ASSIGNMENT SC2002: OBJECT-ORIENTED DESIGN & PROGRAMMING

BUILDING AN OO APPLICATION (FYPMS)

LAB: A50 GROUP: 2

MEMBERS:

BRYAN CHIA SONG NIAN (U2222849J)

PHUN WEI CHENG RUSSELL (U2120850J)

SIEW JIA YANG (U2022366L)

Project Demonstration Video: https://youtu.be/AA8NXlhY16o

GitHub Main Page: https://github.com/l3ryanchia/proj2002

Clearer UML Class Diagram:
https://github.com/l3ryanchia/proj2002/blob/master/A50-grp2-UML.jpg

Report Overview

Final Year Project Management System (FYPMS) is an application for staff and students to manage FYP. Our report details the design process of building this application, developed using the Java language. The application design was modelled using the ECB (Entity - Control - Boundary) pattern, integrated with SOLID principles and OO concepts.

Code Structure

Entity-Control-Boundary (ECB) Pattern

The ECB pattern is a variation of the Model-View-Controller (MVC) pattern, where Entities are objects that represent the system data, Controllers orchestrate the flow of information and coordinates the interaction between the boundary and the database, while Boundaries interface with the user, collecting input and sending commands to the rest of the application.

Programs (Boundaries)

In the package named 'programs', it contains all the applications that manage the information between user and managers. FYPMSApp is the main user interface for the application.
ReadCSV - Reads data from input CSV files and calls methods in serializers for initialisation
StudentMenu - Accessed by student users. Student users are able to view all available projects if he has not registered a project, for which he can select a project and request for it to be allocated to him. He can view his registered project once registration is done, request for a change in title, request for deregistration of FYP, view his request history, and change password.
SupervisorMenu - Accessed by supervisor users. A supervisor can create many projects, one at a time, view the information of all of his projects, modify project title, request to transfer a student, view all his pending requests and approve or reject them, view requests' history and change his password.
FYPCoordinatorMenu - Used by the FYP Coordinator. The coordinator can view projects by status, generate project details report, view pending requests, view request history, and change password. The FYP coordinator would also be able to access the SupervisorMenu as he/she must also be a supervisor, however, if he/she logs in as a supervisor, he/she will not be able to perform FYP coordinator related tasks such as allocating of projects.

Managers (Control)

Managers act as controllers, handling all the main logic of the application. The FYPMSApp will call on the different manager files to handle different aspects of the FYP Management System, such as projects, supervisors, students and requests. It extracts the required information from the database to be returned to the boundary.

ProjectManager - Manages the projects in the FYP Management System, including creating, updating, deleting and querying projects. It also handles filtering and displaying projects based on certain criteria.

SupervisorManager - Manages the supervisors, including creating, updating, deleting and querying supervisor information.

StudentManager - Manages the students, including creating, updating, deleting and querying student information.

RequestManager - Manages the requests sent by students or supervisors, including creating, updating, querying request information and displaying request history.

Models (Entity)

Models represent the different entities involved in the FYP Management System, such as projects, supervisors, coordinator, students and requests which store various information. Models are created as objects and accessed by managers.

Project - Represents a final year project, containing information like project ID, title, description, supervisor and status.

Student - Represents a student, containing information like user ID, name, email, password, project ID and status.

Request - Represents a request sent by students or supervisors, containing information like request ID, type, sender ID, receiver ID and status.

Serializers

Serializers read data from files and create corresponding manager objects. Each serializer is focused on a specific type of data (FYPCoordinator, Project, Student and Supervisor). For example, ProjectSerializer - Reads project data from the csv file and creates a ‘ProjectManager’ object. For each project in the file, it adds a new project to the ‘ProjectManager’ by providing the title and supervisor information.

Design Principles Used

Single Responsibility Principle (SRP)

The Single Responsibility Principle (SRP) states that a class should have only one reason to change, hence focusing on a single responsibility. The FYPMS app adheres to the SRP by separating concerns into different classes, each of which focuses on a specific responsibility. For example, the FYPMSApp class serves as the main entry point and is responsible for coordinating the flow of the app. It initialises the interactions between various components like managers, serializers and menu-driven interfaces. It does not handle the implementation details of specific components, hence sticking to the SRP. For each of the menu options selected by the user, a separate function would be called to execute the operation, ensuring that there is no “master” or “god” class that handles the entire array of multiple tasks. In addition, the processing of data from the CSV files are also implemented in a separate class ReadCSV that calls respective functions within the Serialisers to initialise databases before the main program is run in FYPMSApp.

Another example would be the generateProjectDetails function under FYPCoordinatorMenu, which allows the user to apply filters to the project list before displaying the final report. Instead of incorporating the filtering and display into the same method, each filter being applied has their own method under ProjectSupervisor, which will return a filtered list to be further filtered if required, while the displayProjects method would be only responsible for printing out the final list. Not only does this adhere to the SRP, it also promotes extensibility and modularity of the application, which we have demonstrated by creating an additional filterByStudents function to show how easily new filters can be added. If any of the filters are not working, we are also able to isolate and test the specific method without modifying the other methods which can continue to work correctly, making the code easier to maintain and adaptable to changing requirements.

Open-Closed Principle (OCP)

The Open-Closed Principle (OCP) states that software entities should be open for extension but closed for modification, meaning that the design of a class should allow for new functionalities to be added without the modification of existing code. In the FYPMS App, OCP is applied. For example, the use of inheritance and polymorphism in the Request class hierarchy allows for extending the system with new types of requests like Req_AllocateProj, without modifying the existing abstract Request superclass. The superclass is closed for modification, but provides methods that are meant to be implemented or overridden in any subclasses that are created. For example, when a new request type is designed, the programmer can simply specify the actions to be triggered upon creation, approval and rejection of the request, and if there is any additional information to be displayed, there is an optional protected displayAdditionalInfo method provided. Without modifying the original template format of requests display, the subclass can simply add additional lines of information, such as the suggested new title for Req_ChangeTitle or the suggested new supervisor ID for Req_TransferStudent. This allows for slight extensions of the fixed structure when it is called internally by the displayRequest method.

Liskov- Substitution Principle (LSP)

The Liskov - Substitution Principle (LSP) states that objects of a derived class should be able to replace objects of the base class without affecting the correctness of the program. This principle ensures class and its derived classes have a consistent behaviour, promoting maintainability and reusability. For example, the User interface is implemented by the Student, Supervisor and FYPCoordinator classes, which provide a consistent behaviour for all users in the FYPMS system by adhering to methods defined in the User interface. The LSP is adhered to because any code that works with a User object can work with any object of a class implementing the User interface, such as Supervisor or Student, without affecting the correctness of the program.

Interface Segregation Principle (ISP)

The Interface Segregation Principle (ISP) states that clients should not be forced to depend on interfaces not used. In the FYPMS App, for Manager classes, such as StudentManager and SupervisorManager, each manager class has their own focused set of methods and responsibilities specific to the entities they manage. By separating the management logic into separate classes, the App follows the ISP, ensuring that clients that need to interact with a specific type of entity depend only on the relevant manager class.

Dependency Injection Principle (DIP)

The Dependency Injection Principle (DIP) states that high-level modules should not depend on low-level modules, but both should depend on abstractions. In the FYPMS project, the manager objects would need to access each other to perform certain tasks, however, the instances of the managers are initialised in the programs. Rather than importing the programs into the managers, which would create a circular dependency and tight coupling, we overcame this by passing the manager objects into other managers as method parameters. This way, the method is not dependent on the specific implementation of the manager but can work with any connection object that follows the same structure, which also promotes modularity in the code and simpler code testing.

Furthermore, high cohesion and loose coupling are also adhered to by reducing the number of other objects that are stored as attributes, instead, they are simply passed as method parameters such that the relationship between objects can be decoupled from association to dependency. An example would be Req_ChangeTitle, where only the project is stored as an attribute and the state of the corresponding Student or Supervisor objects are accessed and modified through get and set methods in the Project and User classes. The requests do not need to know of the implementation details within the users but instead depend on abstractions. This further strengthens the SRP as the Requests classes are now more focused with less attributes and less complexity. Less effort would be required in handling boundary cases as well, such as if there is a change title request and a transfer student request for the same project, the change title request would automatically move to the new supervisor after the transfer is approved, without having to update any additional attributes within the change title request.

OODP Concepts

Abstraction

Abstraction refers to the simplification of complex real-world entities and processes into simpler software components. This object-oriented design allows for the separation of concerns, enhancing the program's maintainability and modularity.

Encapsulation

Encapsulation protects specific field data within a class. Public methods such as getters and setters can only be used to access these private/protected data. In the FYPMS, 'models' attributes are private, hence ensuring data is safe within each class. For example, the method used to authenticate passwords are housed in the respective user's objects itself. When a user attempts to login, the string will be passed into this function which will return true for success and false for failure, instead of extracting the user's password out into the main program for the check.

Inheritance

Inheritance is an object-oriented design that enables the creation of new classes from existing ones, hence supporting code reusability and modularity. In the FYPMS app, inheritance is used to create specialised classes that inherit common properties from parent classes, while adding or overriding specific functionality. For example, Student and Supervisor inherit generic properties of User, such as UserID or name, while implementing specialised attributes or methods for their role, such as having a list of projects and number of allocated projects in Supervisor.

Polymorphism

Polymorphism is the ability of objects of different classes to be used interchangeably, while still maintaining their own unique behaviours. This is achieved through inheritance and method overriding. In the boundary classes where supervisors or FYP coordinator would trigger the approval/rejection of requests, the specific type of request being processed does not need to be specified as they are all subclasses of Request and hence can be treated as if they are of the same class. All types of requests can be passed through the same method approveRequest() or displayRequest(), which will behave differently based on the object calling it, also known as dynamic polymorphism. This promotes code reuse, extensibility, and flexibility, where we can create different subclasses that have their own unique behaviours, while still maintaining a common interface through inheritance, while adhering to LSP as mentioned above.

However, it is to be noted that downcasting might also be required in specific situations, such as the updateProjectsStatus function under ProjectManager where the particular type of request passed needs to be known. In such cases, the programmer has to be mindful to conduct necessary type checking to prevent run-time errors such as using the instanceof operator.

Future Features

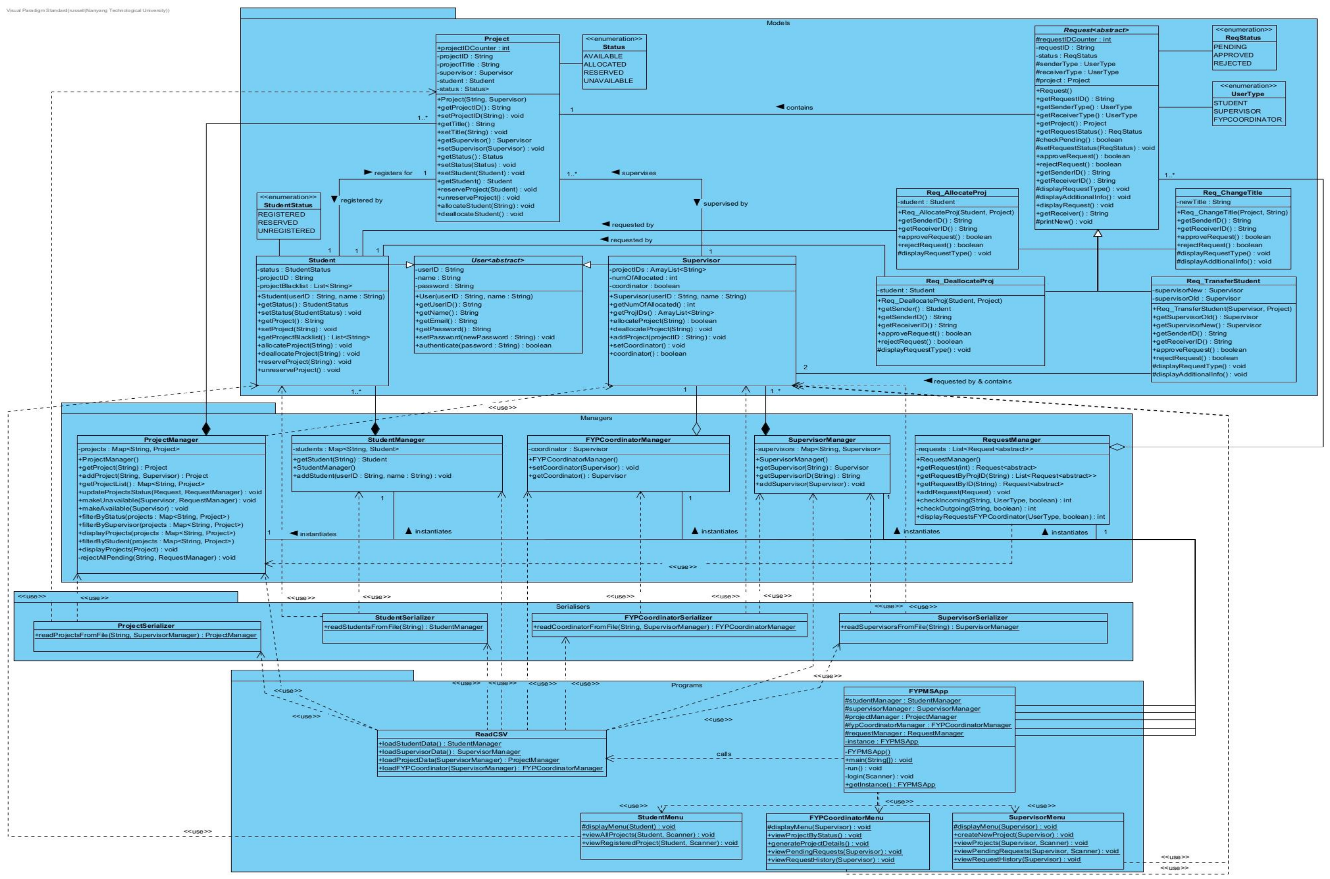
1. **Sending SMS to Users for Pending Requests**

   We can implement SMS manager and SMS entity classes. In the SMS manager class, we can add a new method to get the mobile number of users. Whenever a new request is made to supervisors or FYPCoordinator, or request approved to students, they will be notified, hence saving them time from constantly logging into the FYPMS app to check their request status.

2. **Implement Password Masking**

   Password masking is a security feature that hides the actual characters of a password when a user types it into the password input field. Instead of displaying the typed characters, the input field shows a mask character(usually an asterisk (*) for each character entered. This feature helps protect the password from being seen by others who may be looking at the user's screen, thus enhancing the privacy and security of user credentials.

<u>UML Class Diagram</u> (please refer to the attached link for a clearer picture)



Singleton Class

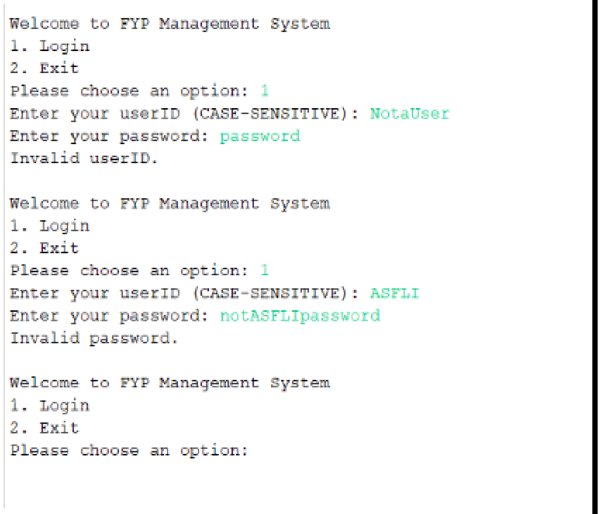
The FYPMSApp class has been designed as a singleton class to ensure that there can only be one instance of it in the application at any point in time, using a private constructor to instantiate itself. As the instances of the managers within the class are all static, it would also not be possible to have more than one of each manager, providing the program with a global database.

Composition relationship

For Projects, Students and Supervisors, the constructor for these classes only exist within the respective Managers, ensuring that all instances of the objects created must be stored in the database. The Manager as a whole composes the Entities as its parts and owns them, hence, the parts cannot exist without the whole.

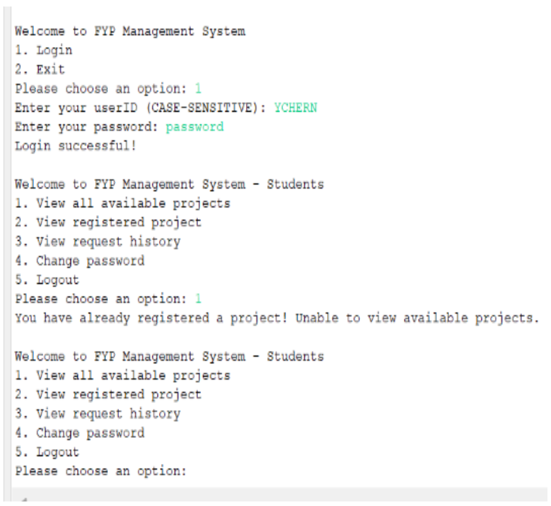
<u>Test Cases (Not Shown in Demon Video + Cases of Input Error Checking)</u>

Invalid user / valid ID but wrong password:



```
Welcome to FYP Management System
1. Login
2. Exit
Please choose an option: 1
Enter your userID (CASE-SENSITIVE): NotaUser
Enter your password: password
Invalid userID.

Welcome to FYP Management System
1. Login
2. Exit
Please choose an option: 1
Enter your userID (CASE-SENSITIVE): ASFLI
Enter your password: notASFLIpassword
Invalid password.

Welcome to FYP Management System
1. Login
2. Exit
Please choose an option:
```

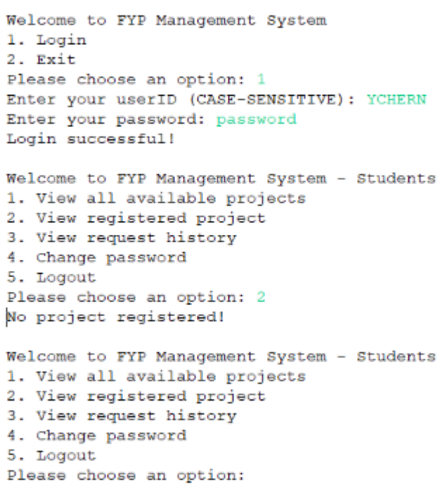
Student already allocated project but tries to view available projects and is unable to view:



```
Welcome to FYP Management System
1. Login
2. Exit
Please choose an option: 1
Enter your userID (CASE-SENSITIVE): YCHERN
Enter your password: password
Login successful!

Welcome to FYP Management System - Students
1. View all available projects
2. View registered project
3. View request history
4. Change password
5. Logout
Please choose an option: 1
You have already registered a project! Unable to view available projects.

Welcome to FYP Management System - Students
1. View all available projects
2. View registered project
3. View request history
4. Change password
5. Logout
Please choose an option:
```

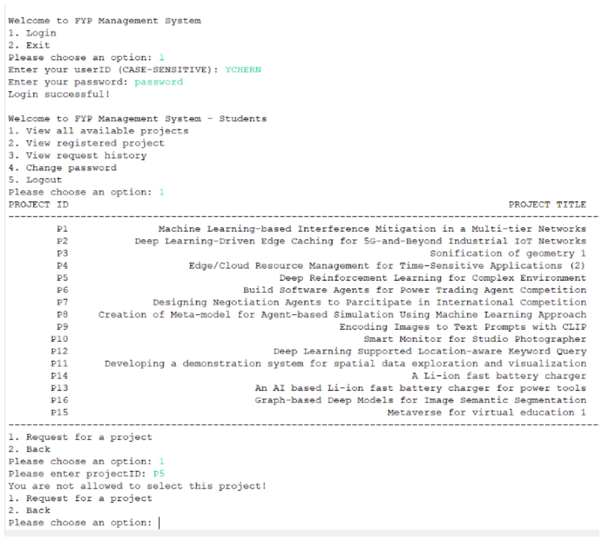
Entering an invalid choice (exception handling):

View registered project fails:



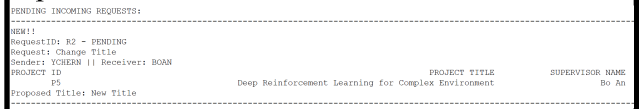
```
Welcome to FYP Management System
1. Login
2. Exit
Please choose an option: 1
Enter your userID (CASE-SENSITIVE): YCHERN
Enter your password: password
Login successful!

Welcome to FYP Management System - Students
1. View all available projects
2. View registered project
3. View request history
4. Change password
5. Logout
Please choose an option: 2
No project registered!

Welcome to FYP Management System - Students
1. View all available projects
2. View registered project
3. View request history
4. Change password
5. Logout
Please choose an option:
```

Student tries to select a project he deregistered before and is unable to register (blacklisted):



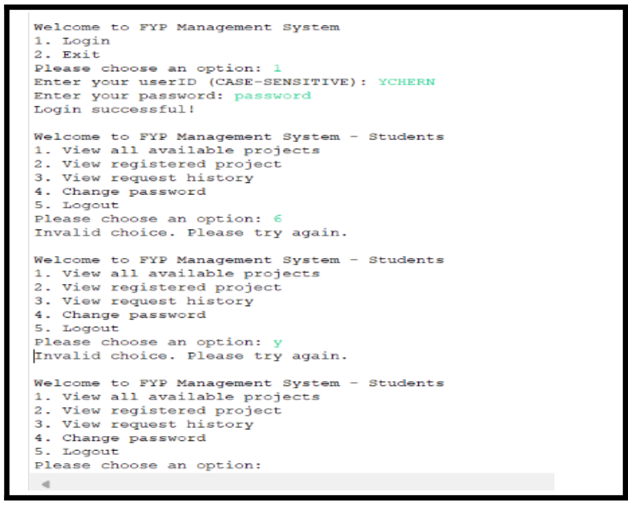
```
Welcome to FYP Management System
1. Login
2. Exit
Please choose an option: 1
Enter your userID (CASE-SENSITIVE): YCHERN
Enter your password: password
Login successful!

Welcome to FYP Management System - Students
1. View all available projects
2. View registered project
3. View request history
4. Change password
5. Logout
Please choose an option: 1
PROJECT ID                                                                                   PROJECT TITLE
-----------------------------------------------------------------------------------------------------------
        P1                     Machine Learning-based Interference Mitigation in a Multi-tier Networks
        P2                 Deep Learning-Driven Edge Caching for 5G-and-Beyond Industrial IoT Networks
        P3                                                                  Sonification of geometry 1
        P4                          Edge/Cloud Resource Management for Time-Sensitive Applications (2)
        P5                                           Deep Reinforcement Learning for Complex Environment
        P6                                      Build Software Agents for Power Trading Agent Competition
        P7                      Designing Negotiation Agents to Parcitipate in International Competition
        P8             Creation of Meta-model for Agent-based Simulation Using Machine Learning Approach
        P9                                                     Encoding Images to Text Prompts with CLIP
       P10                                                         Smart Monitor for Studio Photographer
       P12                                          Deep Learning Supported Location-aware Keyword Query
       P11             Developing a demonstration system for spatial data exploration and visualization
       P14                                                                 A Li-ion fast battery charger
       P13                                             An AI based Li-ion fast battery charger for power tools
       P16                                            Graph-based Deep Models for Image Semantic Segmentation
       P15                                                               Metaverse for virtual education 1
-----------------------------------------------------------------------------------------------------------
1. Request for a project
2. Back
Please choose an option: 1
Please enter projectID: P5
You are not allowed to select this project!
1. Request for a project
2. Back
Please choose an option: |
```

If a student requests to change title while the project's supervisor is transferring him to another supervisor, the change title request would automatically move to the new supervisor:

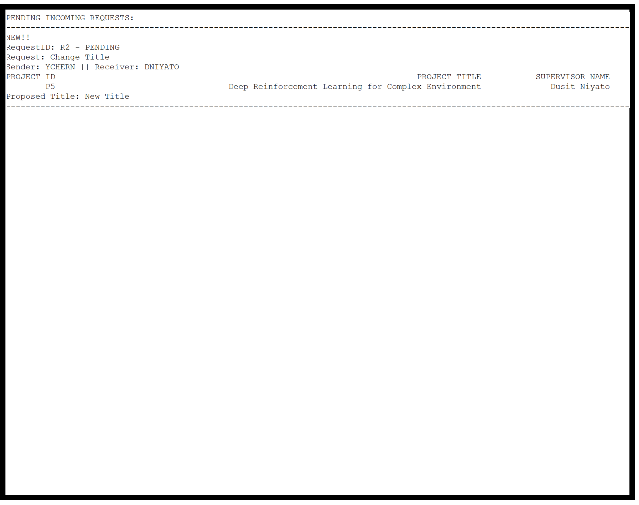


```
PENDING INCOMING REQUESTS:
-------------------------------------------------------------------------------------------------------------------------------
NEW!!
RequestID: R2 - PENDING
Request: Change Title
Sender: YCHERN || Receiver: BOAN
PROJECT ID                                                                             PROJECT TITLE          SUPERVISOR NAME
       P5                                          Deep Reinforcement Learning for Complex Environment                  Bo An
Proposed Title: New Title
-------------------------------------------------------------------------------------------------------------------------------
```

to



```
Welcome to FYP Management System
1. Login
2. Exit
Please choose an option: 1
Enter your userID (CASE-SENSITIVE): YCHERN
Enter your password: password
Login successful!

Welcome to FYP Management System - Students
1. View all available projects
2. View registered project
3. View request history
4. Change password
5. Logout
Please choose an option: 6
Invalid choice. Please try again.

Welcome to FYP Management System - Students
1. View all available projects
2. View registered project
3. View request history
4. Change password
5. Logout
Please choose an option: y
Invalid choice. Please try again.

Welcome to FYP Management System - Students
1. View all available projects
2. View registered project
3. View request history
4. Change password
5. Logout
Please choose an option:
```



```
PENDING INCOMING REQUESTS:
----------------------------------------------------------------------------------------------------------------------------------
NEW!!
RequestID: R2 - PENDING
Request: Change Title
Sender: YCHERN || Receiver: DNIYATO
PROJECT ID                                                                            PROJECT TITLE           SUPERVISOR NAME
        P5                                        Deep Reinforcement Learning for Complex Environment             Dusit Niyato
Proposed Title: New Title
----------------------------------------------------------------------------------------------------------------------------------
```

Reflections

Through this project, we also experienced the iterative process of code improvement, where we had to repeatedly reconsider the way information was being passed from one class to another. Even though the code was working, we discovered several implementations that did not adhere to good design principles, especially through the process of drawing the UML diagram. Much of the code had to be revamped to improve extensibility and maintainability. We learnt that developing a software is not only about writing working codes, but the real challenge lies in writing good code, as there are often multiple ways of doing the same task. There were multiple tradeoffs which we had to consider for each method before deciding on one.

For example, many of the association relationships were changed to dependency to promote loose coupling, reducing the level of awareness between objects. Especially when dealing with requests, we removed most of the unnecessary user data stored within the object and instead relied on the interfaces of each class to extract only the required information. We also found several circular dependencies where packages are being imported into each other, and decided to pass objects into functions instead to align to DIP. We also considered the use of singleton classes on the managers to provide a global point of access and simplify the method signatures, making the code more readable and maintainable.

For further improvement suggestions, the FYPMS app could benefit from the integration of added features such as real-time collaboration, and advanced search and filtering beyond just userID and status of project. These enhancements would provide a more interactive and user-friendly experience for all users.

Overall, the project was a valuable learning experience for us as not only were we able to hone our programming skills, but it also deepened our understanding of OODP and software development best practices. With the knowledge derived, we are better equipped to handle future software development projects and apply the lessons learned to improve and expand upon the current FYPMS application in the future.

Declaration of Original Work for SC/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 the 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 (CE2002 or CZ2002)	Lab Group	Signature /Date
BRYAN CHIA SONG NIAN	SC2002	A50	[illegible] 15.04.2023
PHUN WEI CHENG RUSSELL	SC2002	A50	[illegible] 15.04.2023
SIEW JIA YANG	SC2002	A50	[illegible] 15.04.2023