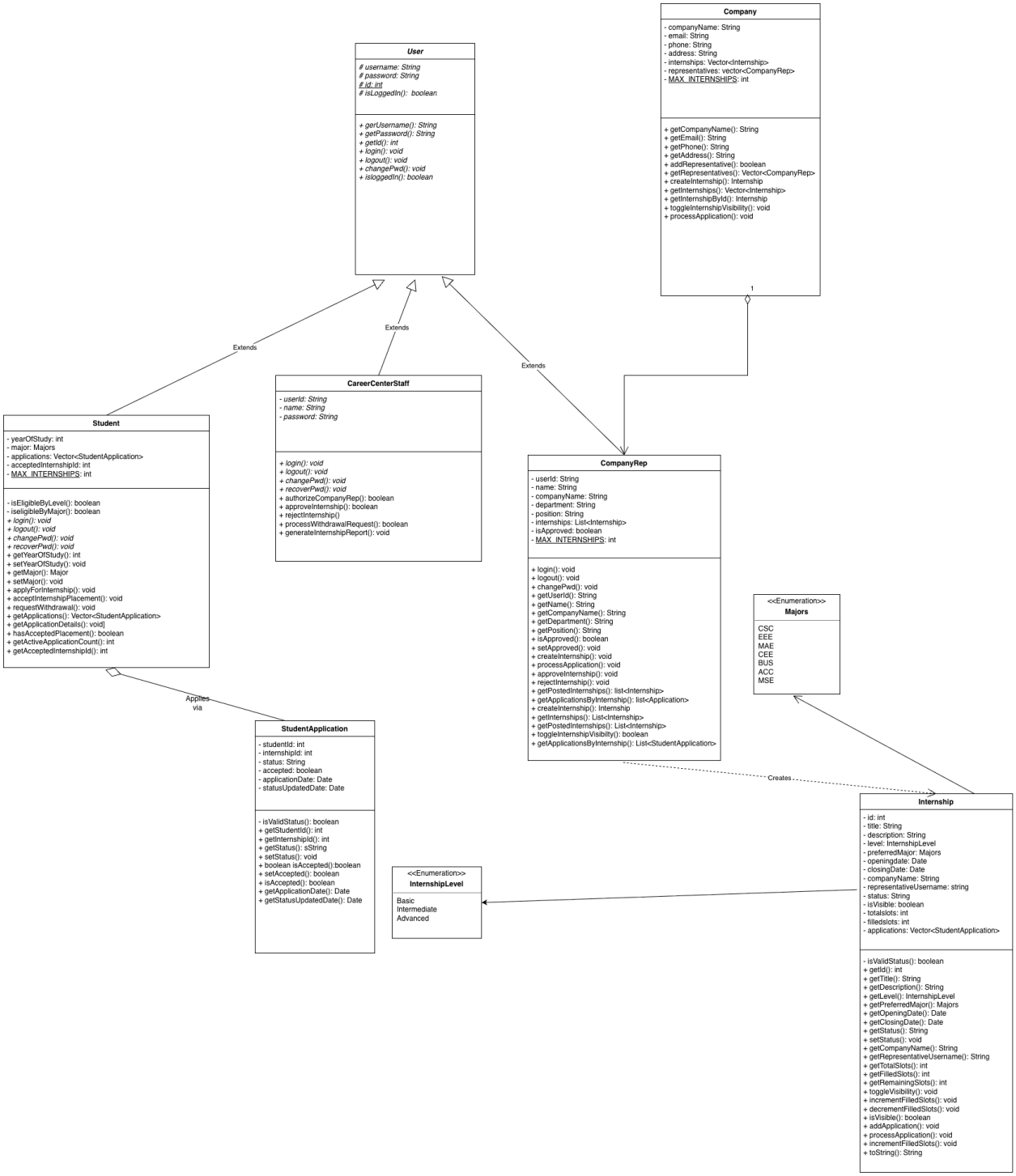
1.3 Design Patterns / Extensibility

The design intentionally supports extensibility:

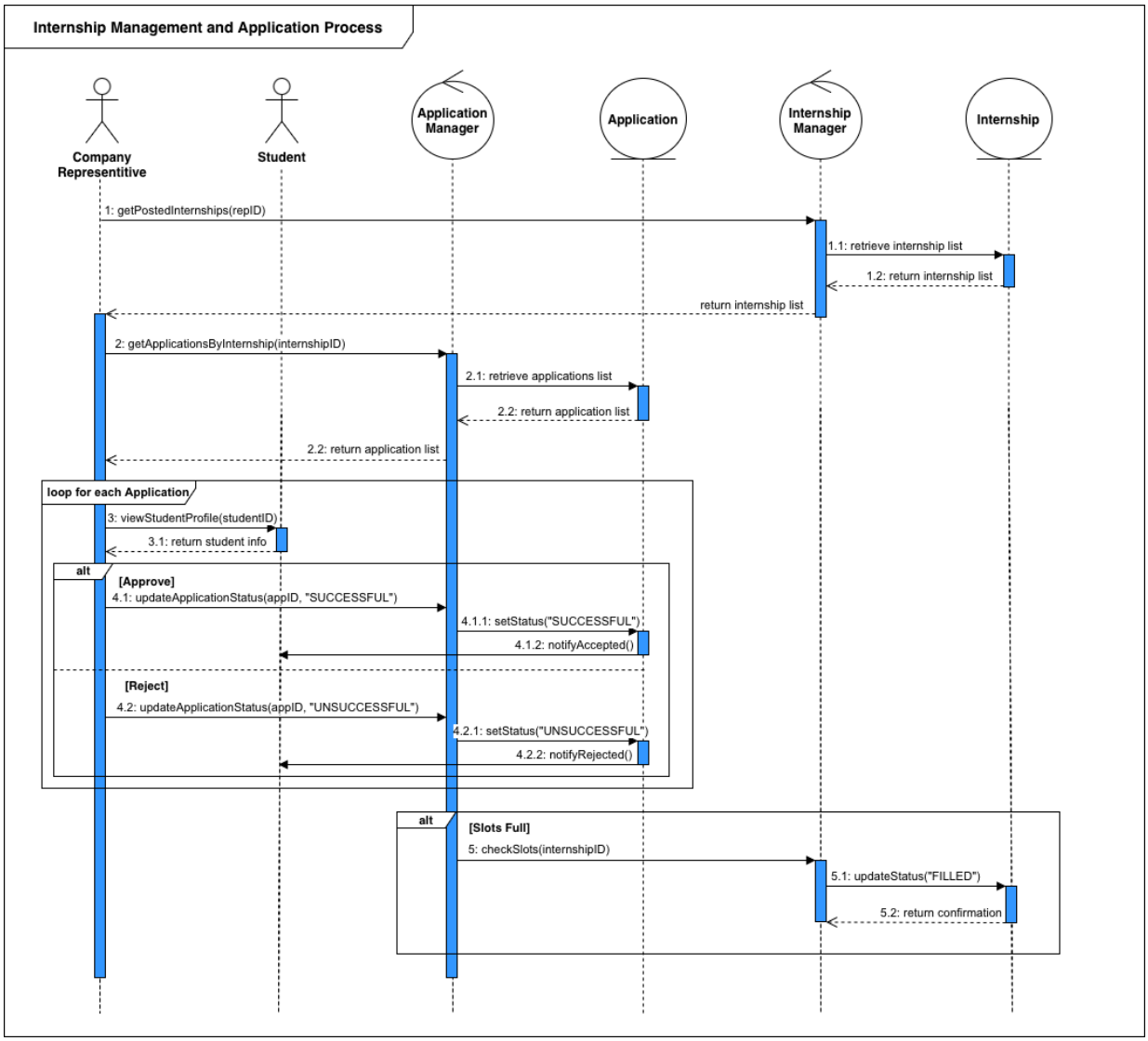
- **Adding new roles**
    - New user types can easily extend the User superclass.
- **Adding new application statuses or business rules**
    - Status changes are centralized in StudentApplication, and internship availability logic is encapsulated in Internship, making rule modifications localized.
- **Introducing persistence**
    - Since objects are serializable and domain logic is self-contained, a file-based or database-backed repository layer can be added with minimal changes.
- **Modifying filtering and reporting logic**
    - The generateReport method can be expanded into a dedicated ReportService without affecting the internship or user classes.

The modular structure and low coupling between major components simplify future updates and testing, and help maintain system stability as requirements evolve.

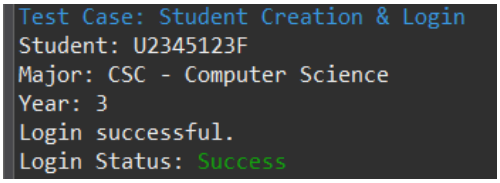
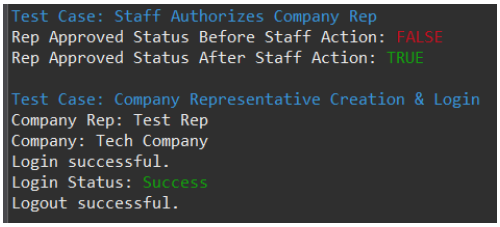
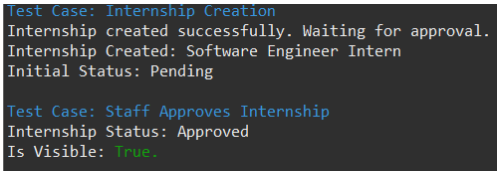
2. UML Class diagram

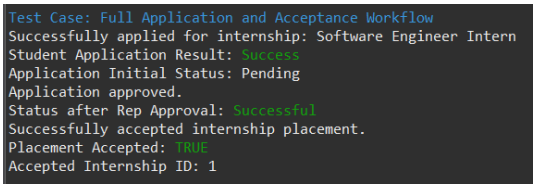
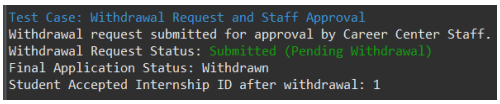


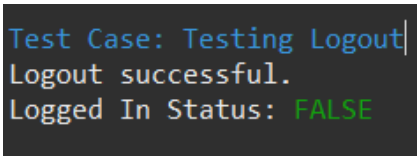
3. UML Sequence Diagram



4. Test Case Demonstration

Test Case	Expected Outcome	Results
1. Student creation & login (Basic functionality)	The student should log in successfully using valid credentials. The system should display the student's profile correctly (with correct major & year of study)	
2. Company Rep approval requirement & login	A newly created CompanyRep account should be unable to login until its account has been approved by staff. When approved, isApproved status is TRUE CompanyRep can login	
3. Internship Creation & Initial Status	A CompanyRep should successfully create an internship, and the system will default its initial status as "Pending" and waiting for approval. Once the staff approves the internship its status will be changed to "Approved" and isVisible will be TRUE.	

4. Full application & Placement Acceptance	When a student successfully applies, the CompanyRep will then approve their application based on the eligibility of the internship and student year of study and then the student can accept the placement. Acceptance will then set the placement ID and change the application status to "Successful"	
5. Withdrawal Request & Staff Approval	The student can submit a withdrawal request, changing their application status to "Pending Withdrawal" (awaiting staff action). Staff will then finalise this action, thus setting the application status to "Withdrawn"	
6. Staff Report Generation & Filtering	The system should successfully apply filters such as (eg, Major = CSC, Level = INTERMEDIATE) and display the matching internship data accurately including its current status and slot information.	

7. User Logout	User executes logout() function. The system must confirm logout success and change user's logged in status from TRUE to FALSE.	

5. Reflection

5.1 Difficulties Encountered

Even as we worked out most of the deliverables efficiently based on the initial plans, there was somewhat limited communication among us, which led to bottlenecks in receiving real-time updates and critical information. Moreover, there were occasions when some of us were relatively busy, which somehow led to an unequal distribution of workload. That was when one of us took the initiative to proactively check in on the progress through a chat platform and agreed on the specific actionables required to be done within a given timeframe or as soon as possible. In addition, we could have made full use of GitHub's issues feature, where each issue raised there can be kept in view for the timely implementation of missing features.

5.2 Key Insights from the Development Process (and the knowledge learnt from this course)

The development process was more than just teamwork - we had to apply the SOLID design principles into our code, primarily the Single Responsibility Principle, to ensure the ease of maintainability and modularity of the program. In addition, we also applied the fundamentals of class and sequence diagrams, as well as used Visual Paradigm to convert Java code into the appropriate class diagrams for ease of reference when creating one. We also learnt about the

importance of using task tracking services like GitHub to keep track of the tasks completed, essentially a digital “paper trail”.

5.3 Future Improvements

What we feel we could improve on is to make use of a software development methodology, such as AGILE. Besides starting earlier, we could practice holding weekly Stand-Up Meetings, allowing each of us to get up to speed with our development plans, as well as planning actionables meant to be done before the next meeting. This way, it aims to minimise any potential bottlenecks that we might encounter when developing the project, as well as ensure everyone has a fair share of responsibilities. We could have also used enhanced functionalities of modern Java versions where appropriate, such that our code can be even more readable and concise. This can range from lambda functions, to enhanced switch statements, and so on.

As for a future Improvements, the system could be enhanced in several ways if extended:

- **Add persistence** (file or database storage) so that internships and applications can be saved and restored across executions.
- **Provide a graphical or web-based interface** to support more realistic multi-user interaction.
- **Extend reporting capabilities** to include additional filters such as company, date ranges, or slot availability.

Link to Github: https://github.com/Albert-Lin227/SC2002-Grp_Project.git