Diploma Projects Management App

Sprint Report

Argyrios Zezos 4588

Fotios Pappas 4773

Hlias Dimopoulos 4869

VERSIONS HISTORY

Date	Version	Description	Author
15/4/2023	1.0	Basic implementation	All

1 Introduction

This document provides information concerning the **1st, 2nd** and 3rd sprint of the project.

1.1 Purpose

The purpose is to document the project’s design and development.

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	Fotios Pappas
Scrum Master	Hlias Dhmopoulos
Development Team	Argyrios Zezos

2.2 Sprints

List of the sprints performed and the user stories realized in each Sprint

Sprint No	Begin Date	End Date	Number of weeks	User stories
1	**1/3**	**15/3**	**2**	**User's user stories**
2	**15/3**	**1/4**	**2**	**Professor's user stories**
3	**1/4**	**15/4**	**2**	**Student's user stories**

3 Use Cases

Specification of the concrete Use Cases that describe the interaction of the user with the applications, as derived from the abstract user stories.

3.1 Create user account

Use case ID	UC1
Actors	User
Pre conditions	The user has not created an account on the application before.
Main flow of events	1. The user opens the application and selects the option to create a new account. 2. The system presents a form that allows the user to select their role (STUDENT or PROFESSOR), enter a username, and enter a password. 3. The user selects their role, enters a username, enters a password, and submits the form. 4. The system verifies the uniqueness of the username and password. 5. If the username and password are unique, the system creates a new user account with the entered information. 6. The system logs in the user and grants them access to the application.
Alternative flow 1	If the user encounters an error while entering their information, such as a non-unique username or password that does not meet the password requirements, the system displays an error message and prompts the user to correct the input.
Alternative flow 2	If the user wants to cancel the account creation process, they can select the option to cancel and return to the login screen.
Post conditions	The user has created a new account on the application and can log in with their username and password to access the application according to their role.

3.2 Login user

Use case ID	UC2
Actors	User
Pre conditions	The user has created an account on the application and has valid login credentials (username and password).
Main flow of events	1. The user opens the application and navigates to the login screen. 2. The user enters their username and password into the login form. 3. The system verifies the user's login credentials and logs the user in. 4. The system grants the user access to the application's functionalities.
Alternative flow 1	If the user enters an incorrect username or password, the system displays an error message and prompts the user to enter the correct credentials.
Alternative flow 2	If the user forgets their password, they can select the "Forgot Password" option and follow the steps to reset their password.
Post conditions	The user has successfully logged in to the application and can use the provided functionalities according to their user role.

3.3 Manage student profile

Use case ID	UC3
Actors	Student
Pre conditions	The student has logged in to the application and has access to their profile page.
Main flow of events	1. The student navigates to their profile page and selects the option to edit their personal information. 2. The system presents a form that allows the student to enter their full name, year of studies, current average grade, and number of remaining courses for graduation. 3. The student enters their personal information and submits the form. 4. The system saves the updated personal information to the student's profile.
Alternative flow 1	If the student encounters an error while entering their information, such as an invalid grade or an empty field, the system displays an error message and prompts the student to correct the input.
Alternative flow 2	If the student wants to cancel the profile editing process, they can select the option to cancel and return to their profile page.
Post conditions	The student has updated their personal information on their profile, which can be viewed by professors when evaluating their application to available diploma thesis subjects.

3.4 View available subjects

Use case ID	UC4
Actors	Student
Pre conditions	The student has logged in to the application and has access to the list of available diploma thesis subjects.
Main flow of events	1. The student navigates to the list of available diploma thesis subjects. 2. The system displays a list of diploma thesis subjects offered by the professors. 3. The student views the list of available subjects and selects the ones that look interesting. 4. The student applies to the selected subjects by submitting their application to the corresponding professors.
Alternative flow 1	If there are no available diploma thesis subjects, the system displays a message indicating that there are no subjects currently available.
Alternative flow 2	If the student encounters an error while submitting their application, such as submitting an incomplete application, the system displays an error message and prompts the student to correct the input.
Post conditions	The student has viewed the list of available diploma thesis subjects and has applied to the ones that look interesting. The corresponding professors can review the student's application and decide whether to accept or reject it.

3.5 View subject details

Use case ID	UC5
Actors	Student
Pre conditions	The student has logged in to the application and has navigated to the list of available diploma thesis subjects.
Main flow of events	1. The student selects a specific diploma thesis subject from the list. 2. The system displays a more detailed description of the selected thesis subject, including its title, objectives, and supervisor. 3. The student reads the detailed description and decides whether to apply for the thesis or not.
Alternative flow 1	If there is no detailed description available for the selected thesis subject, the system displays a message indicating that there is no description available.
Alternative flow 2	If the student encounters an error while viewing the detailed description, such as a technical issue, the system displays an error message and prompts the student to try again later.
Post conditions	The student has viewed a more detailed description of the selected diploma thesis subject and has decided whether to apply for the thesis or not.

3.6 Apply to subject

Use case ID	UC6
Actors	Student
Pre conditions	The student has logged in to the application and has navigated to the list of available diploma thesis subjects.
Main flow of events	1. The student selects a specific diploma thesis subject from the list. 2. The system displays a more detailed description of the selected thesis subject, including its title, objectives, and supervisor. 3. The student decides to apply for the thesis subject and selects the option to submit their application. 4. The system presents a form that allows the student to enter their personal information, such as their full name, email address, and a brief statement of their interest in the thesis subject. 5. The student enters their personal information and submits the application form. 6. The system sends a notification to the corresponding professor indicating that the student has applied for their diploma thesis subject.
Alternative flow 1	If the student encounters an error while submitting their application, such as submitting an incomplete application or entering invalid personal information, the system displays an error message and prompts the student to correct the input.
Alternative flow 2	If there is no available supervisor for the selected diploma thesis subject, the system displays a message indicating that there is no supervisor available.
Post conditions	The student has applied for a diploma thesis subject by submitting their personal information and statement of interest to the corresponding professor. The professor can review the student's application and decide whether to accept or reject it.

3.7 Set professor profile information

Use case ID	UC7
Actors	Professor
Pre conditions	The professor has logged in to the application and has access to their personal profile settings.
Main flow of events	1. The professor navigates to their personal profile settings. 2. The system displays a form that allows the professor to enter their personal information, such as their full name and specialty. 3. The professor enters their personal information and submits the form. 4. The system updates the professor's personal profile information.
Alternative flow 1	If the professor encounters an error while submitting their personal information, such as entering invalid data, the system displays an error message and prompts the professor to correct the input.
Alternative flow 2	-
Post conditions	The professor has set their personal profile information, which includes their full name and specialty. Students can view the professor's personal profile information before applying to their diploma thesis subjects to learn more about the professor's expertise and qualifications.

3.8 View offered subjects

Use case ID	UC8
Actors	Professor
Pre conditions	The professor has logged in to the application and has access to their account dashboard.
Main flow of events	1. The professor navigates to the section that displays the list of available diploma thesis subjects. 2. The system displays a list of available diploma thesis subjects that the professor offers, including their titles and brief descriptions. 3. The professor can select a specific thesis subject from the list to manage its related information, such as the list of students who have applied to it, their personal information, and the status of their applications.
Alternative flow 1	If there are no available diploma thesis subjects offered by the professor, the system displays a message indicating that there are no subjects available.
Alternative flow 2	-
Post conditions	The professor has accessed the list of available diploma thesis subjects that they offer and can manage related information, such as the list of students who have applied to the subjects and the status of their applications.

3.9 Add subject

Use case ID	UC9
Actors	Professor
Pre conditions	The professor has logged in to the application and has access to their account dashboard.
Main flow of events	1. The professor navigates to the section that allows them to add a new diploma thesis subject to their list. 2. The system displays a form that allows the professor to enter the title and objectives of the new thesis subject. 3. The professor enters the required information and submits the form. 4. The system adds the new diploma thesis subject to the professor's list of available subjects.
Alternative flow 1	If the professor encounters an error while adding a new diploma thesis subject, such as submitting an incomplete form or entering invalid data, the system displays an error message and prompts the professor to correct the input.
Alternative flow 2	.....
Post conditions	The professor has added a new diploma thesis subject to their list, which includes the title and objectives of the subject. The subject is now available for students to view and apply to.

3.10 Delete subject

Use case ID	UC10
Actors	Actor(s) related to the use case
Pre conditions	The professor has logged in to the application and has access to their account dashboard.
Main flow of events	1. The professor navigates to the section that allows them to manage their list of diploma thesis subjects. 2. The system displays the professor's list of available thesis subjects. 3. The professor selects the diploma thesis subject they want to delete. 4. The system prompts the professor to confirm the deletion. 5. The professor confirms the deletion. 6. The system removes the selected diploma thesis subject from the professor's list.
Alternative flow 1	If the professor selects a diploma thesis subject that has students who have already applied to it, the system displays a warning message that deleting the subject will also delete all related applications and prompts the professor to confirm the deletion.
Alternative flow 2	.....
Post conditions	The professor has deleted a diploma thesis subject from their list, and it is no longer available for students to view or apply to. The professor's list of available subjects is now up to date.

3.11 View subject applications

Use case ID	Unique ID for the the use case
Actors	Professor
Pre conditions	The professor has logged in to the application and has access to their account dashboard.
Main flow of events	1. The professor navigates to the section that allows them to view applications for diploma thesis subjects. 2. The system displays a list of diploma thesis subjects that have received applications from students. 3. The professor selects the diploma thesis subject they want to view the applications for. 4. The system displays a list of students who have applied for the selected diploma thesis subject. 5. The professor reviews the applications and makes a decision on which student to assign the diploma thesis subject to.
Alternative flow 1	If no students have applied for the selected diploma thesis subject, the system displays a message indicating that there are no applications to review.
Alternative flow 2	.....
Post conditions	The professor has reviewed applications for a diploma thesis subject.

3.12 Assign project

Use case ID	UC12
Actors	Professor
Pre conditions	The professor has logged in to the application and has access to their account dashboard. The professor has received applications from students for the diploma thesis subject they want to assign.
Main flow of events	1. The professor navigates to the section that allows them to assign a diploma thesis subject to a student. 2. The system displays a list of diploma thesis subjects that have received applications from students. 3. The professor selects the diploma thesis subject they want to assign. 4. The system displays a list of students who have applied for the selected diploma thesis subject. 5. The professor selects a strategy for assigning the diploma thesis subject. 6. The system applies the selected strategy to the list of students and selects the student who will be assigned the diploma thesis subject. 7. The system notifies the selected student that they have been assigned the diploma thesis subject. 8. The student can now start working on the diploma thesis subject.
Alternative flow 1	If no students have applied for the selected diploma thesis subject, the system displays a message indicating that there are no students to assign the subject to.
Alternative flow 2	.....
Post conditions	The professor has assigned a diploma thesis subject to a student according to the selected strategy. The selected student has been notified and can start working on the subject.

3.13 View theses

Use case ID	UC13
Actors	Professor
Pre conditions	The professor has logged in to the application and has access to their account dashboard. The professor has previously assigned diploma thesis subjects to students.
Main flow of events	1. The professor navigates to the section that allows them to view the list of assigned diploma thesis projects. 2. The system displays a list of diploma thesis projects that have been assigned to the professor's supervision. 3. The professor can view detailed information about each project, such as the title, objectives, assigned student, and project status. 4. The professor can manage related information, such as progress reports, meetings, and feedback, for each assigned project.
Alternative flow 1	If no diploma thesis projects have been assigned to the professor's supervision, the system displays a message indicating that there are no projects to view.
Alternative flow 2	.....
Post conditions	The professor has viewed the list of assigned diploma thesis projects and managed related information for each project.

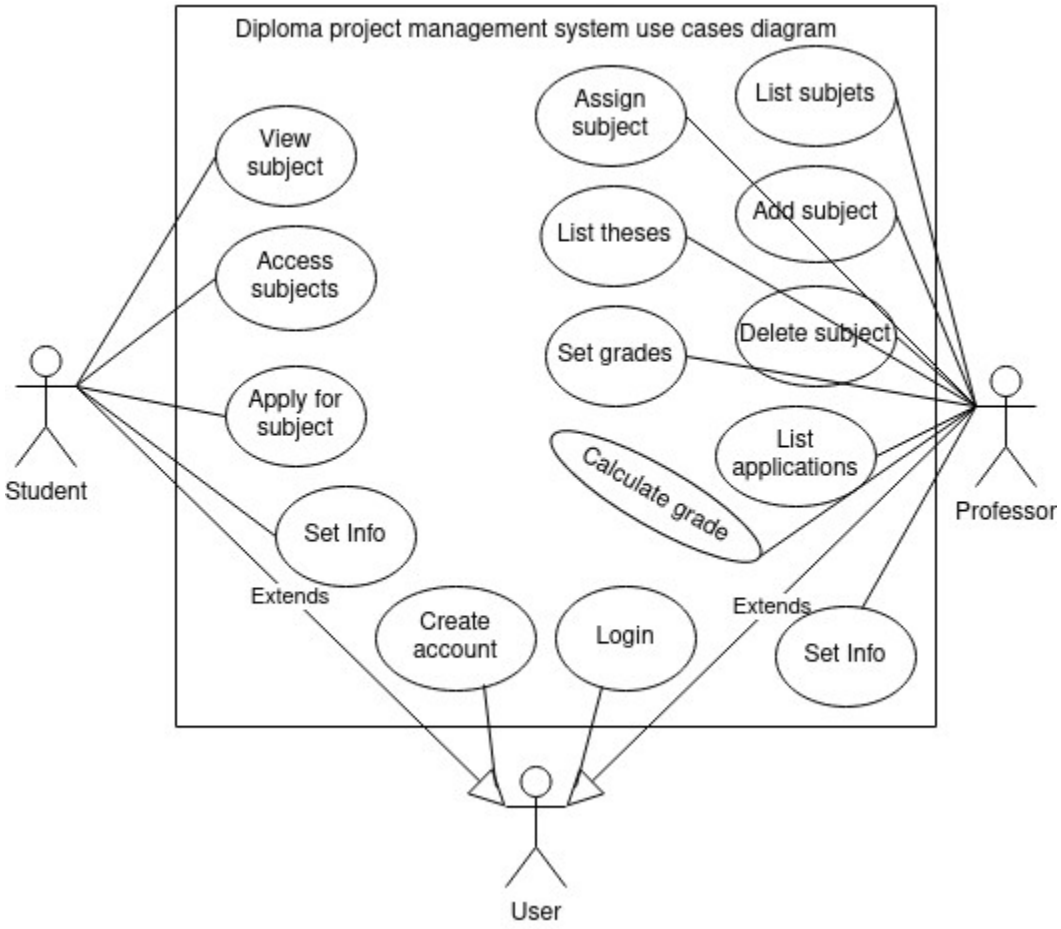
3.14 Set thesis grades

Use case ID	UC14
Actors	Professor
Pre conditions	The professor has logged in to the application and has access to their account dashboard. The professor has assigned diploma thesis subjects to students and the students have completed their work on the projects.
Main flow of events	1. The professor navigates to the section that allows them to set the grades for the implementation, the report, and the presentation of a diploma thesis subject. 2. The system displays a list of diploma thesis projects that have been assigned to the professor's supervision and have been completed by the students. 3. The professor selects a project from the list and sets the grades for the implementation, the report, and the presentation. 4. The professor can view detailed information about the project, such as the title, objectives, assigned student, and project status. 5. The system saves the grades and updates the project status accordingly.
Alternative flow 1	If the professor tries to set grades for a project that is not completed yet, the system displays an error message.
Alternative flow 2	-
Post conditions	The professor has set the grades for the implementation, the report, and the presentation of a diploma thesis subject that they supervise, and the project status has been updated accordingly.

3.15 Calculate thesis grade

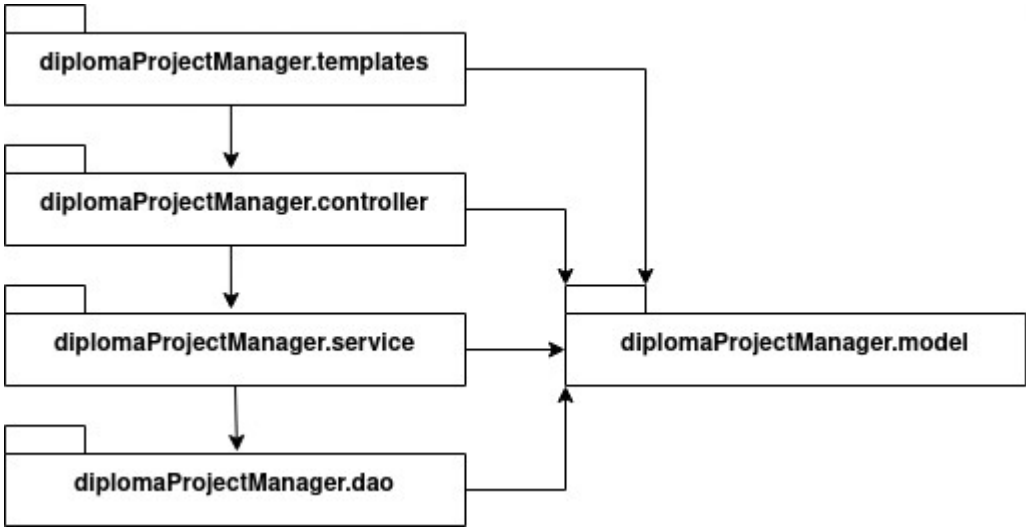
Use case ID	Unique ID for the the use case
Actors	Professor
Pre conditions	The professor has logged in to the application and has access to their account dashboard. The professor has set the grades for the implementation, report, and presentation of a diploma thesis project that they supervise.
Main flow of events	1. The professor navigates to the section that allows them to calculate the overall grade for a diploma thesis project. 2. The system displays a list of diploma thesis projects that have been assigned to the professor's supervision and have been completed by the students. 3. The professor selects a project from the list for which they want to calculate the overall grade. 4. The system retrieves the grades for the implementation, report, and presentation for the selected project. 5. The professor enters the weights for each component of the final grade, according to the provided formula. 6. The system calculates the overall grade for the project based on the provided weights and displays it to the professor. 7. The professor can view detailed information about the project, such as the title, objectives, assigned student, and project status.
Alternative flow 1	If the professor tries to calculate the overall grade for a project that is not completed yet, the system displays an error message.
Alternative flow 2	-
Post conditions	The professor has calculated the overall grade for a diploma thesis project that they supervise, based on the provided formula, and can use this information to evaluate the work done by the student.

3.16 Use case diagram



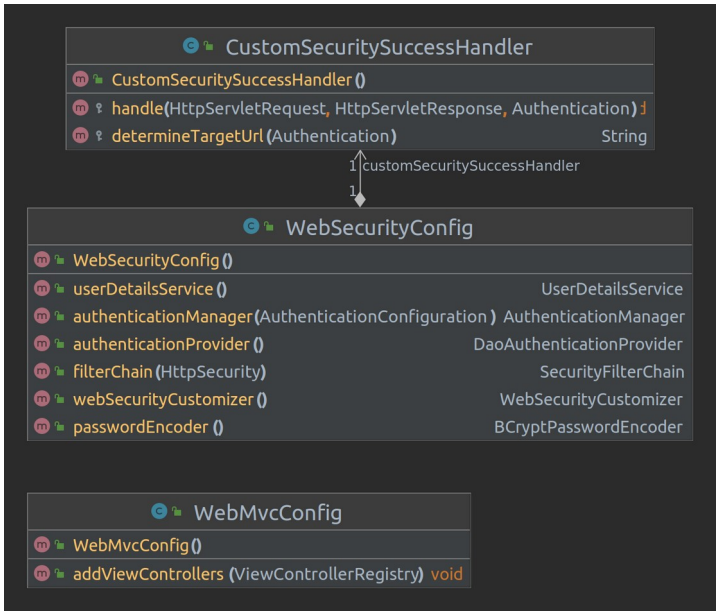
4 Design

4.1 Architecture

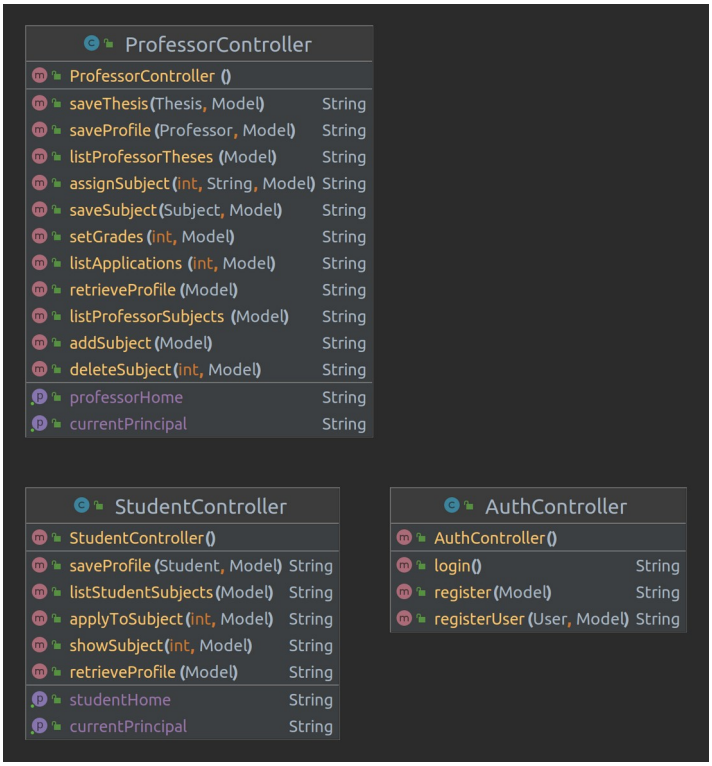


Package diagram

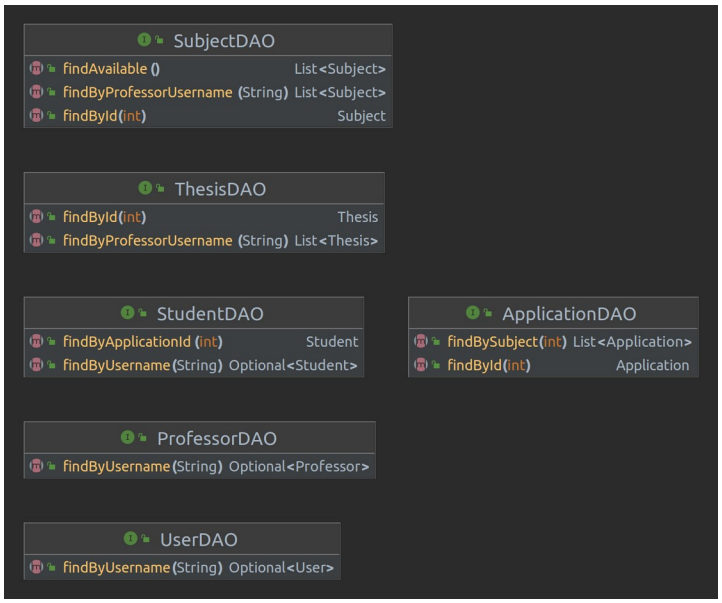
4.2 Design



Package config class diagram

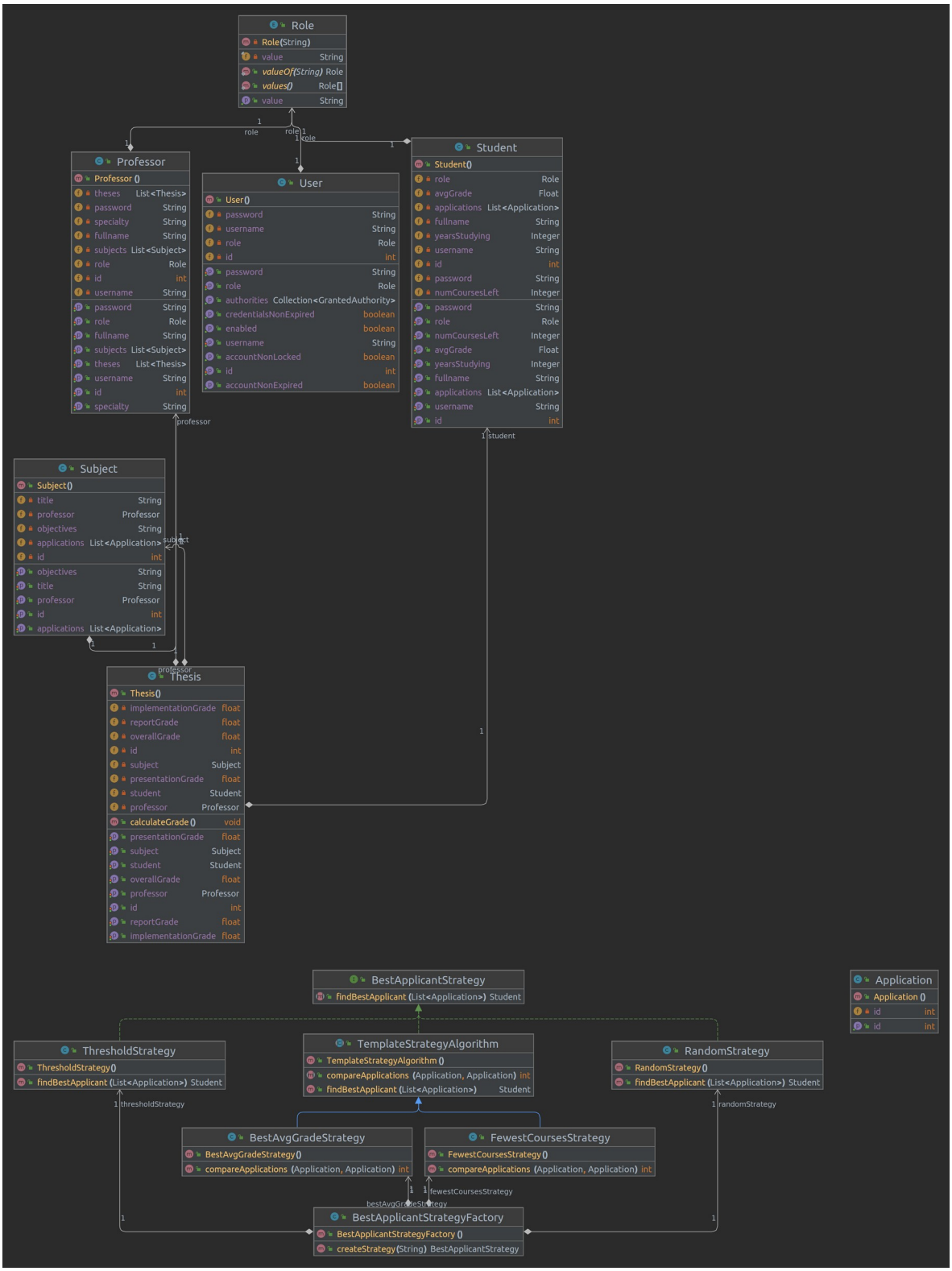


Package controller class diagram



Package dao class diagram

Package model class diagram



Package model class diagram

5 CRC cards

Class Name: DiplomaProjectManagerApplication	
Responsibilities: ▪ Initialize and run the Spring Boot application. ▪ Serve as the entry point for the Diploma Project Manager application.	**Collaborations:** ▪ SpringApplication: used to run the Spring Boot application.

Class Name: CustomSecuritySuccessHandler	
Responsibilities: ▪ Handle authentication success. ▪ Determine the appropriate target URL based on the user's role. ▪ Redirect the user to the appropriate dashboard based on their role.	**Collaborations:** ▪ HttpServletRequest: used to get information about the request made by the user. ▪ HttpServletResponse: used to send a response to the user. ▪ Authentication: used to retrieve information about the user's authentication. ▪ GrantedAuthority: used to represent a granted authority, such as a role.

Class Name: WebMvcConfig	
Responsibilities: ▪ Configure view controllers for the Spring MVC application. ▪ Map the URL "/" to the "homepage" view.	**Collaborations:** ▪ ViewControllerRegistry: used to register view controllers for the Spring MVC application.

Class Name: WebSecurityConfig	
Responsibilities: ▪ Configure Spring Security settings for the application. ▪ Define authorization rules for specific URLs. ▪ Define a custom authentication success handler. ▪ Define a custom authentication provider. ▪ Define a security filter chain for the application.	**Collaborations:** ▪ CustomSecuritySuccessHandler: used to handle successful authentication. ▪ UserServiceImpl: implementation of the UserDetailsService interface. ▪ BCryptPasswordEncoder: used to encode user passwords. ▪ AuthenticationConfiguration: used to configure the authentication manager. ▪ AuthenticationManager: used to authenticate user credentials. ▪ DaoAuthenticationProvider: used to retrieve user details and credentials from the database. ▪ HttpSecurity: used to configure HTTP security settings. ▪ SecurityFilterChain: used to define a chain of filters to process incoming requests.

Class Name: AuthController	
Responsibilities: ▪ Control user authentication and authorization ▪ Handle user registration ▪ Map HTTP requests to appropriate methods for login, registration, and saving a user ▪ Retrieve and manipulate data through the UserService object ▪ Return appropriate views based on user actions and data	**Collaborations:** ▪ User: The User model used for registration and authentication ▪ UserService: The service layer for accessing and manipulating User data ▪ Model: Used for passing data between controller and view ▪ auth/signin.html: The view for the login page ▪ auth/signup.html: The view for the registration page

Class Name: ProfessorController	
Responsibilities: ▪ Manage professor's profile ▪ Manage professor's subjects ▪ Manage professor's applications ▪ Manage professor's theses ▪ Retrieve current logged in user's name	**Collaborations:** ▪ ProfessorService ▪ SubjectService ▪ Authentication ▪ SecurityContextHolder

Class Name: StudentController	
Responsibilities: ▪ Handle HTTP requests related to student profile management and subject selection. ▪ Retrieve and store student profile information. ▪ Retrieve list of available subjects for a student to apply to. ▪ Apply a student to a specific subject. ▪ Show subject details to the student.	**Collaborations:** ▪ StudentService: provides methods to retrieve and store student information. ▪ SubjectService: provides methods to retrieve subject information. ▪ Model: used to pass data to the view layer. ▪ Authentication and SecurityContextHolder: used to retrieve the current authenticated user's name.

Class Name: ApplicationDAO	
Responsibilities: ▪ Accessing and managing Application data in a database through Spring Data JPA repository ▪ Defining custom query methods to retrieve Application data	**Collaborations:** ▪ Application: The model class representing an application object ▪ JpaRepository: A Spring Data interface that provides generic CRUD (create, read, update, delete) operations for a specific domain object type ▪ List: A Java interface that defines an ordered collection of elements ▪ Query: A Spring Data annotation that declares a custom query to retrieve data from the database ▪ Repository: A Spring stereotype annotation that marks the DAO as a bean and allows it to be automatically discovered by component scanning

Class Name: ProfessorDAO	
Responsibilities: ▪ Accessing and managing Professor data in a database through Spring Data JPA repository ▪ Defining custom query methods to retrieve Professor data	**Collaborations:** ▪ Professor: The model class representing a professor object ▪ JpaRepository: A Spring Data interface that provides generic CRUD operations ▪ Optional: A Java interface that represents a container object that may or may not contain a non-null value ▪ Repository: A Spring stereotype annotation that marks the DAO as a bean and allows it to be automatically discovered by component scanning

Class Name: StudentDAO	
Responsibilities: ▪ Accessing and managing Student data in a database through Spring Data JPA repository ▪ Defining custom query methods to retrieve Student data	**Collaborations:** ▪ Student: The model class representing a student object ▪ JpaRepository: A Spring Data interface that provides generic CRUD (create, read, update, delete) operations for a specific domain object type ▪ Optional: A Java interface that represents a container object that may or may not contain a non-null value ▪ Query: A Spring Data annotation that declares a custom query to retrieve data from the database ▪ Repository: A Spring stereotype annotation that marks the DAO as a bean and allows it to be automatically discovered by component scanning

Class Name: SubjectDAO	
Responsibilities: ▪ Accessing and managing Student data in a database through Spring Data JPA repository ▪ Defining custom query methods to retrieve Subject data	**Collaborations:** ▪ Subject: The model class representing a subject object ▪ JpaRepository: A Spring Data interface that provides generic CRUD (create, read, update, delete) operations for a specific domain object type ▪ Query: A Spring Data annotation that declares a custom query to retrieve data from the database

	▪ Repository: A Spring stereotype annotation that marks the DAO as a bean and allows it to be automatically discovered by component scanning

Class Name: ThesisDAO	
Responsibilities: ▪ Accessing and managing Student data in a database through Spring Data JPA repository ▪ Defining custom query methods to retrieve Thesis data	**Collaborations:** ▪ Thesis: The model class representing a student object ▪ JpaRepository: A Spring Data interface that provides generic CRUD (create, read, update, delete) operations for a specific domain object type ▪ Query: A Spring Data annotation that declares a custom query to retrieve data from the database ▪ Repository: A Spring stereotype annotation that marks the DAO as a bean and allows it to be automatically discovered by component scanning

Class Name: UserDAO	
Responsibilities: ▪ Accessing and managing User data in a database through Spring Data JPA repository ▪ Defining custom query methods to retrieve User data	**Collaborations:** ▪ User: The model class representing a student object ▪ JpaRepository: A Spring Data interface that provides generic CRUD (create, read, update, delete) operations for a specific domain object type

	▪ Optional: A Java interface that represents a container object that may or may not contain a non-null value ▪ Query: A Spring Data annotation that declares a custom query to retrieve data from the database ▪ Repository: A Spring stereotype annotation that marks the DAO as a bean and allows it to be automatically discovered by component scanning

Class Name: Application	
Responsibilities: ▪ Represent an application in the diploma project management system. ▪ Manage the application's unique identifier.	**Collaborations:** ▪ None

Class Name: Professor	
Responsibilities: ▪ Store information about a professor, such as their username, password, role, full name, and specialty. ▪ Manage the professor's theses and subjects. ▪ Provide methods for accessing and modifying the professor's information and related theses and subjects.	**Collaborations:** ▪ Thesis: the professor's theses that they supervise. ▪ Subject: the professor's subjects that they teach. ▪ Role: an enumeration of roles that the professor can have. ▪ Database: where the professor's information and related theses and subjects are stored.

Class Name: Role	
Responsibilities: ▪ Represent the role of a user in the system. ▪ Provide the value of the role as a string.	**Collaborations:** ▪ User: a user of the system.

Class Name: Student	
Responsibilities: ▪ Represent a student in the system. ▪ Provide getters and setters for the student's attributes. ▪ Store applications made by the student.	**Collaborations:** ▪ Application: an application made by a student.

Class Name: Subject	
Responsibilities: ▪ Represent a subject in the diploma project management system. ▪ Store the subject's ID, title, objectives, professor and applications. ▪ Provide getters and setters for the attributes.	**Collaborations:** ▪ Professor: a subject has one professor who is responsible for supervising the diploma projects related to the subject. ▪ Application: a subject can have multiple applications, which are submitted by students for the diploma project.

Class Name: Thesis	
Responsibilities: ▪ Represent a thesis in the diploma project management system. ▪ Store the thesis's ID, overall grade, implementation grade, report grade, presentation grade, subject, student, and professor. ▪ Provide getters and setters for the attributes. ▪ Calculate the overall grade of the thesis based on the implementation, report, and presentation grades.	**Collaborations:** ▪ Subject: a thesis is related to one subject. ▪ Student: a thesis is written by one student. ▪ Professor: a thesis is evaluated by one professor.

Class Name: User	
Responsibilities: ▪ Store user information (id, username, password, role) ▪ Implement the UserDetails interface ▪ Provide accessors and mutators for the user information ▪ Provide a method to get the user's authorities ▪ Implement methods to determine if the user's account is non-expired, non-locked, non-expired credentials, and enabled	**Collaborations:** ▪ GrantedAuthority: Interface to represent user's roles/authorities in the system ▪ Role: Enum to represent user's role in the system.

Class Name: BestApplicantStrategy	
Responsibilities: ▪ Find the best applicant from a list of applications	**Collaborations:** ▪ List<Application>: a list of applications to search for the best applicant ▪ Student: the best applicant to be returned

Class Name: BestApplicantStrategyFactory	
Responsibilities: ▪ Create a strategy based on a string passed as an argument. ▪ Use the Autowired annotation to instantiate the different strategies available.	**Collaborations:** ▪ RandomStrategy: A class that implements the BestApplicantStrategy interface and represents a strategy to select a random student. ▪ BestAvgGradeStrategy: A class that implements the BestApplicantStrategy interface and represents a strategy to select a student with the best average grade. ▪ FewestCoursesStrategy: A class that implements the BestApplicantStrategy interface and represents a strategy to select a student with the fewest courses. ▪ ThresholdStrategy: A class that implements the BestApplicantStrategy interface and represents a strategy to select a student with a grade above a certain threshold.

Class Name: `RandomStrategy`	
Responsibilities: ▪ Implement the `BestApplicantStrategy` interface. ▪ Select a random applicant from a list of applications. ▪ Retrieve the student data from the `StudentDAO` using the selected application's ID.	**Collaborations:** ▪ `BestApplicantStrategy`: Implements the interface. ▪ `StudentDAO`: Collaborates to retrieve the student data using the application's ID.

Class Name: `ThresholdStrategy`	
Responsibilities: ▪ Implement the `BestApplicantStrategy` interface to define a threshold strategy for selecting the best applicant. ▪ Find the student who meets the threshold criteria with the highest priority, where priority is given to the earliest such student.	**Collaborations:** ▪ `StudentDAO`: Accesses the student object in the database.

Class Name: `TemplateStrategyAlgorithm`	
Responsibilities: ▪ Provides a template for implementing different best applicant strategies. ▪ Defines the algorithm for selecting the best applicant from a list of applications.	**Collaborations:** ▪ `BestApplicantStrategy`: The interface that this class implements. ▪ `StudentDAO`: A data access object for retrieving student information.

Class Name: `BestAvgGradeStrategy`	
Responsibilities: ▪ Implements a strategy for comparing applications based on the average grade of the student associated with each application. ▪ Uses a DAO (`StudentDAO`) to retrieve the student information from the database.	**Collaborations:** ▪ `TemplateStrategyAlgorithm`: Inherits from this class and implements the `compareApplications` method. ▪ `StudentDAO`: Uses this DAO to retrieve the student information.

Class Name: `FewestCoursesStrategy`	
Responsibilities: ▪ Implements a strategy for comparing applications based on the number of courses left for the student associated with each application. ▪ Uses a DAO (`StudentDAO`) to retrieve the student information from the database.	**Collaborations:** ▪ `TemplateStrategyAlgorithm`: Inherits from this class and implements the `compareApplications` method. ▪ `StudentDAO`: Uses this DAO to retrieve the student information.

Class Name: `ProfessorServiceImpl`	
Responsibilities: ▪ `retrieveProfile(String username)`: retrieves the profile of a professor based on their username. ▪ `saveProfile(Professor professor)`: saves the profile of a professor. ▪ `listProfessorSubjects(String username)`: returns a list of subjects taught by the professor. ▪ `addSubject(String username)`: creates a new subject and associates it with the professor. ▪ `listApplications(int subjectId)`: returns a list of applications for a subject. ▪ `listProfessorTheses(String username)`: returns a list of theses supervised by the professor. ▪ `assignSubject(String username, String strategyName, int subjectId)`: assigns a subject to a student based on a given strategy. ▪ `retrieveThesis(int thesisId)`: retrieves the thesis based on the given thesis id. ▪ `saveThesis(Thesis thesis)`: saves the thesis.	**Collaborations:** ▪ `ProfessorDAO`: used to retrieve and save the profile of a professor. ▪ `SubjectDAO`: used to retrieve and save subjects. ▪ `ApplicationDAO`: used to retrieve a list of applications for a subject. ▪ `ThesisDAO`: used to retrieve and save theses. ▪ `BestApplicantStrategyFactory`: used to create a strategy for selecting the best applicant for a subject.

Class Name: `StudentServiceImpl`	
Responsibilities: ▪ `saveProfile(Student student)`: saves the profile of a student. ▪ `retrieveProfile(String username)`: retrieves the profile of a student based on their username. ▪ `listStudentSubjects()`: returns a list of subjects available for the student. ▪ `applyToSubject(String username, int subjectId)`: applies to a subject by creating a new application.	**Collaborations:** ▪ `StudentDAO`: used to retrieve and save the profile of a student. ▪ `SubjectService`: used to retrieve a list of available subjects for the student. ▪ `ApplicationDAO`: used to save the application.

Class Name: User`ServiceImpl`	
Responsibilities: ▪ Save a new user to the system by encoding their password using a BcryptPasswordEncoder. ▪ Check if a user is already present in the system. ▪ Load user details by their username for authentication purposes. ▪ Retrieve a user from the system by their id.	**Collaborations:** ▪ BCryptPasswordEncoder: A Spring Security class used for encoding passwords. ▪ UserDAO: A data access object used for accessing the User entity in the database. ▪ User: An entity representing a user in the system. ▪ UserDetails: An interface from Spring Security that represents a user's details for authentication purposes.