Diploma Projects Management App

Sprint Report

<Fill here the Team Name>
<Στυλιανός Σύρρος ΑΜ:4805, Νικολέτα Κακούρη ΑΜ:4369,Στέφανος Φωτόπουλος ΑΜ:4829 >

VERSIONS HISTORY

Date	Version	Description	Author
23/3/23	v.0.0	Κατανόηση Εργασίας	
29/3/23	v.0.1	Κώδικας σε model,sql	
2/4/23	v.0.2	Κώδικας σε model,sql	
4/4/23	v.0.3	Κώδικας σε model	
10/4/23	v.0.4	Κώδικας σε dao	
15/4/23	v.0.5	Κώδικας σε dao	
16/4/23	v.0.6	Κώδικας σε model	
17/4/23	v.0.7	Κώδικας σε service	
19/4/23	v.0.8	Κώδικας σε service	
25/4/23	v.0.9	Κώδικας σε service	
26/4/23	v.1.0	Κώδικας σε service	
27/4/23	v.1.1	Κώδικας σε service	
1/5 -18/5	v.1.2	Service,Controllers,Htmls,testing	
18/5/23	v.1.3	Testing, Αναφορά	
19/5/23	v.1.4	Τελική Αναφορά	

1 Introduction

1.1 Purpose

Δημιουργία εφαρμογής ανάθεσης διπλωματικών σε φοιτητές μέσω spring

1.2 Document Structure

The rest of this document is structured as follows. Section 2 describes out Scrum team and specifies the this Sprint's backlog. Section 3 specifies the main design concepts for this release of the project.

2 Scrum team and Sprint Backlog

2.1 Scrum team

Product Owner	4805,4369,4829
Scrum Master	4805,4369,4829
Development Team	4805,4369,4829

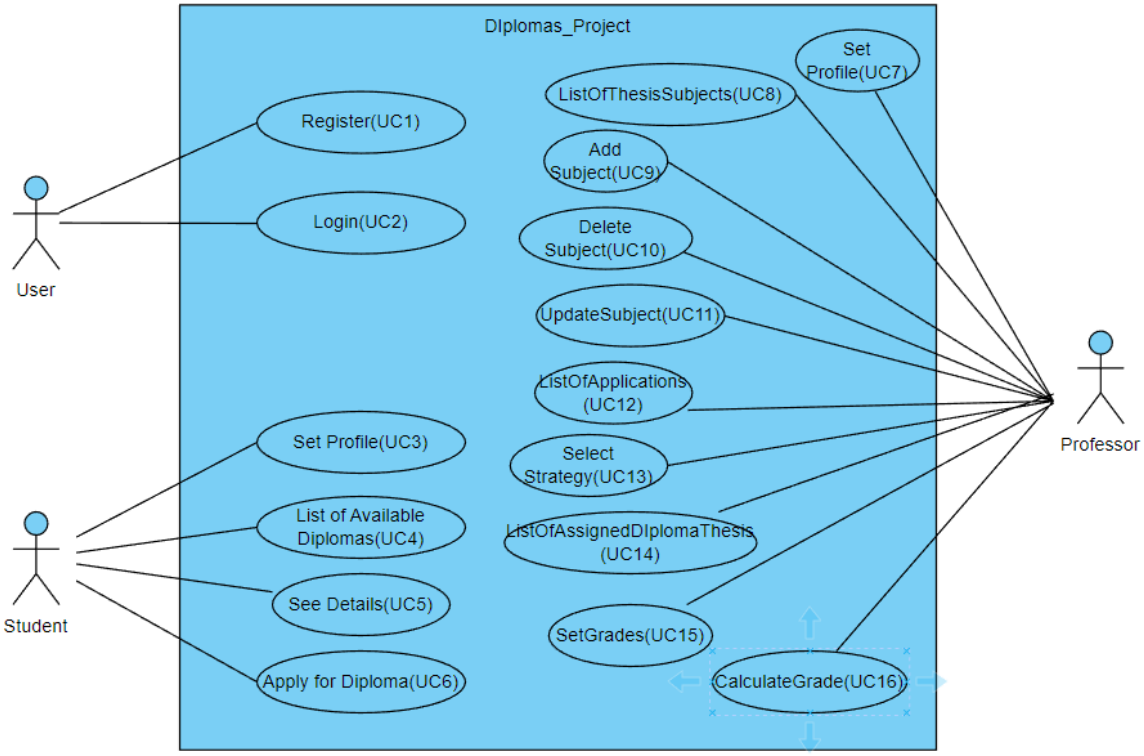
2.2 Sprints

<List below the sprints that you performed and the user stories that have been realized in each Sprint>

Sprint No	Begin Date	End Date	Number of weeks	User stories
1	**17/4**	**25/4**	**1**	**U1, U2**
2	**25/4**	**15/5**	**2**	**S1, S2, S3, S4**
3	**25/4**	**15/5**	**2**	**P1, P2, P3, P4, P5, P6, P7, P8, P9**

3 Use Cases

<Specify the concrete Use Cases that describe the interaction of the user with the applications, as derived from the abstract user stories. Give a **UML Use Case diagram** and the **detailed use case descriptions**.>



3.1 <Use Case 1>

Use case ID	UC1
Actors	Ο χρήστης
Pre conditions	Ο χρήστης να έχει πατήσει το κουμπί register για να μεταφερθεί στην σελίδα εγγραφής
Main flow of events	[Main flow of events that describes the interaction between the user and the application] 1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί register 1.1. Το σύστημα επιστρέφει την σελίδα εγγραφής για να συμπληρωθούν τα

	στοιχεία του 2. Για εγγραφή μαθητή ο χρήστης πρέπει να επιλέξει την επιλογή STUDENT 2.1. Ο χρήστης πατά το κουμπί register για να αποθηκευτούν τα στοιχεία του 2.2. Το σύστημα τον επιστρέφει στην σελίδα του login 3. Για εγγραφή καθηγητή ο χρήστης πρέπει να επιλέξει την επιλογή PROFESSOR 3.1. Ο χρήστης πατά το κουμπί register για να αποθηκευτούν τα στοιχεία του 3.2. Το σύστημα τον επιστρέφει στην σελίδα του login
Alternative flow 1	Σε περίπτωση που υπάρχει ήδη χρήστης με το ίδιο username,επιστρέφει στην σελίδα login με μήνυμα ότι το username που εκχωρήθηκε έχει παρθεί από άλλον χρήστη
Post conditions	Αποθηκεύονται το στοιχεία του χρήστη στην βάση και επιστρέφει στην σελίδα του login

3.2 <Use Case 2>

Use case ID	UC2
Actors	Ο χρήστης
Pre conditions	Ο χρήστης να έχει πραγματοποιήσει την εγγραφή του
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί login 1.1. Το σύστημα επιστρέφει την σελίδα login για να συμπληρωθούν τα στοιχεία του 1.2. Ο χρήστης συμπληρώνει Username και password
Alternative flow 1	Στην περίπτωση που ανοίξει ο χρήστης βάλει λανθασμένα το username ή το password θα τυπωθεί μήνυμα λάθους στην σελίδα
Post conditions	Το σύστημα επιστρέφει την σελίδα όπου ο μαθητής ή ο καθηγητής εισάγουν τα προσωπικά τους στοιχεία

3.3 <Use Case 3>

Use case ID	UC3
Actors	Ο μαθητής
Pre conditions	Ο μαθητής να έχει πραγματοποιήσει την εγγραφή του και την σύνδεση του στην εφαρμογή(login)
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί login 1.1. Το σύστημα επιστρέφει μια σελίδα στην οποία ο μαθητής χρειάζεται να εισάγει τα προσωπικά του στοιχεία(full name,year of studies, current average grade, number of remaining courses for graduation)
Alternative flow 1	Στην περίπτωση που ανοίξει ο χρήστης πατήσει το κουμπί Back to Homepage επιστρέφει στην homepage σελίδα όπου μπορεί να δει πληροφορίες για τον ίδιο και να επιστρέψει στο dashboard με το κουμπί Student Dashboard
Post conditions	Το σύστημα επιστρέφει την σελίδα dashboard του μαθητή

3.4 <Use Case 4>

Use case ID	UC4
Actors	Ο μαθητής
Pre conditions	Ο μαθητής να έχει εισάγει τα προσωπικά του στοιχεία
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί save για να αποθηκευτούν τα στοιχεία του 1.1. Το σύστημα επιστρέφει μια σελίδα στην οποία ο μαθητής βλέπει όλα τα διαθέσιμα θέματα διπλωματικής που μπορεί να πραγματοποιήσει αίτηση
Alternative flow 1	Στην περίπτωση που ανοίξει ο χρήστης πατήσει το κουμπί Back to Homepage επιστρέφει στην homepage σελίδα όπου μπορεί να δει πληροφορίες για τον ίδιο και να επιστρέψει στο dashboard με το κουμπί Student Dashboard
Post conditions	Το σύστημα επιστρέφει την σελίδα dashboard του μαθητή ή του καθηγητή ανάλογα με τον ρόλο που έχει επιλέξει

3.5 <Use Case 5>

Use case ID	UC5
Actors	Ο μαθητής
Pre conditions	Ο μαθητής να έχει εισάγει τα προσωπικά του στοιχεία
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί see Details 1.1. Το σύστημα επιστρέφει μια σελίδα στην οποία ο μαθητής βλέπει μια πιο εκτενή περιγραφή του διπλωματικής με στοιχεία όπως(title, objectives,supervisor).
Alternative flow 1	Στην περίπτωση που ανοίξει ο χρήστης πατήσει το κουμπί Back to Dashboard επιστρέφει στο dashboard του μαθητή
Alternative flow 2	Στην περίπτωση που ανοίξει ο χρήστης πατήσει το κουμπί Back to Homepage επιστρέφει στην homepage σελίδα όπου μπορεί να δει πληροφορίες για τον ίδιο και να επιστρέψει στο dashboard με το κουμπί Student Dashboard
Post conditions	Το σύστημα παραμένει στην σελίδα αυτή μέχρι να πατηθεί το κουμπί Back to Dashboard όπου επιστρέφει στο dashboard του μαθητή

3.6 <Use Case 6>

Use case ID	UC6
Actors	Ο μαθητής
Pre conditions	Ο μαθητής να έχει εισάγει τα προσωπικά του στοιχεία και ο καθηγητής να έχει δημιουργήσει subjects για διπλωματική
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί apply 1.1. Το σύστημα παραμένει στην σελίδα και ο καθηγητής έχει ενημερωθεί για την συγκεκριμένη αίτηση.
Post conditions	Το σύστημα παραμένει στην σελίδα αυτή μέχρι να πατηθεί το κουμπί Back to Dashboard όπου επιστρέφει στο dashboard του μαθητή

3.7 <Use Case 7>

Use case ID	UC7
Actors	Ο καθηγητής
Pre conditions	Ο καθηγητής να έχει πραγματοποιήσει την εγγραφή του και την σύνδεση του στην εφαρμογή(login)
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί login 1.1. Το σύστημα επιστρέφει μια σελίδα στην οποία ο καθηγητής χρειάζεται να εισάγει τα προσωπικά του στοιχεία(full name,specialty).
Alternative flow 1	Στην περίπτωση που ανοίξει ο χρήστης πατήσει το κουμπί Back to Homepage επιστρέφει στην homepage σελίδα όπου μπορεί να δει πληροφορίες για τον ίδιο και να επιστρέψει στο dashboard με το κουμπί Professor Dashboard
Post conditions	Το σύστημα επιστρέφει την σελίδα dashboard του καθηγητή

3.8 <Use Case 8>

Use case ID	UC8
Actors	Ο καθηγητής
Pre conditions	Ο καθηγητής να έχει εισάγει τα προσωπικά του στοιχεία
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί save στην σελίδα όπου εισάγει τα προσωπικά του στοιχεία 1.1. Το σύστημα επιστρέφει μια σελίδα στην οποία ο καθηγητής μπορεί να δει όλες τις διαθέσιμες διπλωματικές εργασίες που προσφέρει ο ίδιος και δεν έχουν ακόμη ανατεθεί
Alternative flow 1	Στην περίπτωση που ανοίξει ο χρήστης πατήσει το κουμπί Back to Homepage επιστρέφει στην homepage σελίδα όπου μπορεί να δει πληροφορίες για τον ίδιο και να επιστρέψει στο dashboard με το κουμπί Professor Dashboard
Post conditions	Το σύστημα παραμένει στην σελίδα dashboard του καθηγητή

3.9 <Use Case 9>

Use case ID	UC9
Actors	Ο καθηγητής
Pre conditions	Ο καθηγητής να έχει εισάγει τα προσωπικά του στοιχεία
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί Add Subject 1.1. Το σύστημα επιστρέφει μια σελίδα στην οποία ο καθηγητής μπορεί να δημιουργήσει νέα διπλωματική εργασία με (title,description) 1.2. Ο χρήστης πατά το κουμπί save και το σύστημα επιστρέφει την σελίδα dashboard
Post conditions	Το σύστημα παραμένει στην σελίδα dashboard του καθηγητή

3.10 Use Case 10>

Use case ID	UC10
Actors	Ο καθηγητής
Pre conditions	Ο καθηγητής να έχει εισάγει τα προσωπικά του στοιχεία και να έχει δημιουργήσει subjects για διπλωματική
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί Delete στην σελίδα dashboard του καθηγητή 1.1. Το σύστημα διαγράφει από την βάση το συγκεκριμένο μάθημα μαζί με όλες τις αιτήσεις που είχαν δημιουργηθεί για αυτό
Alternative flow 1	Στην περίπτωση που ανοίξει ο χρήστης πατήσει το κουμπί Back to Homepage επιστρέφει στην homepage σελίδα όπου μπορεί να δει πληροφορίες για τον ίδιο και να επιστρέψει στο dashboard με το κουμπί Professor Dashboard
Post conditions	Το σύστημα παραμένει στην σελίδα dashboard του καθηγητή

3.11 Use Case 11>

Use case ID	UC11
Actors	Ο καθηγητής
Pre conditions	Ο καθηγητής να έχει εισάγει τα προσωπικά του στοιχεία και να έχει δημιουργήσει subjects για διπλωματική
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί Update στην σελίδα dashboard του καθηγητή 1.1. Το σύστημα ενημερώνει τα πεδία title,description για την συγκεκριμένη διπλωματική εργασία
Alternative flow 1	Στην περίπτωση που ανοίξει ο χρήστης πατήσει το κουμπί Back to Homepage επιστρέφει στην homepage σελίδα όπου μπορεί να δει πληροφορίες για τον ίδιο και να επιστρέψει στο dashboard με το κουμπί Professor Dashboard
Post conditions	Το σύστημα παραμένει στην σελίδα dashboard του καθηγητή

3.12 Use Case 12>

Use case ID	UC12
Actors	Ο καθηγητής
Pre conditions	Ο καθηγητής να έχει εισάγει τα προσωπικά του στοιχεία και να έχει δημιουργήσει subjects για διπλωματική
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί List Applications στην σελίδα dashboard του καθηγητή 1.1. Το σύστημα επιστρέφει μια σελίδα με όλες τις αιτήσεις που έχουν πραγματοποιηθεί από μαθητές για την συγκεκριμένη διπλωματική
Alternative flow 1	Στην περίπτωση που ανοίξει ο χρήστης πατήσει το κουμπί Back to Homepage επιστρέφει στην homepage σελίδα όπου μπορεί να δει πληροφορίες για τον ίδιο και να επιστρέψει στο dashboard με το κουμπί Professor Dashboard
Post conditions	Το σύστημα παραμένει στην σελίδα των αιτήσεων μέχρι ο χρήστης να πατήσει το κουμπί Back to Dashboard όπου επιστρέφει στο dashboard του καθηγητή

3.13 Use Case 13>

Use case ID	UC13
Actors	Ο καθηγητής
Pre conditions	Ο καθηγητής να έχει εισάγει τα προσωπικά του στοιχεία, να έχει δημιουργήσει subjects για διπλωματική και να έχουν πραγματοποιηθεί αιτήσεις για τη συγκεκριμένη διπλωματική
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί Assign Subject στην σελίδα dashboard του καθηγητή 1.1. Το σύστημα επιστρέφει μια σελίδα με όλες τις πιθανές στρατηγικές ανάθεσης μιας διπλωματικής(random choice, student with the best average courses grade, student with the fewest remaining courses for graduation, student with average courses grade greater than a given threshold Th1 and number of remaining courses graduation less than a given threshold Th2) 1.2. Ο καθηγητής επιλέγει μια από αυτές και το μάθημα ανατίθεται σε έναν μαθητή 1.3. Το σύστημα επιστρέφει στην σελίδα dashboard του καθηγητή που είναι πλέον χωρίς το μάθημα που ανέθεσε προηγουμένως

Alternative flow 1	Στην περίπτωση που ο καθηγητής επιλέξει την στρατηγική με τα thresholds,το σύστημα επιστρέφει μια σελίδα που χρειάζεται να εισάγει Threshold για το μέσο όρο του μαθητή και Threshold για τα μαθήματα που χρωστάει ο μαθητής. Έπειτα, το σύστημα επιστρέφει στην σελίδα dashboard του καθηγητή χωρίς το μάθημα που ανατέθηκε. Σε περίπτωση που δεν υπάρχει μαθητής που να τηρεί τα Thresholds η διπλωματική δεν θα ανατεθεί σε κάποιον και θα παραμείνει στη λίστα του καθηγητή
Alternative flow 2	Στην περίπτωση που ο καθηγητής επιλέξει την στρατηγική , student with the fewest remaining courses for graduation εάν τα μαθήματα για πτυχίο δυο μαθητών ισοβαθμούν ο μαθητής που θα πάρει το μάθημα θα επιλεγεί τυχαία
Alternative flow 3	Στην περίπτωση που ο καθηγητής επιλέξει την στρατηγική student with the best average courses grade εάν ο μέσος όρος δυο μαθητών ισοβαθμεί ο μαθητής που θα πάρει το μάθημα θα επιλεγεί τυχαία
Post conditions	Το σύστημα παραμένει στην σελίδα των αιτήσεων μέχρι ο χρήστης να πατήσει το κουμπί Back to Dashboard όπου επιστρέφει στο dashboard του καθηγητή

3.14 Use Case 14>

Use case ID	UC14
Actors	Ο καθηγητής
Pre conditions	Ο καθηγητής να έχει αναθέσει διπλωματική σε κάποιον μαθητή
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί See Thesis στην σελίδα dashboard του καθηγητή 1.1. Το σύστημα επιστρέφει μια σελίδα με όλες τις διπλωματικές του συγκεκριμένου καθηγητή που έχουν ανατεθεί σε μαθητές 1.2. Το σύστημα επιστρέφει στην σελίδα dashboard του καθηγητή όταν πατηθεί το κουμπί Back to Dashboard
Alternative flow 1	Στην περίπτωση που ανοίξει ο χρήστης πατήσει το κουμπί Back to Homepage επιστρέφει στην homepage σελίδα όπου μπορεί να δει πληροφορίες για τον ίδιο και να επιστρέψει στο dashboard με το κουμπί Professor Dashboard
Post conditions	Το σύστημα παραμένει στην σελίδα των διπλωματικών που έχουν ανατεθεί, μέχρι ο χρήστης να πατήσει το κουμπί Professor Dashboard όπου επιστρέφει στο dashboard του καθηγητή

3.15 Use Case 15>

Use case ID	UC15
Actors	Ο καθηγητής
Pre conditions	Ο χρήστης έχει πατήσει το κουμπί See Thesis
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί Set Grades στην σελίδα για το κουμπί See Thesis του καθηγητή 1.1. Το σύστημα επιστρέφει μια σελίδα στην οποία ο καθηγητής εισάγει τους βαθμούς της διπλωματικής εργασίας 1.2. Το σύστημα παραμένει στην σελίδα αυτή μέχρι να πατηθεί το κουμπί Save Grades το οποίο μας επιστρέφει στην σελίδα των διπλωματικών που έχουν ανατεθεί με ενημερωμένες πλέον τις τιμές στους βαθμούς της εργασίας
Alternative flow 1	Στην περίπτωση που ανοίξει ο χρήστης πατήσει το κουμπί Calculate Thesis Grade χωρίς να έχουν εισαχθεί πρώτα οι βαθμοί της εργασίας θα παρατηρήσουμε ότι ο συνολικός βαθμός παραμένει μηδέν
Post conditions	Το σύστημα παραμένει στην σελίδα των διπλωματικών που έχουν ανατεθεί, μέχρι ο χρήστης να πατήσει το κουμπί Professor Dashboard όπου επιστρέφει στο dashboard του καθηγητή

3.16 <Use Case 16>

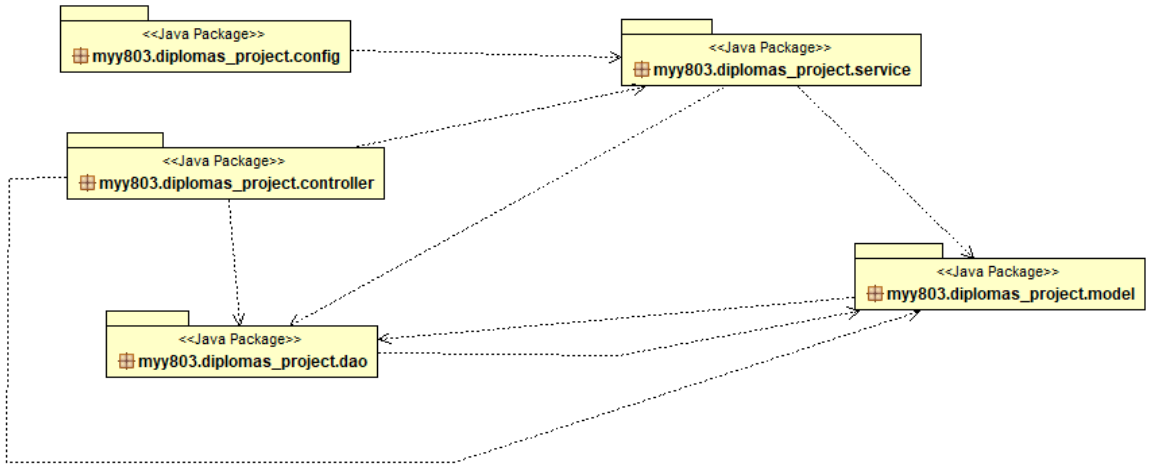
Use case ID	UC16
Actors	Ο καθηγητής
Pre conditions	Ο καθηγητής να έχει δώσει βαθμούς για την συγκριμένη διπλωματική
Main flow of events	1. Το use case ξεκινάει όταν ο χρήστης πατήσει το κουμπί Calculate Thesis Grade στην σελίδα για το κουμπί See Thesis του καθηγητή 1.1. Το σύστημα παραμένει στην σελίδα στην οποία βρισκόταν με ενημερωμένη πλέον την τιμή στο Average Grade
Alternative flow 1	Στην περίπτωση που ανοίξει ο χρήστης πατήσει το κουμπί Set Grades Grade ,το σύστημα επιστρέφει την σελίδα που ο καθηγητής ορίζει τους βαθμούς για μια εργασία.
Post	Το σύστημα παραμένει στην σελίδα των διπλωματικών που έχουν ανατεθεί,

conditions	μέχρι ο χρήστης να πατήσει το κουμπί Professor Dashboard όπου επιστρέφει στο dashboard του καθηγητή

4 Design

4.1 Architecture

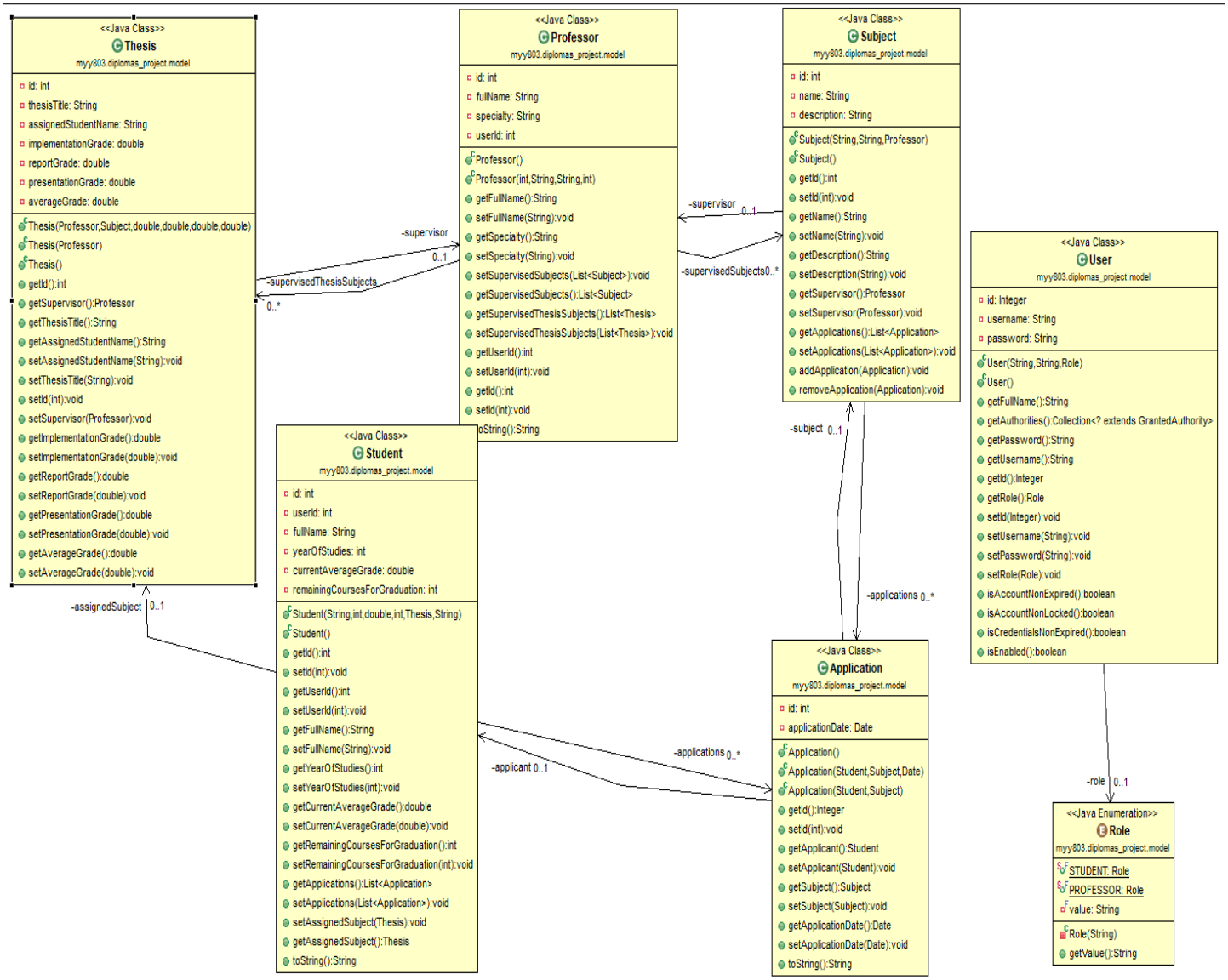
<Specify the overall architecture for this release in terms of a **UML package diagram**.>



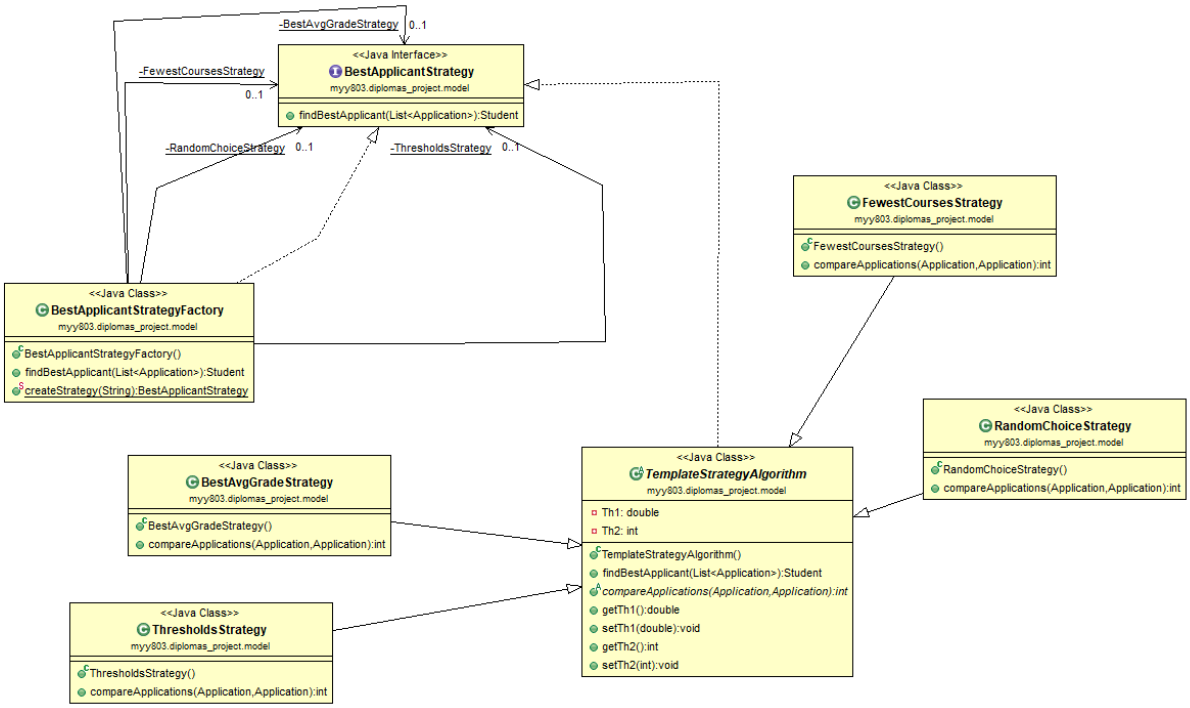
4.2 Design

<Specify the detailed design for this release in terms of **UML class diagrams**.>

Model class diagram:



MODEL CLASS DIAGRAM PART1



MODEL CLASS DIAGRAM PART2

Class Name: Professor	
Responsibilities: ▪ Store and retrieve information about a professor, including their full name, specialty, and user ID. ▪ Manage a list of subjects that the professor supervises, including both regular subjects and thesis subjects. ▪ Provide methods to set and retrieve the professor's supervised subjects and thesis subjects. ▪ Generate a unique ID for each professor. ▪ Convert the Professor object to a string representation.	**Collaborations:** ▪ Subject class: The Professor class collaborates with the Subject class through the supervisedSubjects field, which represents the list of subjects that the professor supervises. The Professor class can add, remove, and retrieve subjects from this list. ▪ Thesis class: The Professor class collaborates with the Thesis class through the supervisedThesisSubjects field, which represents the list of thesis subjects supervised by the professor. The Professor class can add, remove, and retrieve thesis subjects from this list. ▪ ThesisDAO class: The Professor class might collaborate with the ThesisDAO

	class, assuming it exists, to perform database operations related to thesis subjects. ▪ User class: The Professor class is associated with a user through the userId field, which represents the ID of the user associated with the professor. The User class might contain additional information about the user.

Class Name: Student	
Responsibilities: ▪ Store and retrieve information about a student, including their full name, year of studies, current average grade, and remaining courses for graduation. ▪ Manage a list of applications made by the student for theses. ▪ Store and retrieve the assigned subject (diploma) for the student. ▪ Generate a unique ID for each student. ▪ Convert the Student object to a string representation.	**Collaborations:** ▪ Application class: The Student class collaborates with the Application class through the applications field, which represents the list of applications made by the student. The Student class can add, remove, and retrieve applications from this list. ▪ Thesis class: The Student class collaborates with the Thesis class through the assignedSubject field, which represents the assigned subject (diploma) for the student. The Student class can set and retrieve the assigned subject. ▪ User class: The Student class is associated with a user through the userId field, which represents the ID of the user associated with the student. The User class might contain additional information about the user.

Class Name: User	
Responsibilities: ▪ Store and retrieve information about a user, including their username, password, and role. ▪ Implement the UserDetails interface from Spring Security, providing necessary information for user authentication and authorization. ▪ Provide methods to set and retrieve the user's ID and role. ▪ Generate a unique ID for each user. ▪ Determine the user's authorities (roles) for authentication and authorization purposes. ▪ Indicate whether the user account is expired, locked, or has expired credentials. ▪ Indicate whether the user account is enabled.	**Collaborations:** ▪ Role enum: The User class collaborates with the Role enum to represent the user's role or authority. The Role enum provides a set of predefined roles that can be assigned to the user. ▪ GrantedAuthority interface: The User class collaborates with the GrantedAuthority interface from Spring Security. The getAuthorities() method in the User class returns a collection of GrantedAuthority objects that represent the user's authorities or roles. ▪ Spring Security: The User class interacts with Spring Security to provide user authentication and authorization capabilities. It implements the UserDetails interface, which is part of Spring Security's authentication and authorization framework.

Class Name: Subject	
Responsibilities: ▪ Store and retrieve information about a subject, including its name and description. ▪ Maintain a reference to the professor who supervises the subject. ▪ Manage a list of applications made for the subject. ▪ Provide methods to add and remove applications from the list. ▪ Generate a unique ID for each subject.	**Collaborations:** ▪ Professor class: The Subject class collaborates with the Professor class by maintaining a reference to the professor who supervises the subject. The Professor object represents the supervisor for the subject. ▪ Application class: The Subject class collaborates with the Application class through the applications field, which represents the list of applications made for the subject. The Subject class can add, remove, and retrieve applications from this list.

Class Name: Thesis	
Responsibilities: ▪ Store and retrieve information about a thesis, including its title, implementation grade, report grade, presentation grade, and average grade. ▪ Maintain a reference to the professor who supervises the thesis. ▪ Track the name of the assigned student for the thesis. ▪ Generate a unique ID for each thesis.	**Collaborations:** ▪ Professor class: The Thesis class collaborates with the Professor class by maintaining a reference to the professor who supervises the thesis. The Professor object represents the supervisor for the thesis. ▪ Subject class (commented out): The Thesis class may collaborate with the Subject class, as indicated by the commented out Subject instance variable. However, in the provided code, the collaboration is not active.

Class Name: Application	
Responsibilities: ▪ Store and retrieve information about an application, including its ID, applicant, subject, and application date. ▪ Maintain references to the student who made the application and the subject to which the application is related. ▪ Generate a unique ID for each application.	**Collaborations:** ▪ Student class: The Application class collaborates with the Student class by maintaining a reference to the student who made the application. The Student object represents the applicant for the application. ▪ Subject class: The Application class collaborates with the Subject class by maintaining a reference to the subject to which the application is related. The Subject object represents the subject for which the application is made.

Class Name: BestApplicantStrategy	
Responsibilities: ▪ Define the contract for a strategy to find the best applicant from a list of applications. ▪ Specify the method findBestApplicant that takes a list of Application objects as input and returns the best Student object based on some criteria.	**Collaborations:** ▪ Application class: The implementation of the BestApplicantStrategy interface will collaborate with the Application class, as it requires a list of Application objects to find the best applicant. ▪ Student class: The implementation of the BestApplicantStrategy interface may collaborate with the Student class, as it returns the best Student object from the list of applications.

Class Name: BestApplicantStrategyFactory	
Responsibilities: ▪ Implement the BestApplicantStrategy interface. ▪ Create instances of different strategies for finding the best applicant based on the provided strategy name. ▪ Provide a method createStrategy that takes a strategy name as input and returns the corresponding strategy implementation. ▪ Define the method findBestApplicant inherited from the BestApplicantStrategy interface, which is responsible for finding the best applicant based on the selected strategy.	**Collaborations:** ▪ BestApplicantStrategy interface: The BestApplicantStrategyFactory class implements this interface, indicating a collaboration with the strategies defined by the interface. ▪ Strategy classes (BestAvgGradeStrategy, FewestCoursesStrategy, RandomChoiceStrategy, ThresholdsStrategy): The BestApplicantStrategyFactory class collaborates with these strategy classes by creating instances of them based on the strategy name provided. ▪ Application class: The findBestApplicant method of the factory class takes a list of Application objects as input to find the best applicant. It implies a collaboration with the Application class.

Class Name: BestAvgGradeStrategy	
Responsibilities: ▪ Extend the TemplateStrategyAlgorithm class to implement a specific strategy for finding the best applicant. ▪ Implement the compareApplications method, which compares two Application objects based on the average grades of their applicants and returns an integer value indicating the comparison result. ▪ Generate a random selection between two applications if their average grades are equal.	**Collaborations:** ▪ TemplateStrategyAlgorithm class: The BestAvgGradeStrategy class extends this class, indicating a collaboration with the template algorithm for the strategy implementation. ▪ Application class: The compareApplications method takes two Application objects as input to compare their average grades. It implies a collaboration with the Application class.

Class Name: FewestCoursesStrategy	
Responsibilities: ▪ Extend the TemplateStrategyAlgorithm class to implement a specific strategy for finding the best applicant. ▪ Implement the compareApplications method, which compares two Application objects based on the number of remaining courses for graduation of their applicants and returns an integer value indicating the comparison result. ▪ Generate a random selection between two applications if the number of remaining courses for graduation is equal.	**Collaborations:** ▪ TemplateStrategyAlgorithm class: The FewestCoursesStrategy class extends this class, indicating a collaboration with the template algorithm for the strategy implementation. ▪ Application class: The compareApplications method takes two Application objects as input to compare the number of remaining courses for graduation. It implies a collaboration with the Application class.

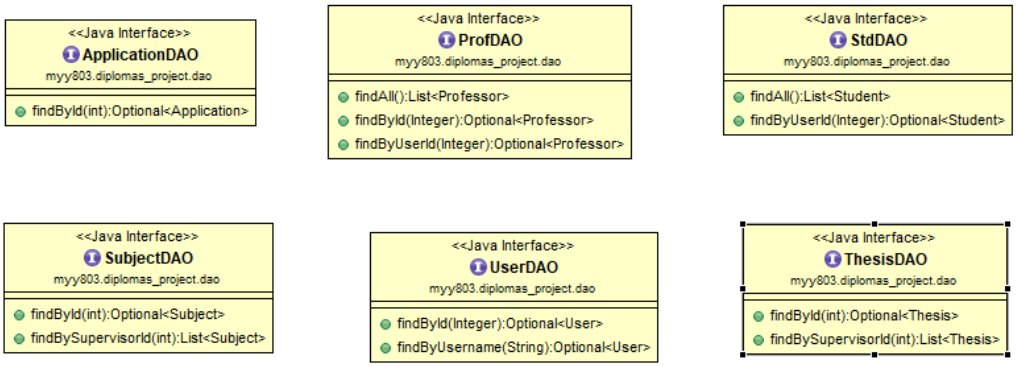
Class Name: RandomChoiceStrategy	
Responsibilities: ▪ Extend the TemplateStrategyAlgorithm class to implement a specific strategy for finding the best applicant. ▪ Implement the compareApplications method, which compares two Application objects randomly and returns an integer value indicating the comparison result. ▪ Generate a random selection between two applications, assigning a higher value to one application and a lower value to the other.	**Collaborations:** ▪ TemplateStrategyAlgorithm class: The RandomChoiceStrategy class extends this class, indicating a collaboration with the template algorithm for the strategy implementation. ▪ Application class: The compareApplications method takes two Application objects as input to compare them randomly. It implies a collaboration with the Application class. ▪ Random class: The RandomChoiceStrategy class utilizes the Random class to generate a random number for making the choice between two applications. It collaborates with the Random class to generate random values.

Class Name: TemplateStrategyAlgorithm	
Responsibilities: ▪ Define a template algorithm for finding the best applicant based on a specific strategy. ▪ Implement the findBestApplicant method, which takes a list of Application objects and iterates through them to determine the best applicant based on the implemented strategy. ▪ Manage the process of comparing applications and selecting the best applicant based on the strategy. ▪ Provide a placeholder for the compareApplications method, which	**Collaborations:** ▪ BestApplicantStrategy interface: The TemplateStrategyAlgorithm class implements this interface, indicating a collaboration with the strategy pattern for finding the best applicant. ▪ Application class: The findBestApplicant method takes a list of Application objects as input and performs comparisons among them to determine the best applicant. It implies a collaboration with the Application class. ▪ Subclasses: The TemplateStrategyAlgorithm class

will be implemented in the subclasses to define the specific comparison logic.	serves as a base class for implementing different strategies. Subclasses extend this class and provide specific implementations for the compareApplications method, defining the comparison logic for their respective strategies. The collaboration occurs with the subclasses that extend the TemplateStrategyAlgorithm class and implement the required method.

Class Name: ThresholdsStrategy	
Responsibilities: ▪ Implement the compareApplications method, which takes two Application objects and compares them based on the defined thresholds for average grade and remaining courses. ▪ Retrieve the average grade and remaining courses for each application from the corresponding Student objects associated with the applications. ▪ Determine which application is considered better based on the thresholds and the comparison results. ▪ Handle various scenarios: ▪ If both applications satisfy the thresholds, randomly choose one as better. ▪ If only one application satisfies the thresholds, consider it as better. ▪ If neither application satisfies the thresholds, consider them equal.	**Collaborations:** ▪ TemplateStrategyAlgorithm class: The ThresholdsStrategy class extends this abstract class, inheriting its structure and behavior. It indicates a collaboration with the template pattern for finding the best applicant. ▪ Application class: The compareApplications method takes two Application objects as input and accesses the associated Student objects to retrieve their average grade and remaining courses. It implies a collaboration with the Application class. ▪ Random class: The compareApplications method utilizes the Random class to generate a random value for handling the scenario when both applications satisfy the thresholds. It implies a collaboration with the Random class for generating random numbers.

Dao class diagram:



DAO CLASS DIAGRAM

Class Name: ApplicationDAO	
Responsibilities: ▪ Provide data access operations for the Application entity, such as saving, retrieving, updating, and deleting application records in the underlying database. ▪ Support additional operations like finding an application by its ID.	**Collaborations:** ▪ Extends the JpaRepository interface, indicating a collaboration with the Spring Data JPA framework. ▪ Collaborates with the Application entity to perform database operations related to applications.

Class Name: ProfDAO	
Responsibilities: ▪ Provide data access operations for the Professor entity, such as saving, retrieving, updating, and deleting professor records in the underlying database. ▪ Support additional operations like	**Collaborations:** ▪ Extends the JpaRepository interface, indicating a collaboration with the Spring Data JPA framework. ▪ Collaborates with the Professor and User entities to perform database operations related to professors.

finding professors by their IDs or associated user IDs.	

Class Name: StdDAO	
Responsibilities: ▪ Provide data access operations for the Student entity, such as saving, retrieving, updating, and deleting student records in the underlying database. ▪ Support additional operations like finding students by their associated user IDs.	**Collaborations:** ▪ Extends the JpaRepository interface, indicating a collaboration with the Spring Data JPA framework. ▪ Collaborates with the Student and User entities to perform database operations related to students.

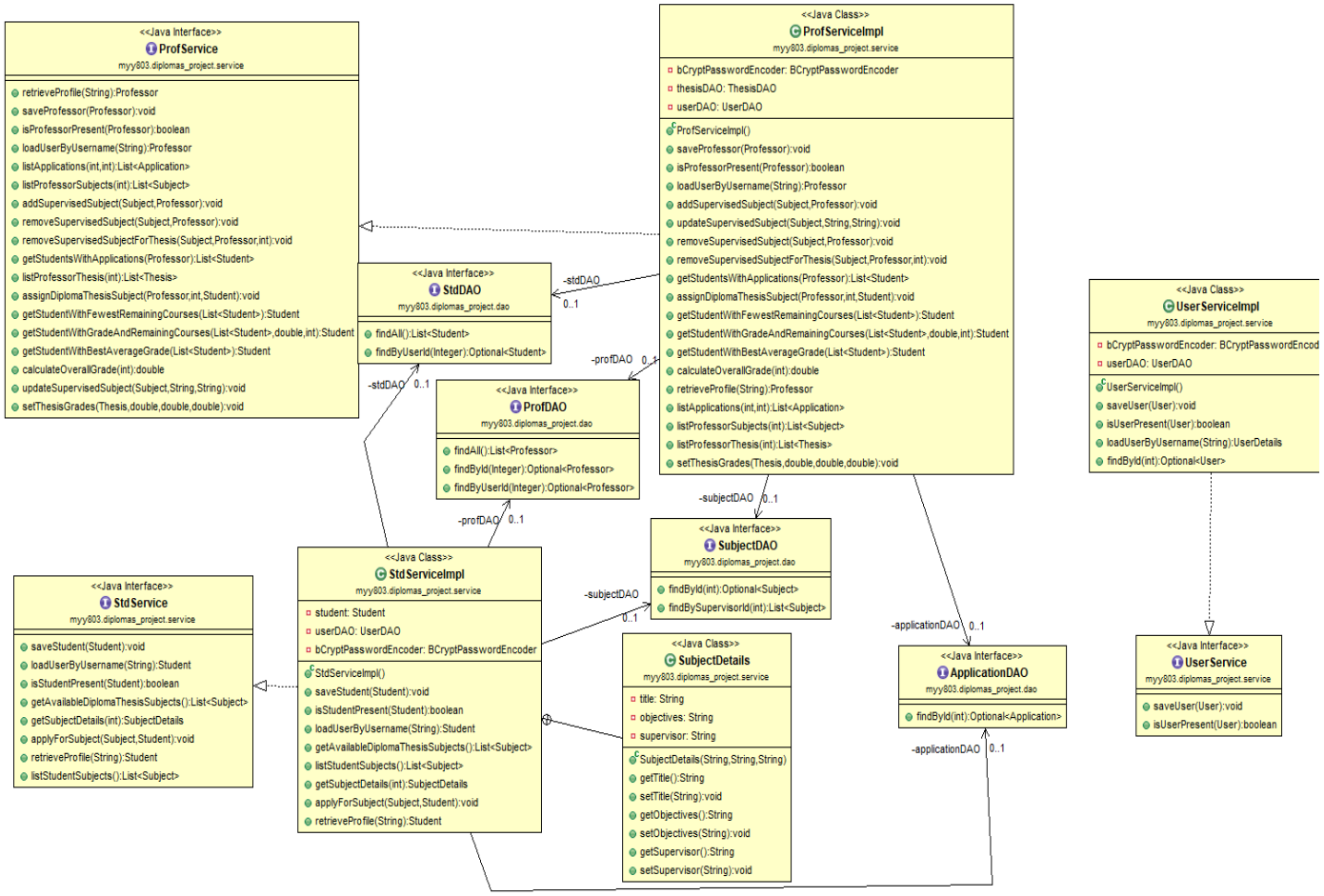
Class Name: SubjectDAO	
Responsibilities: ▪ Provide data access operations for the Subject entity, such as saving, retrieving, updating, and deleting subject records in the underlying database. ▪ Support additional operations like finding subjects by their IDs or associated supervisor IDs.	**Collaborations:** ▪ Extends the JpaRepository interface, indicating a collaboration with the Spring Data JPA framework. ▪ Collaborates with the Subject entity to perform database operations related to subjects.

Class Name: ThesisDAO	
Responsibilities: ▪ Provide data access operations for the Thesis entity, such as saving, retrieving, updating, and deleting thesis records in the underlying database. ▪ Support additional operations like finding theses by their IDs or	**Collaborations:** ▪ Extends the JpaRepository interface, indicating a collaboration with the Spring Data JPA framework. ▪ Collaborates with the Thesis entity to perform database operations related to theses.

associated supervisor IDs.	

Class Name: UserDAO	
Responsibilities: ▪ Provide data access operations for the User entity, such as saving, retrieving, updating, and deleting user records in the underlying database. ▪ Support additional operations like finding users by their IDs or usernames.	**Collaborations:** ▪ Extends the JpaRepository interface, indicating a collaboration with the Spring Data JPA framework. ▪ Collaborates with the User entity to perform database operations related to users.

Service class diagram:



SERVICE CLASS DIAGRAM

Class Name: ProfService	
Responsibilities: ▪ Retrieve the profile of a professor based on their username. ▪ Save a professor to the database. ▪ Check if a professor is already present in the database.	**Collaborations:** ▪ Collaborates with the Professor, Application, Subject, Student, and Thesis entities to perform various operations related to professors and their responsibilities. ▪ Annotates the class with @Service to

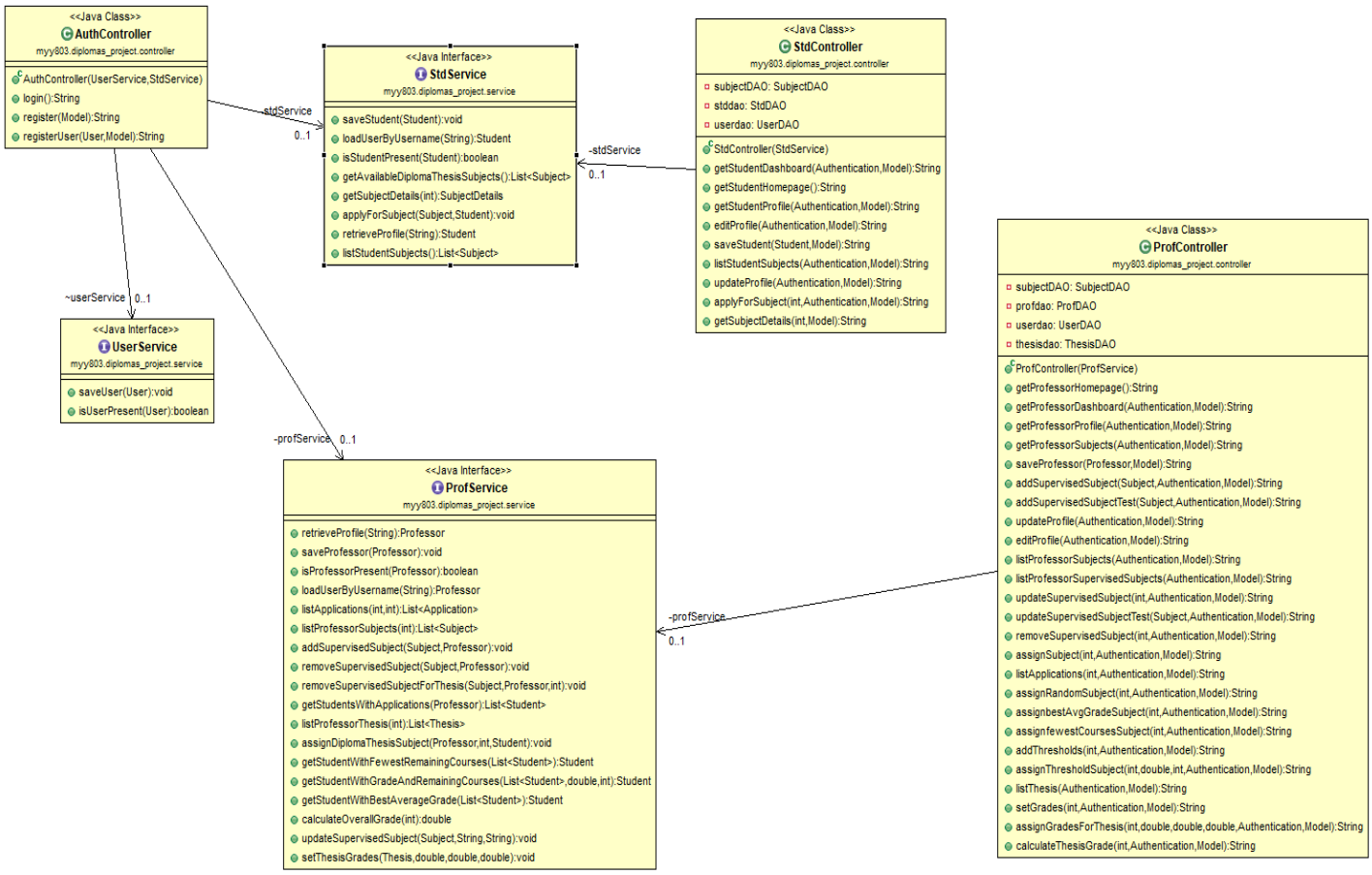
▪ Load a professor by their username for authentication purposes. ▪ List applications for a specific professor and subject. ▪ List subjects supervised by a professor. ▪ Add a supervised subject to a professor. ▪ Remove a supervised subject from a professor. ▪ Remove a supervised subject for a specific student from a professor. ▪ Get a list of students with applications for a specific professor. ▪ List theses supervised by a professor. ▪ Assign a diploma thesis subject to a professor and student. ▪ Determine the student with the fewest remaining courses among a list of students. ▪ Determine the student with a specific grade threshold and remaining course threshold among a list of students. ▪ Determine the student with the best average grade among a list of students. ▪ Calculate the overall grade for a thesis. ▪ Update the description and name of a supervised subject. ▪ Set grades for a thesis.	indicate it as a service component in the Spring framework.

Class Name: StdService	
Responsibilities: ▪ Save a student to the database. ▪ Load a student by their username for authentication purposes. ▪ Check if a student is already present in	**Collaborations:** ▪ Collaborates with the Student, Subject, and User entities to perform various operations related to students and their responsibilities. ▪ Annotates the class with @Service to

the database. ▪ Get a list of available diploma thesis subjects. ▪ Get the details of a subject by its ID. ▪ Apply for a subject as a student. ▪ Retrieve the profile of a student based on their username. ▪ List subjects of a student.	indicate it as a service component in the Spring framework.

Class Name: UserService	
Responsibilities: ▪ Save a user to the database. ▪ Check if a user is already present in the database.	**Collaborations:** ▪ Collaborates with the User entity to perform operations related to users. ▪ Annotates the class with @Service to indicate it as a service component in the Spring framework.

Controller class diagram:



CONTROLLER CLASS DIAGRAM

Class Name: AuthController	
Responsibilities: ▪ Handle login requests and return the login view. ▪ Handle register requests and return the registration view. ▪ Register a new user and handle the user registration process. ▪ Save a user to the database and handle the user registration success message. ▪ Determine the role of the registered user and create a corresponding	**Collaborations:** ▪ Collaborates with the UserService, StdService, ProfService, User, Student, and Professor entities to perform various operations related to user authentication and registration. ▪ Annotates the class with @Controller to indicate it as a controller component in the Spring framework.

student or professor entity. ▪ Collaborate with the UserService, StdService, User, Student, and Professor entities to perform user registration and role-specific operations.	

Class Name: StdController	
Responsibilities: ▪ Handle requests related to the student dashboard and return the corresponding view. ▪ Retrieve the currently logged-in student's profile and display the student information. ▪ Retrieve the currently logged-in student's profile for editing purposes. ▪ Save the updated student profile to the database. ▪ List available diploma thesis subjects for the student. ▪ Apply for a subject as a student. ▪ Retrieve the details of a specific subject. ▪ Collaborate with the StdService, SubjectDAO, StdDAO, UserDAO, User, and Subject entities to perform various operations related to students and their responsibilities.	**Collaborations:** ▪ Collaborates with the StdService, SubjectDAO, StdDAO, UserDAO, User, Student, and Subject entities to perform various operations related to students and their responsibilities. ▪ Annotates the class with @Controller to indicate it as a controller component in the Spring framework.

Class Name: ProfController	
Responsibilities: ▪ Handle requests related to the professor's homepage, dashboard, profile, and subjects	**Collaborations:** ▪ ProfService: Used to retrieve and update professor-related data ▪ SubjectDAO: Used to interact with the

▪ Save and update professor's information ▪ Add, update, and remove supervised subjects ▪ Assign subjects to professors ▪ Handle requests related to applications, subject assignments, thresholds, and thesis management ▪ List applications for a specific subject ▪ Assign subjects to students using different strategies (random, best average grade, fewest courses, thresholds) ▪ Add and assign thresholds for subject assignments ▪ List and manage supervised theses ▪ Set and calculate grades for theses	subject data access object ▪ ProfDAO: Used to interact with the professor data access object ▪ UserDAO: Used to interact with the user data access object ▪ ThesisDAO: Used to interact with the thesis data access object ▪ Authentication: Used to retrieve the current user's authentication information ▪ Application: Represents an application for a subject ▪ TemplateStrategyAlgorithm: Defines the template strategy algorithm interface for subject assignments ▪ RandomChoiceStrategy: Implements the random choice strategy for subject assignments ▪ BestAvgGradeStrategy: Implements the best average grade strategy for subject assignments ▪ FewestCoursesStrategy: Implements the fewest courses strategy for subject assignments ▪ ThresholdsStrategy: Implements the thresholds strategy for subject assignments ▪ Student: Represents a student ▪ Thesis: Represents a thesis ▪ Model: Used to pass data to the view