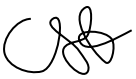
NANYANG TECHNOLOGICAL UNIVERSITY

SC2002 Z51 Assignment Group 1

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	Lab Group	Signature/Date
Anand Chaanan Singh	BCE	Z51	 16/04/23
Anand Chiraag Singh	BCE	Z51	 16/04/23
Guo Chenrui	BCG	Z51	 16/04/23
Chalamalasetti Sree Vaishnavi	BCG	Z51	 16/04/23
Pan Zaiyu, Ryan	BCG	Z51	 16/04/23

Design considerations

Our primary objective is to develop an extensible and maintainable FYP management system.

Assumptions regarding the functionality of the app:

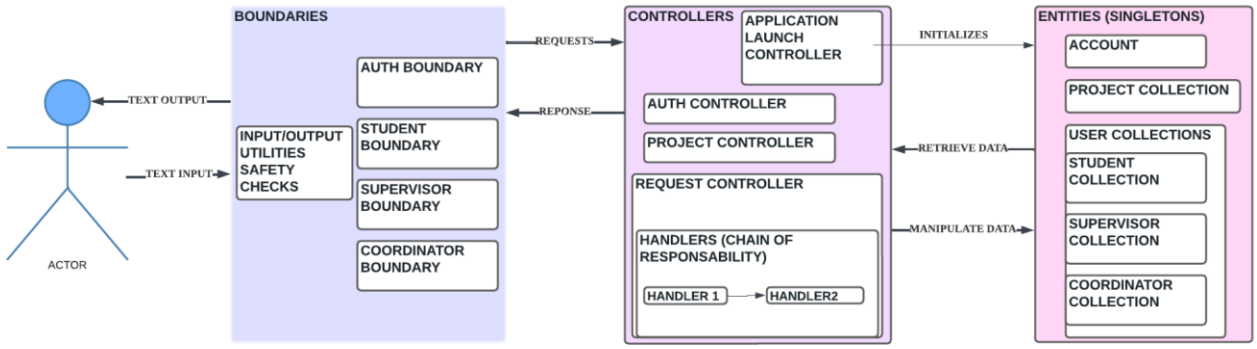
1. A single user accesses the app at any one time; it is not necessary to consider concurrent users.
2. Each student can only send ONE (1) project request, meaning they cannot send any more if they have a pending request.
3. Supervisors cannot choose their students for their FYP project.
4. The requestor cannot cancel any request once it is submitted. For example, students cannot change or withdraw their choice of FYP once a request is submitted (even though the coordinator is yet to respond to their request)
5. Users will never forget their password.

Further, we make some assumptions surrounding the architectural constraints of the app:

6. There are no resource constraints on our system of operation (e.g., memory usage)
7. No existing framework Java Frameworks are to be used.

To implement our application architecture, we use three main design patterns: Entity-Control-Boundary (ECB), the chain of responsibility (COR) and singleton. Furthermore, we use the SOLID design principles to design our architecture's various classes, interfaces, and hierarchies. Subsequent sections elaborate on how we implement our chosen patterns and principles throughout our application and how they allow our system to be extensible and maintainable.

Overview of architecture



Our core application implements the ECB pattern to segregate the classes based on their responsibilities in use-case realisation (Figure 1). The ECB pattern consists of three layers: entity, control, and boundary. The different layers fulfil specific roles (specified later) within the application. By isolating these layers, we achieved loose coupling between them, allowing us to modify a single layer with minimal impact on other layers. For example, if we needed to change the way data is stored, we could modify the implementation of the entity layer without affecting the control or boundary layers, so long as the entity layer applies the concepts of encapsulation and abstraction. This approach results in a more modular and maintainable application that enables ease of modification.

The following sections detail the three main layers of our application and relevant design considerations and principles.

Actors

Our application has three main user types: students, supervisors and coordinators.

Boundary layer

The boundary layer serves as the application interface. Boundary classes take in user input and display output through the command line. To handle various requests from the user, the boundary classes call methods which are provided by classes in the controller layer.

Maintainability & Extensibility

The IPageBoundary interface gives information on how to implement a page in our application. A page is the highest-level Object that renders something for the user to see and contains a void display() method.

The BasePageBoundary class that implements IPageBoundary contains commonly used functionality for most views in a command line app (the ability to print menu items and allow users to select them). Utilising this class enables better code reuse and separation of concern as classes that extend BasePageBoundary no longer need to be concerned about the logic of how to allow the user to select between various options from a menu.

Furthermore, by utilising the UIOption class, BasePageBoundary can dynamically generate menu options that are handled by different handlers. The advantage of this is that when an option needs to be added or removed, the programmer can simply register or de-register it in the constructor of the children of BasePageBoundary. It also means that the handlers that generate the view for a given UIOption are loosely coupled to the Page. A new handler can be used from any other class or package so long as it is Runnable. This class is extended by all the other classes in the boundaries package to implement the views for different user types. This implementation style is, in fact, a working example of the Dependency Injection Principle (DIP).

<u>Entity layer</u>

The entity layer represents the data that needs to be stored and manipulated by our system. There are four main entities: Students, Supervisors, Coordinators, and Projects stored in resizable arrays. Attributes of these entities can be used and updated by the control layer.

The entities layer implements the singleton pattern for storing user and project data. The pattern ensures that there is only one instance of each data class, promoting data consistency throughout the application. And therefore, it helps avoid bugs and makes the code easier to understand and maintain.

There are 3 types of user classes for our application: student, supervisor and coordinator. Each of these classes share similar functionality and attributes. For example, each user class will store account information, and will allow users to perform account-related actions such as retrieving account name. Therefore, we leveraged the concept of inheritance and stored these recurring attributes and methods in an abstract User class. This makes it easier to create the classes for each user and reduces the duplication of code as each user class can extend from the User class. We also considered the Liskov Substitution principle (LSP) by using Final methods in our User class so that subclasses would be unable to override methods from the superclass. This also follows the Open-Closed principle (OCP) since subclasses can only extend the class by adding new attributes and methods without being able to modify inherited methods. This ensures that subclasses are focused on expanding the functionality of the superclass and will hence be substitutable by the superclass.

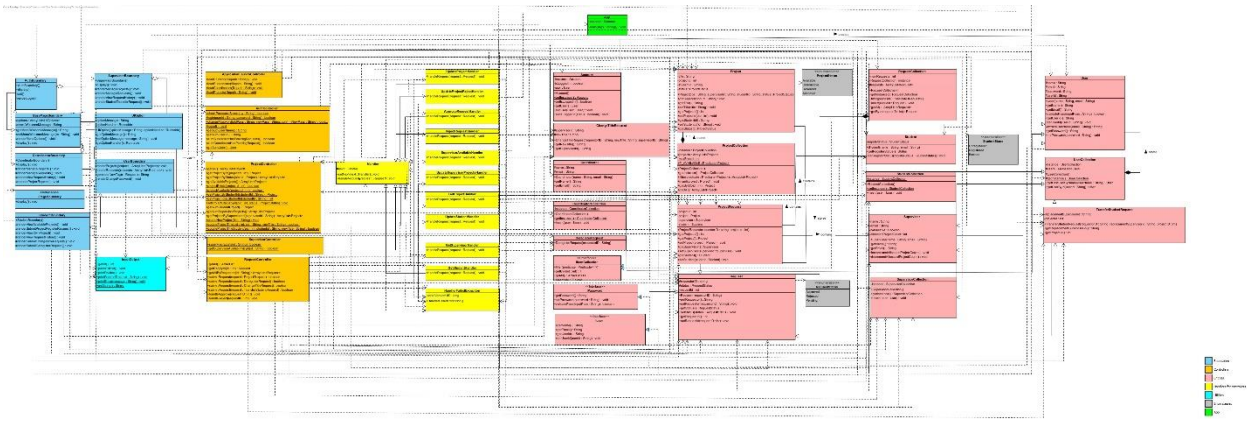
Control layer

The control layer serves as the link between the entity and the boundary layers. It consists of classes that determine and manage the business logic of the application. The control layer receives instructions from the boundary layer which calls the methods required to achieve the intended outcome. Necessary data will be obtained from the entity layer. For example, to view all available FYP projects, the project controller will call the method getAvailableProjects() which will return a list of available FYP projects.

There are 4 main components in the control layer: project, request, authorisation, and application. These components are decoupled into sub-components to improve maintainability and reusability of code across the controllers.

Further the requests controller uses the COR pattern to handle requests (i.e., approve/reject requests of different types). Depending on its type, the request, some combination of Handler objects within a chain, handles the request. Each Handler abstracts common functionalities, which reduces program immobility. Overall, this increases extensibility and flexibility for new types of requests in the future, where we can easily define and add Handler objects to the chain where required

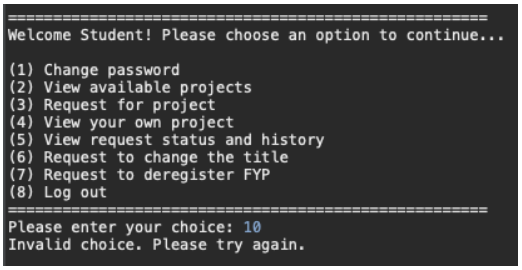
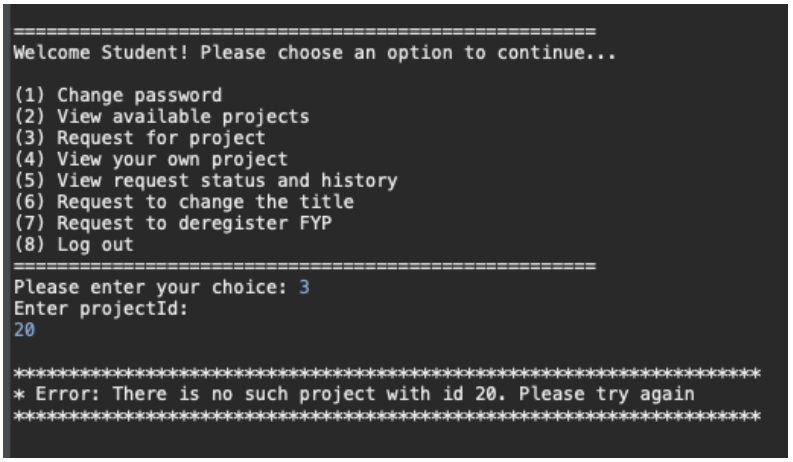
UML Class Diagram



Please refer to the UML diagram in the folder.

Testing

We illustrate a few of the many test cases our system was subjected to.

Test Case	Most Salient Test Results
Registering and confirming project 1. Login as student: Chern 2. View registered projects 3. View available projects 4. Select the project to send to the coordinator (supervisor: Bo An) 5. View available projects 6. View request status and history 7. Logout 8. Login as coordinator: Li Fang 9. View request 10. Approve request 11. Logout 12. Login as supervisor: Bo An 13. View projects 14. Logout	 Invalid choice from student's CLI (Input validation #1)  Message displayed when students attempt to register for invalid project (Input validation #2)

15. Login as student: Chern
16. View registered projects
17. Log out

```
==================================================
Welcome Student! Please choose an option to continue...

(1) Change password
(2) View available projects
(3) Request for project
(4) View your own project
(5) View request status and history
(6) Request to change the title
(7) Request to deregister FYP
(8) Log out
==================================================
Please enter your choice: 4
Your registered fyp project is:
Id: 5 Title: Deep Reinforcement Learning for Complex Environment
```

Student registers successfully

Changing and approving the change of project title

(continued from Case 1)

1. Login as student: Koh
2. View available projects
3. Select the same project as Chern to send to the coordinator (supervisor: Bo An)
4. Select the project to send to the coordinator (supervisor: Bo An)
5. Logout
6. Login as coordinator: Li Fang
7. View request
8. Approve request
9. Logout
10. Login as supervisor: Bo An
11. View request history
12. Update project title
13. View change in project title
14. Logout
15. Login as student: Koh
16. Request a change to project title

```
==================================================
Welcome Student! Please choose an option to continue...

(1) Change password
(2) View available projects
(3) Request for project
(4) View your own project
(5) View request status and history
(6) Request to change the title
(7) Request to deregister FYP
(8) Log out
==================================================
Please enter your choice: 3
Enter projectId:
5

**********************************************************************
* Error: You are not allowed to register for this project!
**********************************************************************
```

Student attempts to register for unavailable project (Input validation #3)

```
Enter a new title for project ID 5 :
New Project Title
Project renamed successfully!
You currently have 3 projects in total. They are as follows:
ID:  5 | Status: ALLOCATED | Title: New Project Title | Student id: YCHERN |
ID:  6 | Status: AVAILABLE | Title: Build Software Agents for Power Trading A
```

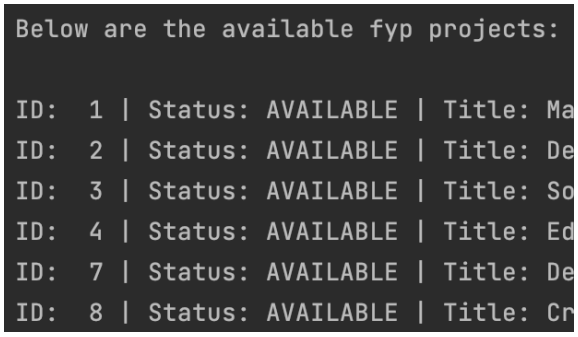
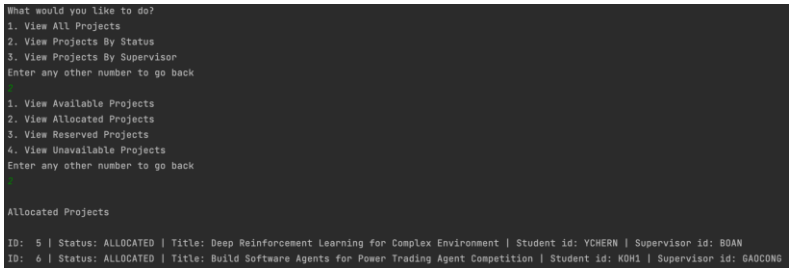
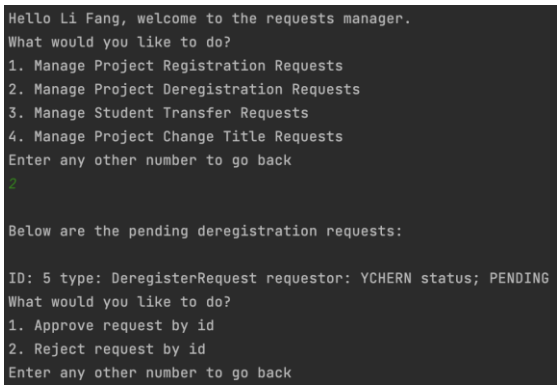
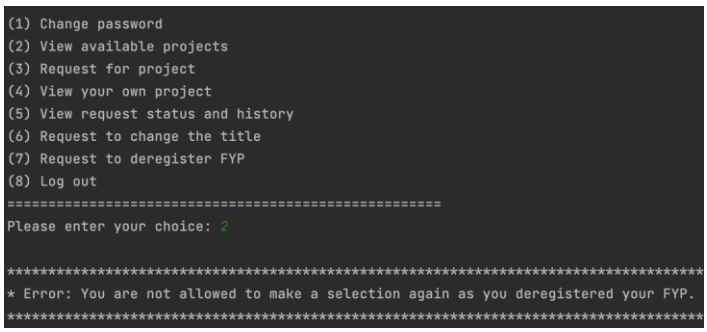
Supervisor renames project

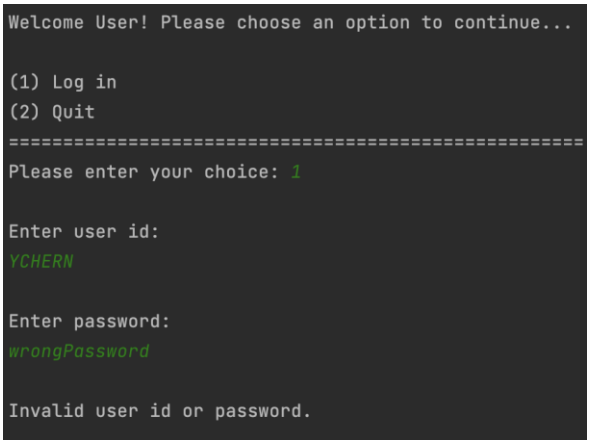
```
Welcome Supervisor! Please select an option to continue...

(1) Change password
(2) Manage projects (View, Create and Update)
(3) Manage pending requests !!NEW!!
(4) View request history
(5) Request to transfer student to a replacement supervisor
(6) Log out
==================================================
Please enter your choice: 3
Hello Bo An, welcome to the requests manager.
You have 1 project title change request(s) pending. They are as follows:
1. Request ID: 2 Student ID: YCHERN Old title: Deep Reinforcement Learning for Complex Environment New title: newTitle
```

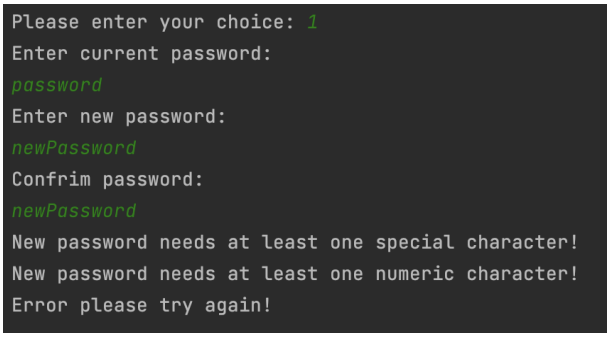
Student requests to rename project to “newTitle”

17. View request history
18. Logoff
19. Login as supervisor: Bo An
20. View pending request history
21. Approve request
22. View request history
23. Logoff
24. Login as student: Koh
25. View registered project
26. Logoff
	`Your registered fyp project is:`
	`Id: 5 Title: newTitle`
	
	Supervisor approves' student request to rename project

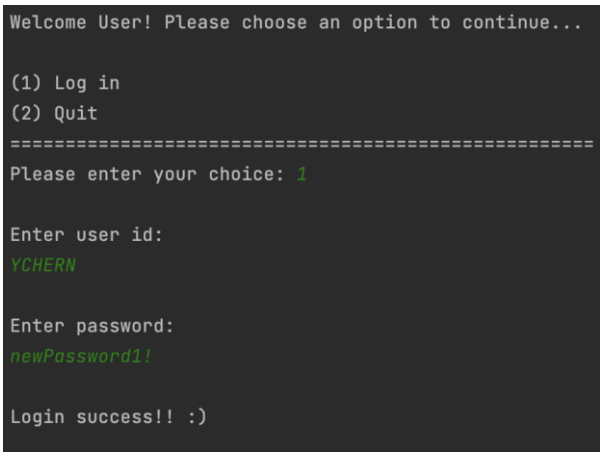
Transferring projects between supervisors
(Continued from Case 2)
1. Login to supervisor: Bo An
2. Create new project
3. Request to transfer Koh's project to Cong Gao
4. Logoff
5. Log in as student: Brandon
6. View available projects
7. Logoff
8. Login as coordinator: Li Fang
9. Approve request for project transfer
10. Logoff
11. Login as supervisor: Cong Gao
12. View projects
13. Logoff
14. Login as student: Brandon
15. View available projects
16. Select the project to send to coordinator (Supervisor: Bo An)
	`Below are the available fyp projects:`
	
	`ID: 1 | Status: AVAILABLE | Title: Ma`
	`ID: 2 | Status: AVAILABLE | Title: De`
	`ID: 3 | Status: AVAILABLE | Title: So`
	`ID: 4 | Status: AVAILABLE | Title: Ed`
	`ID: 8 | Status: AVAILABLE | Title: Cr`
	
	Project ID 7 is not in the available project list as the supervisor (Bo An) already has two existing projects registered.
	
	`Welcome Supervisor! Please select an option to continue...`
	`(1) Change password`
	`(2) Manage projects (View, Create and Update)`
	`(3) Manage pending requests`
	`(4) View request history`
	`(5) Request to transfer student to a replacement supervisor`
	`(6) Log out`
	`Please enter your choice: 2`
	`Hello Cong Gao!You currently have 3 projects in total. They are as follows:`
	`ID: 6 | Status: ALLOCATED | Title: Build Software Agents for Power Trading`
	`ID: 11 | Status: AVAILABLE | Title: Developing a demonstration system for s`
	`ID: 12 | Status: AVAILABLE | Title: Deep Learning Supported Location-aware`
	
	Supervisor Cong Gao receives project ID 6 from Bo An

17. Logoff	 Students can view project ID 7 now since Bo An has 1 vacancy
Viewing and rejecting projects as coordinator and deregistering projects (Continued from Case 3) 1. Login as Calvin 2. Select the project to send to coordinator (supervisor: Bo An) 3. Logoff 4. Login as coordinator: Li Fang 5. View projects according to different filters 6. Reject request 7. Accept request 8. Logoff 9. Login as student: Brandon 10. View outcome of his project registration request 11. Logoff 12. Login as student: Calvin 13. Deregister 14. Logoff 15. Login as coordinator: Li Fang 16. Approve deallocation 17. Logoff	 Coordinator views projects by project status  Coordinator can view deallocation request submitted by student 

<table>
<tr>
<td>
18. Login as student: Calvin
19. View available projects
20. View registered projects
21. Logoff
</td>
<td>Coordinator confirms deallocation, student can no longer view available projects</td>
</tr>
<tr>
<td>
Rejecting transfer of project between supervisors and changing account password
(Restarting the app)
1. Login as student: Chern
2. View available projects
3. Select project 5 to send to coordinator (supervisor: Bo An)
4. Logoff
5. Login as student: Koh
6. Select project 6 to send to coordinator (supervisor: Bo An)
7. Logoff
8. Login as student: Brandon
9. Select project 12 to send to coordinator (supervisor: Gao Cong)
10. Logoff
11. Login as coordinator: Li Fang
12. Approve all 3 project registration requests
13. Logoff
14. Login as supervisor: Cong Gao
15. Transfer project 12 to Bo An
16. Logoff
17. Login as coordinator: Li Fang
18. View request (with message that warns that Bo An has hit his cap of 2 projects)
19. Reject transfer request
</td>
<td>


Invalid user login due to wrong user ID or password (input validation #6)


Attempt to change password fail due to security requirements (input validation #5)
</td>
</tr>
</table>

20. Logoff 21. Login as supervisor: Bo An 22. Entered wrong password on login 23. Change password (does not meet requirements) 24. Change password (that meets requirements) 25. Login with old password failed 26. Login with new password succeeds	 User can log in using new password

Reflections

Challenges

1. Enforcing adherence to SOLID principles without a pattern enforcing framework

In the modern day, most software developers have become users of robust frameworks that are often highly opinionated and restrict the ways in which their users can use them. These restrictions make it very easy to design good code. However, in this project we had complete freedom to design a system from scratch and with this freedom, a key struggle we faced was preventing ourselves from deviating from our chosen design patterns.

2. Creating an intuitive user interface on the command line using reusable components

Another critical challenge was designing a good user interface in a command line. Many difficult decisions had to be made regarding inputs and outputs such as whether the users should be allowed to enter text or only enter integers.

How we tackled them

To tackle this, we created a draft model following the SOLID design principles and ECB architecture. Utilising proper design principles, we realised that a top-down approach where we plan our entire application is not necessary. So long as we design good classes and interfaces it is relatively easy to extend and modify them when we realise that there are new requirements. This realisation enabled us to speed up our development process by simultaneously implementing code

and designing the application. Simultaneous implementation of code also makes you discover new abstractions that can be made, which you would not have realised by simply reading the project requirements and hence it makes the program even more robust.

The ECB architecture allowed us to separate classes based on their responsibilities. This was supported by the SOLID design principles. Single-Responsibility principle (SRP) allowed us to determine the classes needed such that each class can only carry out one specific function. Interface-Segregation principle (ISP) helped determine which classes need to be shown to the respective users. OCP allowed us to consider the methods necessary in each class such that they can be easily extensible without modification of source code. LSP helped us establish the type of methods that can be utilised in superclass and subclass. DIP ensured that we designed classes with minimal coupling and dependency. Overall, the SOLID design principles and ECB architecture helped us better visualise the classes and the relationships to help design the first UML model which was further enhanced throughout the project.

Future enhancements

Since our classes support the OCP and are extensible, additional attributes and methods can easily be added to the relevant class to enable more functionalities in the future:

Students

Since FYP projects tend to focus on specific interest areas, we can implement a recommendation system that suggests the most relevant FYP project to students based on their course and specialisation. This leverages existing information that is already accessible by the school.

Supervisor

To account for the varying levels of prior commitments among supervisors, we can allow them to specify the number of FYP projects that they are available to take on for the year. This will ensure that supervisors are only assigned FYP projects that they can accommodate.