Traineeship Application

Sprint Report

VERSION <1.0>

Georgios Dimoudis 5212

Georgios Strouggis 5357

Omiros Chatziiordanis 5388

VERSIONS HISTORY

Date	Version	Description	Author
2025-05-26	1.0	The first complete version of the traineeship app.	Dimoudis Georgios, Strouggis Georgios, Chatziiordanis Omiros

1 Introduction

1.1 Purpose

This document provides information concerning the **1.0** sprint of the project.

1.2 Document Structure

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

2 Scrum team and Sprint Backlog

2.1 Scrum team

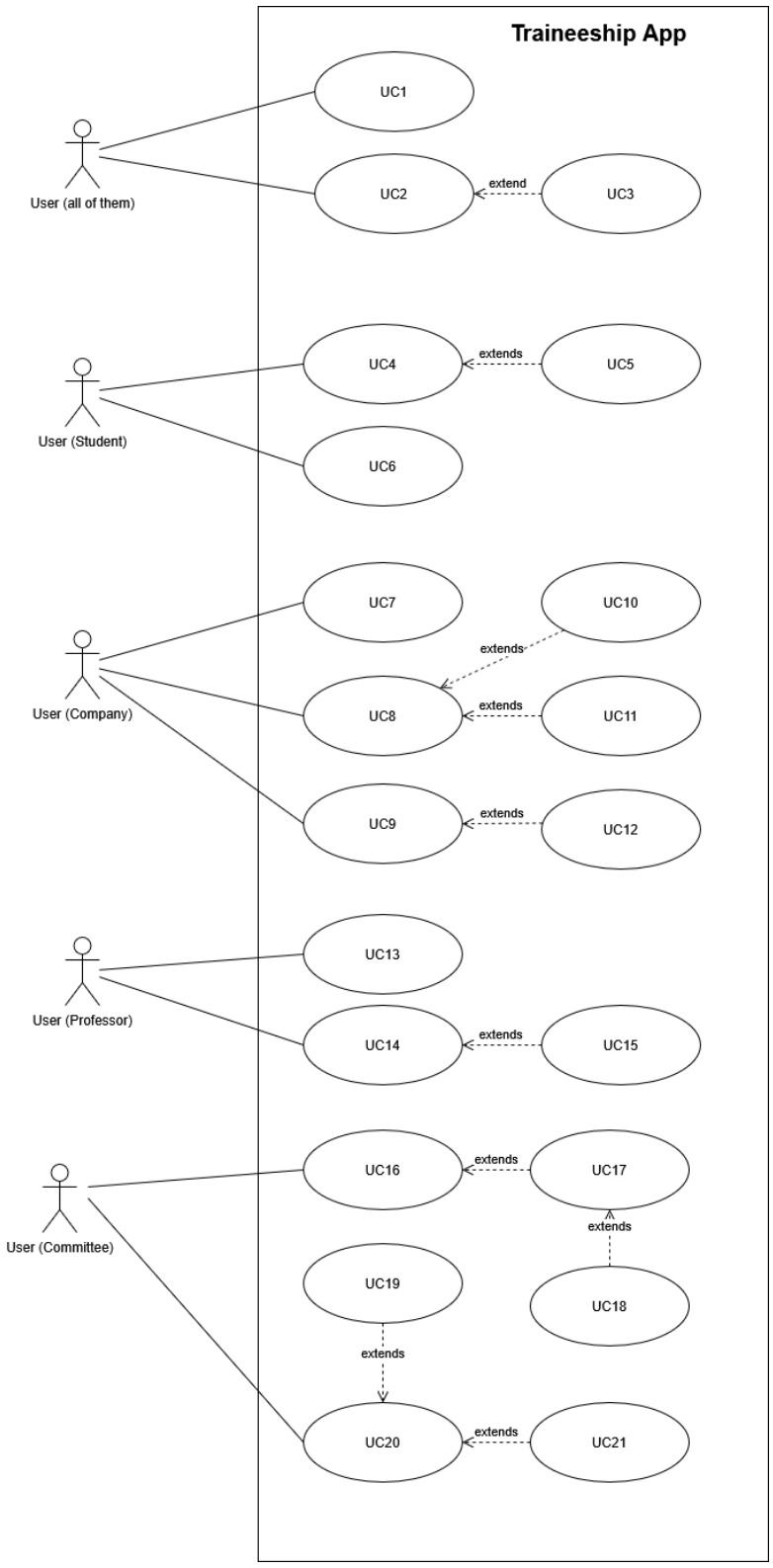
Product Owner	Omiros Chatziiordanis
Scrum Master	Strouggis Georgios
Development Team	Dimoudis Georgios, Strouggis Georgios, Chatziiordanis Omiros

2.2 Sprints

Sprint No	Begin Date	End Date	Number of weeks	User stories
00	**2025-03-10**	**2025-03-15**	**1**	**None (Decided on the different user stories, delegated them, made the 21 use cases needed. Decided on scrum team roles.)**
01	**2025-03-15**	**2025-03-25**	**1.5**	**User stories 1-7**

02	**2025-03-22**	**2025-03-31**	**1.5**	**User stories 8-15**
03	**2025-04-1**	**2025-04-25**	**3**	**User stories 15-21 and fixes for the rest of the user stories**
04	**2025-04-25**	**2025-05-26**	**4**	**General fixes of bugs and testing of running the application**

3 Use Cases


Traineeship App
UC1
UC2
UC3
extend
User (all of them)
UC4
UC5
extends
UC6
User (Student)
UC7
UC10
extends
UC8
extends
UC11
UC9
extends
UC12
User (Company)
UC13
UC14
extends
UC15
User (Professor)
UC16
extends
UC17
extends
UC19
UC18
extends
UC20
extends
UC21
User (Committee)

3.1 <Use Case 1>

Use case ID	UC1
Actors	User
Main flow of events	1. The use case starts when the user asks to register an account. 2. The user inputs the desired username and password. 3. The user selects the type of account they want to create (student, company, professor) 4. The user account is added to the database.
Alternative flow 1	The user may click on the login button at any point to return to leave the register screen and enter the login page.
Alternative flow 2	The user gets an error message if there exists another account with the same username.
Post conditions	The account is successfully created, the details are added to the database and the user enters the login page.

3.2 <Use Case 2>

Use case ID	UC2
Actors	User
Pre conditions	The user needs to have registered an account.
Main flow of events	1. The use case starts when the user asks to log in 2. The user inputs a username and password. 3. The user logs into their account
Alternative flow 1	The user may click on the register button at any point to leave the login screen and enter the register page.
Alternative flow 2	If the user inputs the wrong details, or the account doesn't exist, he is asked for the details again.
Post conditions	The user enters the user dashboard page.

3.3 <Use Case 3>

Use case ID	UC3
Actors	User
Pre conditions	The user is logged into his account.
Main flow of events	1. The use case starts when the user asks to log out of his account. 2. The user exits his dashboard page and enters the login screen.
Post conditions	The user enters the starting page, where he will need to re-enter his details in order to login.

3.4 <Use Case 4>

Use case ID	UC4
Actors	User (Student)
Pre conditions	The student is logged into his account.
Main flow of events	1. The use case starts when the user asks to view his profile. 2. The user updates their preferences (full name, university id number, interests, skills and preferred location for traineeship). 3. The user saves their preferences.
Alternative flow	The user may at any point return to the dashboard without saving their preferences.
Post conditions	The system updates the user's preferences, and the user returns to the dashboard.

3.5 <Use Case 5>

Use case ID	UC5
Actors	User (Student)

Pre conditions	The user is logged into his account and has made a profile.
Main flow of events	1. The use case starts when the student asks to view his profile. 2. The student selects that he is available for a traineeship position. 3. The user confirms his interest and returns to the dashboard.
Alternative flow 1	The student can return to the dashboard at any point without confirming his interest.
Post conditions	The system updates the student's preferences, and the student returns to the dashboard.

3.6 <Use Case 6>

Use case ID	UC6
Actors	User (Student)
Pre conditions	1. The user has logged into their account. 2. The user has been accepted into a traineeship position.
Main flow of events	1. The use case starts when the user selects to view his logbooks. 2. The user updates one of their logbook. 3. The user saves their work.
Alternative flow 1	The user can exit to the dashboard at any point without saving.
Post conditions	The system updates the student's logbook.

3.7 <Use Case 7>

Use case ID	UC7
Actors	User (Company)
Pre conditions	The user has logged into their account
Main flow of	1. The use case starts when the user asks to view their profile.

events	2. The user updates their preferences (company name and location). 3. The user saves their preferences.
Alternative flow 1	The user may at any point return to the dashboard without saving their preferences.
Post conditions	The system updates the user's preferences, and the user returns to the dashboard.

3.8 <Use Case 8>

Use case ID	UC8
Actors	User (Company)
Pre conditions	The user has logged into their account
Main flow of events	1. The use case starts when the user enters the main dashboard page. 2. The user clicks on the traineeship offers button. 3. The system takes the user to the traineeship offers page and loads all the offers the user has made before.

3.9 <Use Case 9>

Use case ID	UC9
Actors	User (Company)
Pre conditions	The user has logged into their account.
Main flow of events	1. The use case starts when the user enters the main dashboard page. 2. The user clicks on the assigned positions button. 3. The system takes the user to the traineeship positions page and loads all the offers the user has made, that another user (Student) has taken up.

3.10 <Use Case 10>

Use case ID	UC10
Actors	User (Company)
Pre conditions	The user has logged into their account.
Main flow of events	1. The use case starts when the user enters the traineeship offers page. 2. The user clicks on the add offer button. 3. The system loads the offer form page. 4. The user fills out the fields of the form (start date, end date, description, skills, interests). 5. The user clicks the save button. 6. The system saves the offer in the database and brings the user to the traineeship offers page.
Alternative flow 1	4. The user clicks the go back button. 5. The system brings the user to the traineeship offers page.
Post conditions	The traineeship offer and position is added to the database and is accessible for the other use cases.

3.11 <Use Case 11>

Use case ID	UC11
Actors	User (Company)
Pre conditions	1. The user has logged into their account. 2. The user has created at least one traineeship offer.
Main flow of events	1. The use case starts when the user enters the traineeship offers page. 2. The user clicks the delete offer button next to the corresponding traineeship offer. 3. The system deletes the offer from the database. 4. The system reloads the traineeship offers page.
Post conditions	The corresponding traineeship offer is removed from the database and it is not available anymore for other use cases.

3.12 <Use Case 12>

Use case ID	UC12
Actors	User (Company)
Pre conditions	1. The user has logged into their account. 2. The user has created at least one traineeship offer. 3. Another user (Student) has been assigned to one of them.
Main flow of events	1. The use case starts when the user enters the assigned positions page. 2. The user clicks the evaluate button on the corresponding traineeship position. 3. The system loads the evaluation page of the corresponding traineeship position. 4. The user fills out the fields (motivation, effectiveness, efficiency). 5. The user clicks the save button. 6. The system updates the traineeship position in the database with the evaluation data. 7. The system loads the traineeship positions page.
Alternative flow 1	5. The user clicks the back button. 6. The system loads the traineeship positions page.
Alternative flow 2	4. The system blocks any change to the fields if the position has been graded. 5. The user clicks the back button. 6. The system loads the traineeship positions page.
Post conditions	The traineeship position in the database has been updated values in the evaluation fields.

3.13 <Use Case 13>

Use case ID	UC13
Actors	User (Professor)
Pre conditions	The user is logged into his account.
Main flow of events	1. The use case starts when the user asks to view his profile. 2. The user updates their preferences (full name, interests).

	3. The user saves their preferences.
Alternative flow 1	The user may at any point return to the dashboard without saving their preferences.
Post conditions	The system updates the user's preferences, and the user returns to the dashboard.

3.14 <Use Case 14>

Use case ID	UC14
Actors	User (Professor)
Pre conditions	The user is logged into his account.
Main flow of events	1. The use case starts when the user enters the main dashboard page. 2. The user clicks the supervised traineeships button. 3. The system loads the traineeship positions page and all the traineeship positions that the user is assigned to.
Post conditions	The system remains the same.

3.15 <Use Case 15>

Use case ID	UC15
Actors	User (Professor)
Pre conditions	1. There must be at least one Student, one Professor, one Company and one TraineeshipPosition. 2. The Company must offer the TraineeshipPosition. 3. The Professor must supervise the TraineeshipPosition. 4. The Student must have taken up the TraineeshipPosition.
Main flow of events	1. The use case starts when the user (a Professor) chooses to fill in an evaluation. 2. The user evaluates the student motivation, effectiveness and efficiency

	as well as the hosting company in terms of the facilities provided and the guidance of the trainee student on a scale from 1 to 5.
Alternative flow 1	The user chooses to leave out certain fields without evaluating them.
Post conditions	An Evaluation object about the Student and Company is saved in the database now available.

3.16 <Use Case 16>

Use case ID	UC16
Actors	User (Traineeship Committee)
Pre conditions	1. There must be at least one Student 2. The Student must have applied for a TraineeshipPosition and be pending acceptance.
Main flow of events	1. The use case starts when the user selects to see the list of Students who applied for a TraineeshipPosition. 2. The system presents the list of Students to him.
Alternative flow 1	If no Students have applied then the list is empty.
Post conditions	The user can now select a Student.

3.17 <Use Case 17>

Use case ID	UC17
Actors	User (Traineeship Committee)
Pre conditions	1. There must be at least one Student, one Company and one TraineeshipPosition. 2. The Company must offer the TraineeshipPosition. 3. The Student must have applied for the TraineeshipPosition and be pending acceptance. 4. There must be available positions that have yet to be taken.

Main flow of events	1. The use case starts when the user selects one of the available Students. 2. The system presents the user with a list of available positions that match the interests, location or both offered by the Student.
Alternative flow 1	If no positions are available or none match the Student's interest or location or both, or the Student doesn't have the required skills for any of them, then the list is empty.
Post conditions	The user can now select a position for the Student.

3.18 <Use Case 18>

Use case ID	UC18
Actors	User (Traineeship Committee)
Pre conditions	1. There must be at least one Student, one Company and one TraineeshipPosition. 2. The Company must offer the TraineeshipPosition. 3. The Student must have applied for the TraineeshipPosition and be pending acceptance. 4. There must be available positions that have yet to be taken. 5. The user was presented with a list of available positions that match the selected Student's interest, location and have all their skills filled.
Main flow of events	1. The use case starts when the user selects an available position. 2. The user assigns that position to the selected Student.
Post conditions	The Student and TraineeshipPosition now are assigned to one another.

3.19 <Use Case 19>

Use case ID	UC19
Actors	User (Traineeship Committee)
Pre conditions	1. There must be at least one Professor, one Company and one TraineeshipPosition.

	2. The Company must offer the TraineeshipPosition. 3. The TraineeshipPosition must not have a supervising Professor.
Main flow of events	1. The use case starts when the system looks through the list of available Professors. 2. The system finds Professors whose interests match the available position and the limit of other supervising roles they're attached to. 3. The user assigns the Professor to supervise that TraineeshipPosition.
Alternative flow 1	If the list of available Professors is empty, the position is left unsupervised.
Post conditions	The TraineeshipPosition now has a supervising Professor and the position is added in the counter of supervised positions for that Professor.

3.20 <Use Case 20>

Use case ID	UC20
Actors	User (Traineeship Committee)
Pre conditions	1. There must be at least one Company and one TraineeshipPosition. 2. The Company must offer the TraineeshipPosition.
Main flow of events	1. The use case starts when the user requests to receive a list of traineeships that don't have a mark yet. 2. The system returns a list of all traineeships in progress.
Alternative flow 1	If there are no traineeships in progress then the user receives an empty list.

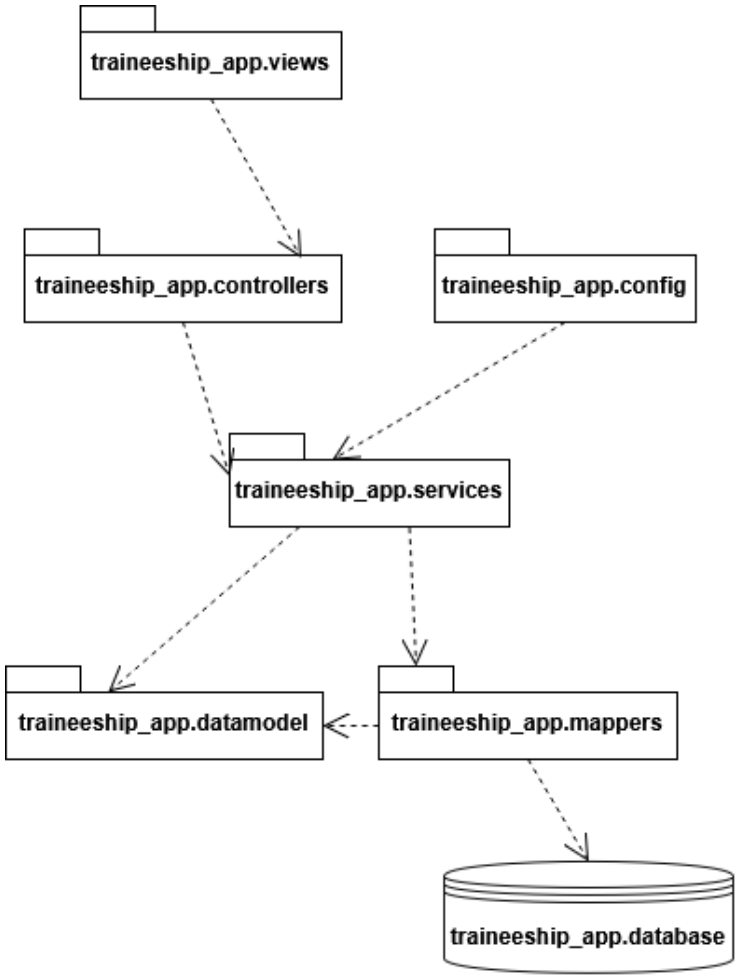
3.21 <Use Case 21>

Use case ID	UC21
Actors	User (Traineeship Committee)
Pre conditions	1. There must be at least one Student, one Professors, one Company, one TraineeshipPosition and the two Evaluations. 2. The Company must offer the TraineeshipPosition.

	3. One Professor must supervise the TraineeshipPosition. 4. The Student must have been assigned to the TraineeshipPosition.
Main flow of events	1. The use case starts when the user selects a traineeship that is in progress. 2. The system presents the user with the Evaluations made for the traineeship and suggests a mark based on their grades. 3. The user gives it a pass or fail mark.
Alternative flow 1	If one of the Evaluations is missing, the user is warned.
Alternative flow 2	The user can return to the dashboard without giving a mark.
Post conditions	The traineeship has been marked as either passed or failed.

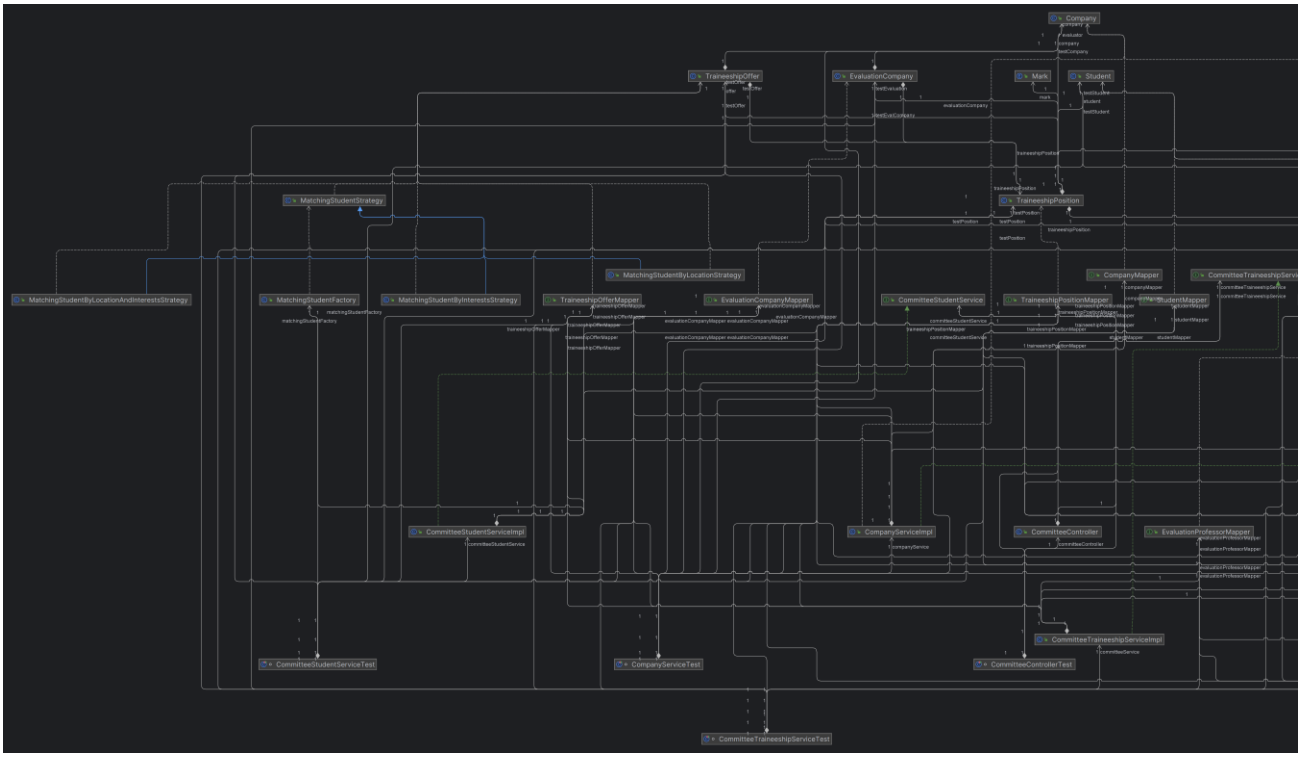
4 Design

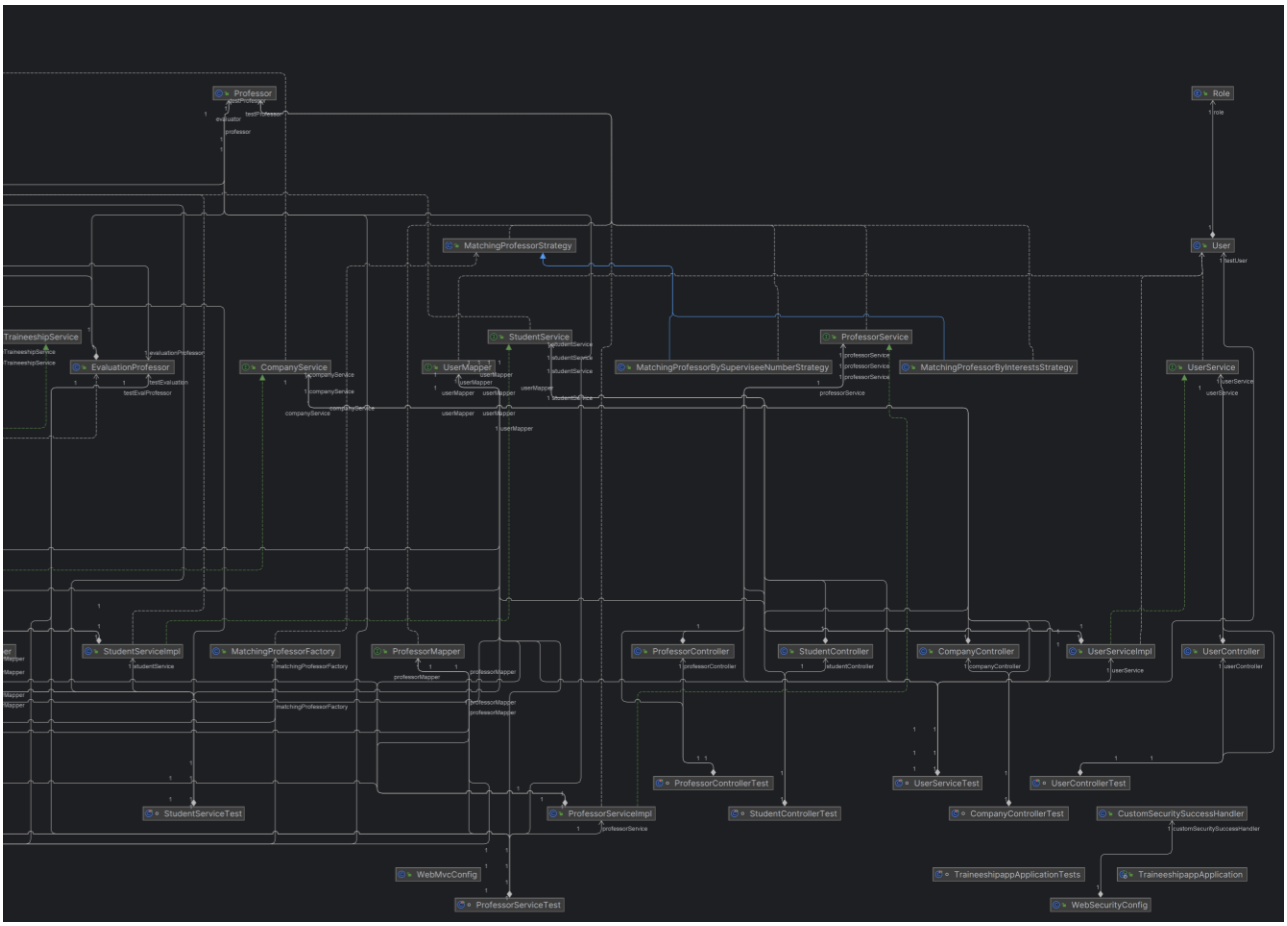
4.1 Architecture



The above diagram specifies the architectural structure of our release, with the distinct layers that make up the application. Specifically, there exists the presentation (views), business logic (services), data (datamodel, database) and intermediate helping layers (controllers, mappers, config) that make possible their connections. The controllers are responsible for the handling of user requests in the views and calling the right services, the config stores important settings for the application and the mappers are the ones that access the database and bring data from it, in the form of the datamodel, for the services to use.

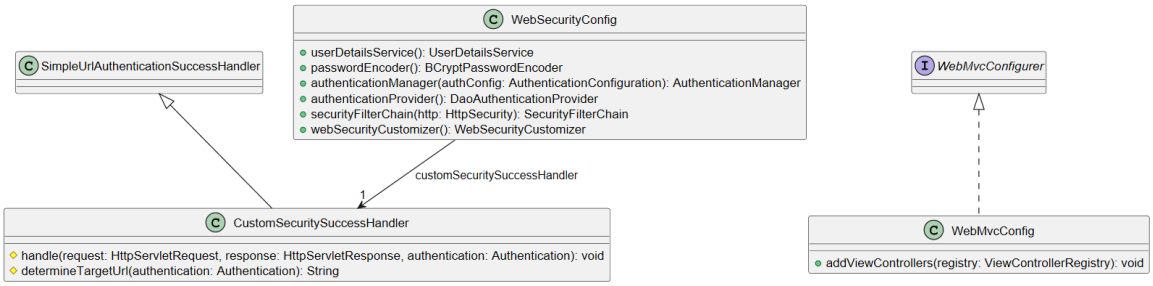
4.2 Design


Company
TraineeshipOffer
EvaluationCompany
Mark
Student
MatchingStudentStrategy
TraineeshipPosition
MatchingStudentByLocationStrategy
CompanyMapper
MatchingStudentByLocationAndInterestsStrategy
MatchingStudentFactory
MatchingStudentByInterestsStrategy
TraineeshipOfferMapper
EvaluationCompanyMapper
CommitteeStudentService
TraineeshipPositionMapper
StudentMapper
CommitteeStudentServiceImpl
CompanyServiceImpl
CommitteeController
EvaluationProfessorMapper
CommitteeTraineeshipServiceImpl
CommitteeStudentServiceTest
CompanyServiceTest
CommitteeControllerTest
CommitteeTraineeshipServiceTest


Professor
Role
MatchingProfessorStrategy
User
EvaluationProfessor
CompanyService
StudentService
UserMapper
MatchingProfessorBySuperviseeNumberStrategy
ProfessorService
MatchingProfessorByInterestsStrategy
UserService
StudentServiceImpl
MatchingProfessorFactory
ProfessorMapper
ProfessorController
StudentController
CompanyController
UserServiceImpl
UserController
ProfessorControllerTest
UserServiceTest
UserControllerTest
StudentServiceTest
ProfessorServiceImpl
StudentControllerTest
CompanyControllerTest
CustomSecuritySuccessHandler
WebMvcConfig
TraineeshipappApplicationTests
TraineeshipappApplication
ProfessorServiceTest
WebSecurityConfig

Package myy803.traineeshipapp:

Class Name: TraineeshipappApplication	
Responsibilities: Starts the application	**Collaborations:** org.springframework.boot.SpringApplication org.springframework.boot.autoconfigure.SpringBootApplication



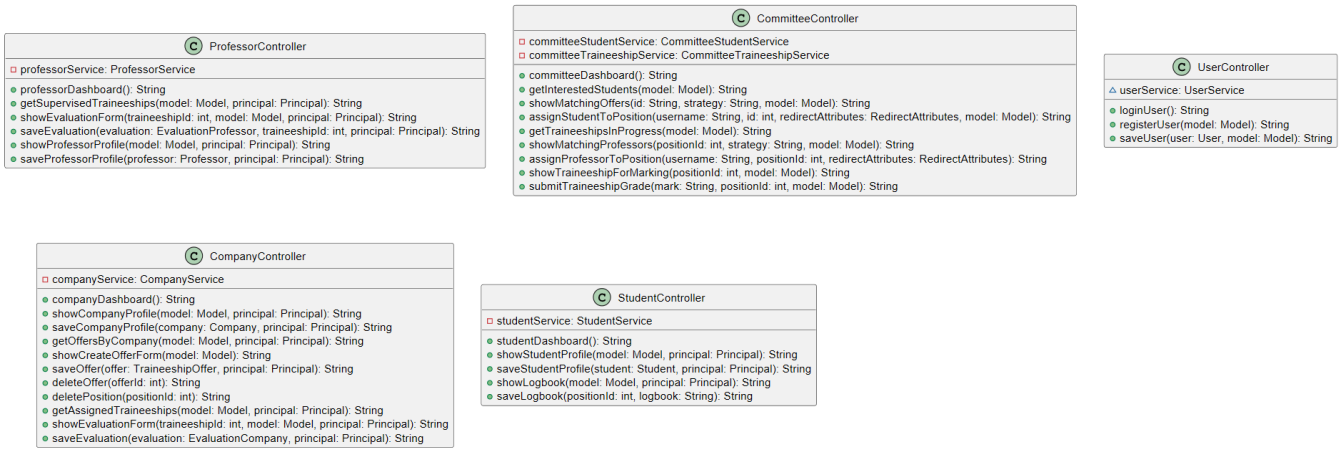
Package myy803.traineeshipapp.config

Class Name: CustomSecuritySuccessHandler	
Responsibilities: Handles successful redirect requests, and redirects the user to the correct dashboard, according to his role.	**Collaborations:** jakarta.servlet.http.HttpServletRequest jakarta.servlet.http.HttpServletResponse org.springframework.context.annotation.Configuration org.springframework.security.core.Authentication org.springframework.security.core.GrantedAuthority org.springframework.security.web.DefaultRedirectStrategy org.springframework.security.web.RedirectStrategy org.springframework.security.web.authentication.SimpleUrlAuthenticationSuccessHandler

Class Name: WebMvcConfig	
Responsibilities:	**Collaborations:**

Defines the root path	org.springframework.context.annotation.Configuration org.springframework.web.servlet.config.annotation.ViewControllerRegistry org.springframework.web.servlet.config.annotation.WebMvcConfigurer

Class Name: WebSecurityConfig	
Responsibilities: Encrypts passwords Checks for user authority Authorizes users' login	**Collaborations:** **Collaborations:** org.springframework.beans.factory.annotation.Autowired org.springframework.context.annotation.Bean org.springframework.context.annotation.Configuration org.springframework.security.authentication.AuthenticationManager org.springframework.security.authentication.dao.DaoAuthenticationProvider org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration org.springframework.security.config.annotation.web.builders.HttpSecurity org.springframework.security.config.annotation.web.configuration.EnableWebSecurity org.springframework.security.config.annotation.web.configuration.WebSecurityCustomizer org.springframework.security.core.userdetails.UserDetailsService org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder org.springframework.security.web.SecurityFilterChain org.springframework.security.web.util.matcher.AntPathRequestMatcher



Package myy803.traineeshipapp.controllers

Class Name: CommitteeController	
Responsibilities: Handles HTTP requests Delegates business logic to CommitteeStudentService and CommitteeTraineeshipService.	**Collaborations:** org.springframework.beans.factory.annotation.Autowired org.springframework.stereotype.Controller org.springframework.ui.Model org.springframework.web.bind.annotation.GetMapping org.springframework.web.bind.annotation.PostMapping org.springframework.web.bind.annotation.RequestMapping org.springframework.web.bind.annotation.RequestParam org.springframework.web.servlet.mvc.support.RedirectAttributes myy803.traineeshipapp.services.CommitteeStudentService myy803.traineeshipapp.services.CommitteeTraineeshipService

Class Name: CompanyController	
Responsibilities: Handles HTTP requests Delegates business logic to CompanyService	**Collaborations:** org.springframework.beans.factory.annotation.Autowired org.springframework.stereotype.Controller org.springframework.ui.Model org.springframework.web.bind.annotation.*

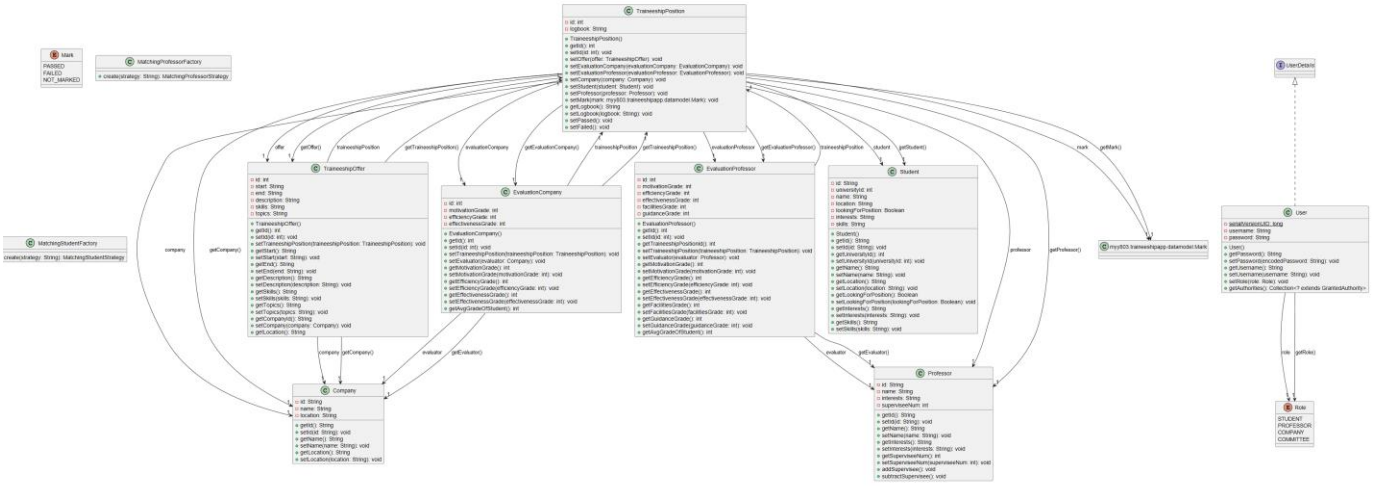
	myy803.traineeshipapp.datamodel.Company myy803.traineeshipapp.datamodel.EvaluationCompany myy803.traineeshipapp.datamodel.TraineeshipOffer myy803.traineeshipapp.services.CompanyService

Class Name: ProfessorController	
Responsibilities: Handles HTTP requests Delegates business logic to ProfessorService	**Collaborations:** org.springframework.beans.factory.annotation.Autowired org.springframework.stereotype.Controller org.springframework.ui.Model org.springframework.web.bind.annotation.* myy803.traineeshipapp.datamodel.EvaluationProfessor myy803.traineeshipapp.datamodel.Professor myy803.traineeshipapp.services.ProfessorService

Class Name: StudentController	
Responsibilities: Handles HTTP requests Delegates business logic to StudentService	**Collaborations:** org.springframework.beans.factory.annotation.Autowired org.springframework.stereotype.Controller org.springframework.ui.Model org.springframework.web.bind.annotation.* myy803.traineeshipapp.datamodel.Student myy803.traineeshipapp.services.StudentService

Class Name: UserController	
Responsibilities: Handles HTTP requests Delegates business logic to UserService	**Collaborations:** org.springframework.beans.factory.annotation.Autowired org.springframework.stereotype.Controller org.springframework.ui.Model org.springframework.web.bind.annotation.ModelAttribute

	org.springframework.web.bind.annotation.RequestMapping myy803.traineeshipapp.datamodel.User myy803.traineeshipapp.services.UserService



Package myy803.traineeshipapp.datamodel

Class Name: Company	
Responsibilities: Represents table “company”	**Collaborations:** jakarta.persistence.* org.springframework.stereotype.Component

Class Name: EvaluationCompany	
Responsibilities: Represents table “evaluation_company”	**Collaborations:** jakarta.persistence.*

Class Name: EvaluationProfessor	
Responsibilities: Represents table “evaluation_professor”	**Collaborations:** jakarta.persistence.*

Class Name: Mark

Responsibilities:	**Collaborations:**
Defines the grading states	-

Class Name: Professor	
Responsibilities:	**Collaborations:**
Represents table “professor”	**Jakarta.persistence.*** **org.springframework.stereotype.Component**

Class Name: Role	
Responsibilities:	**Collaborations:**
Defines the available program roles	-

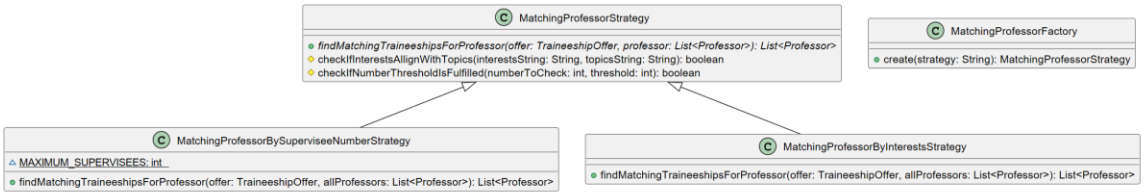
Class Name: Student	
Responsibilities:	**Collaborations:**
Represents table “student”	jakarta.persistence.* org.springframework.stereotype.Component

Class Name: TraineeshipOffer	
Responsibilities:	**Collaborations:**
Represents table “traineeship_offer”	jakarta.persistence.*

Class Name: TraineeshipPosition	
Responsibilities:	**Collaborations:**
Represents table “traineeship_position”	jakarta.persistence.*

Class Name: User	
Responsibilities:	**Collaborations:**
Represents table “users”	java.util.Collection java.util.Collections

	jakarta.persistence.* org.springframework.security.core.GrantedAuthority; org.springframework.security.core.authority.SimpleGrantedAuthority org.springframework.security.core.userdetails.UserDetails



Package myy803.traineeshipapp.datamodel.matchingProfessorStrategies

Class Name: MatchingProfessorByInterestsStrategy	
Responsibilities: Provides list of professors whose interests align with the topics covered by a traineeship.	**Collaborations:** java.util.ArrayList java.util.List org.springframework.stereotype.Component myy803.traineeshipapp.datamodel.Professor myy803.traineeshipapp.datamodel.TraineeshipOffer

Class Name: MatchingProfessorBySuperviseeNumberStrategy	
Responsibilities: Provides list of professors who supervise less than nine traineeships.	**Collaborations:** java.util.ArrayList java.util.Comparator java.util.List org.springframework.stereotype.Component myy803.traineeshipapp.datamodel.Professor myy803.traineeshipapp.datamodel.TraineeshipOffer

Class Name: MatchingProfessorFactory	
Responsibilities: Selects which professor finding strategy will be employed.	**Collaborations:** org.springframework.stereotype.Component

Class Name: MatchingProfessorStrategy	
Responsibilities: Abstract class that the two strategies implement.	**Collaborations:** java.util.Arrays java.util.HashSet java.util.List java.util.Set myy803.traineeshipapp.datamodel.Professor myy803.traineeshipapp.datamodel.TraineeshipOffer

Package myy803.traineeshipapp.datamodel.matchingStudentStrategies

Class Name: MatchingStudentByInterestsStrategy	
Responsibilities: Provides list of traineeship offers whose topics align with the student's interests as long as the student has the necessary skills.	**Collaborations:** java.util.ArrayList java.util.List org.springframework.stereotype.Component

	myy803.traineeshipapp.datamodel.Student myy803.traineeshipapp.datamodel.TraineeshipOffer

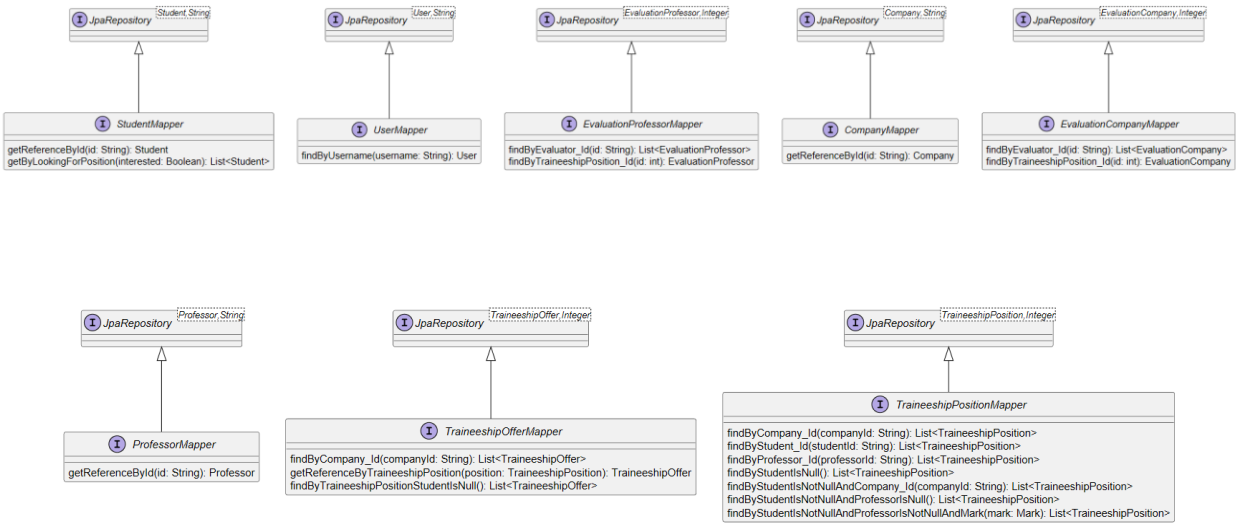
Class Name: MatchingStudentByLocationAndInterestsStrategy	
Responsibilities: Provides list of traineeship offers whose topics and location align with the student's interests and preferred location as long as the student has the necessary skills.	**Collaborations:** java.util.ArrayList java.util.List org.springframework.stereotype.Component myy803.traineeshipapp.datamodel.Student myy803.traineeshipapp.datamodel.TraineeshipOffer

Class Name: MatchingStudentByLocationStrategy	
Responsibilities: Provides list of traineeship offers whose locations with the student's preferred location as long as the student has the necessary skills.	**Collaborations:** java.util.ArrayList java.util.List org.springframework.stereotype.Component myy803.traineeshipapp.datamodel.Student myy803.traineeshipapp.datamodel.TraineeshipOffer

Class Name: MatchingStudentFactory	
Responsibilities: Selects which traneeship offer finding strategy will be employed.	**Collaborations:** org.springframework.stereotype.Component

Class Name: MatchingStudentStrategy	
Responsibilities: Abstract class that the three strategies implement.	**Collaborations:** java.util.Arrays java.util.HashSet java.util.List

	java.util.Set myy803.traineeshipapp.datamodel.Student myy803.traineeshipapp.datamodel.TraineeshipOffer



Package myy803.traineeshipapp.mappers

Class Name: CompanyMapper	
Responsibilities: Provides method for searching “company” table.	**Collaborations:** org.springframework.data.jpa.repository.JpaRepository myy803.traineeshipapp.datamodel.Company

Class Name: EvaluationCompanyMapper	
Responsibilities: Provides methods for searching “evaluation_ company” table.	**Collaborations:** java.util.List org.springframework.data.jpa.repository.JpaRepository myy803.traineeshipapp.datamodel.EvaluationCompany

Class Name: EvaluationProfessorMapper

Responsibilities:	**Collaborations:**
Provides methods for searching “evaluation_professor” table.	java.util.List org.springframework.data.jpa.repository.JpaRepository myy803.traineeshipapp.datamodel.EvaluationProfessor

Class Name: ProfessorMapper	
Responsibilities:	**Collaborations:**
Provides method for searching “professor” table.	org.springframework.data.jpa.repository.JpaRepository myy803.traineeshipapp.datamodel.Professor

Class Name: StudentMapper	
Responsibilities:	**Collaborations:**
Provides method for searching “student” table.	java.util.List; org.springframework.data.jpa.repository.JpaRepository myy803.traineeshipapp.datamodel.Student

Class Name: TraineeshipOfferMapper	
Responsibilities:	**Collaborations:**
Provides methods for searching “traineeship_offer” table.	java.util.List org.springframework.data.jpa.repository.JpaRepository myy803.traineeshipapp.datamodel.TraineeshipOffer myy803.traineeshipapp.datamodel.TraineeshipPosition

Class Name: TraineeshipPositionMapper	
Responsibilities:	**Collaborations:**
Provides methods for searching “traineeship_position” table.	java.util.List org.springframework.data.jpa.repository.JpaRepository myy803.traineeshipapp.datamodel.Mark

	myy803.traineeshipapp.datamodel.TraineeshipPosition

Class Name: UserMapper	
Responsibilities: Provides method for searching “users” table.	**Collaborations:** org.springframework.data.jpa.repository.JpaRepository myy803.traineeshipapp.datamodel.User

Package myy803.traineeshipapp.services

Class Name: CommitteeStudentService	
Responsibilities: Interface for use cases 16,17,18.	**Collaborations:** org.springframework.ui.Model

Class Name: CommitteeStudentServiceImpl

Responsibilities:	Collaborations:
Implementation of use cases 16,17,18.	org.springframework.beans.factory.annotation.Autowired org.springframework.stereotype.Service org.springframework.ui.Model myy803.traineeshipapp.mappers.EvaluationCompanyMapper myy803.traineeshipapp.mappers.StudentMapper myy803.traineeshipapp.mappers.TraineeshipOfferMapper myy803.traineeshipapp.mappers.TraineeshipPositionMapper myy803.traineeshipapp.datamodel.TraineeshipPosition myy803.traineeshipapp.datamodel.matchingStudentStrategies.MatchingStudentFactory myy803.traineeshipapp.datamodel.matchingStudentStrategies.MatchingStudentStrategy Myy803.traineeshipapp.datamodel.EvaluationCompany myy803.traineeshipapp.datamodel.Student myy803.traineeshipapp.datamodel.TraineeshipOffer

Class Name: CommitteeTraineeshipService	
Responsibilities:	**Collaborations:**
Interface for use cases 19,20,21.	org.springframework.ui.Model

Class Name: CommitteeTraineeshipServiceImpl	
Responsibilities:	**Collaborations:**
Implementation for use cases 19,20,21.	org.springframework.beans.factory.annotation.Autowired org.springframework.stereotype.Service; import org.springframework.ui.Model myy803.traineeshipapp.datamodel.EvaluationCompany myy803.traineeshipapp.datamodel.EvaluationProfessor myy803.traineeshipapp.datamodel.Professor myy803.traineeshipapp.datamodel.TraineeshipOffer myy803.traineeshipapp.datamodel.TraineeshipPosition myy803.traineeshipapp.datamodel.matchingProfessorStrategies.MatchingProfessorFactory myy803.traineeshipapp.datamodel.matchingProfessorStrategies.MatchingProfessorStrategy

	myy803.traineeshipapp.mappers.EvaluationCompanyMapper myy803.traineeshipapp.mappers.EvaluationProfessorMapper myy803.traineeshipapp.mappers.ProfessorMapper myy803.traineeshipapp.mappers.TraineeshipOfferMapper myy803.traineeshipapp.mappers.TraineeshipPositionMapper

Class Name: CompanyService	
Responsibilities: Interface for use cases 7,8,9,10,11,12.	**Collaborations:** myy803.traineeshipapp.datamodel.Company myy803.traineeshipapp.datamodel.EvaluationCompany myy803.traineeshipapp.datamodel.TraineeshipOffer java.security.Principal org.springframework.ui.Model

Class Name: CompanyServiceImpl	
Responsibilities: Implementation for use cases 7,8,9,10,11,12.	**Collaborations:** org.springframework.beans.factory.annotation.Autowired org.springframework.stereotype.Service import org.springframework.ui.Model myy803.traineeshipapp.mappers.CompanyMapper myy803.traineeshipapp.mappers.EvaluationCompanyMapper myy803.traineeshipapp.mappers.TraineeshipOfferMapper myy803.traineeshipapp.mappers.TraineeshipPositionMapper myy803.traineeshipapp.mappers.UserMapper myy803.traineeshipapp.datamodel.TraineeshipOffer myy803.traineeshipapp.datamodel.TraineeshipPosition myy803.traineeshipapp.datamodel.User myy803.traineeshipapp.datamodel.Company myy803.traineeshipapp.datamodel.EvaluationCompany java.security.Principal

	java.util.List java.util.stream.Collectors java.util.stream.IntStream

Class Name: ProfessorService	
Responsibilities: Interface for use cases 13,14,15.	**Collaborations:** myy803.traineeshipapp.datamodel.EvaluationProfessor myy803.traineeshipapp.datamodel.Professor java.security.Principal org.springframework.ui.Model

Class Name: ProfessorServiceImpl	
Responsibilities: Implementation for use cases 13,14,15.	**Collaborations:** org.springframework.beans.factory.annotation.Autowired org.springframework.stereotype.Service org.springframework.ui.Model myy803.traineeshipapp.datamodel.EvaluationProfessor myy803.traineeshipapp.datamodel.Professor myy803.traineeshipapp.datamodel.TraineeshipPosition myy803.traineeshipapp.datamodel.User myy803.traineeshipapp.mappers.EvaluationProfessorMapper myy803.traineeshipapp.mappers.ProfessorMapper myy803.traineeshipapp.mappers.TraineeshipPositionMapper myy803.traineeshipapp.mappers.UserMapper java.security.Principal java.util.List java.util.stream.Collectors java.util.stream.IntStream

Class Name: StudentService

Responsibilities:	Collaborations:
Interface for use cases 4,5,6.	org.springframework.ui.Model myy803.traineeshipapp.datamodel.Student

Class Name: StudentServiceImpl	
Responsibilities:	**Collaborations:**
Implementation for use cases 4,5,6.	org.springframework.beans.factory.annotation.Autowired org.springframework.stereotype.Service org.springframework.ui.Model myy803.traineeshipapp.datamodel.Student myy803.traineeshipapp.datamodel.TraineeshipPosition myy803.traineeshipapp.datamodel.User myy803.traineeshipapp.mappers.StudentMapper myy803.traineeshipapp.mappers.TraineeshipPositionMapper myy803.traineeshipapp.mappers.UserMapper

Class Name: UserService	
Responsibilities:	**Collaborations:**
Interface for use cases 1,2,3.	org.springframework.security.core.userdetails.UserDetailsService org.springframework.ui.Model myy803.traineeshipapp.datamodel.User

Class Name: UserServiceImpl	
Responsibilities:	**Collaborations:**
Implementation for use cases 1,2,3.	org.springframework.beans.factory.annotation.Autowired org.springframework.dao.EmptyResultDataAccessException org.springframework.security.core.userdetails.UserDetails org.springframework.security.core.userdetails.UsernameNotFoundException org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder org.springframework.stereotype.Service

	org.springframework.ui.Modeli.Model myy803.traineeshipapp.datamodel.Company myy803.traineeshipapp.datamodel.Professor myy803.traineeshipapp.datamodel.Student myy803.traineeshipapp.datamodel.User myy803.traineeshipapp.mappers.UserMapper