Bilkent University

Department of Computer Engineering

Object Oriented Software Engineering Course Project

BilHub: A Classroom Helper

Design Report

Barış Ogün Yörük, Halil Özgür Demir, Mustafa Çağrı Durgut, Yusuf Miraç Uyar, Aybala Karakaya, Oğuzhan Özçelik

Instructor: Eray Tüzün

Teaching Assistant(s): Elgun Jabrayilzade, Erdem Tuna, Barış Ardıç

Progress Report

April 11, 2021

Contents

1. Introduction

1.1 Purpose of the System

BilHub is a student management system designed for Bilkent students and instructors that enables students to form groups, submit assignments, comment on other projects, and peer grade their group members. Also, this system makes it easier for instructors and TA's to create courses and give assignments.

1.2 Design Goals

1.2.1 Usability

This system must be useful, because, there are already some systems that have some functionalities of this system (eg. Moodle, Slack) and users of these systems may not want to use a program which is harder to use than the programs they are already using or the programs alternative to it. In order to make this system usable, it will offer mostly used preferences for creating courses, assignments, comment mode (grading and/or commenting other projects), peer grading by default. Also it won't show unnecessary things while creating those things (not shows section number selection if the section is chosen to be sectionless).

1.2.2 Reusability

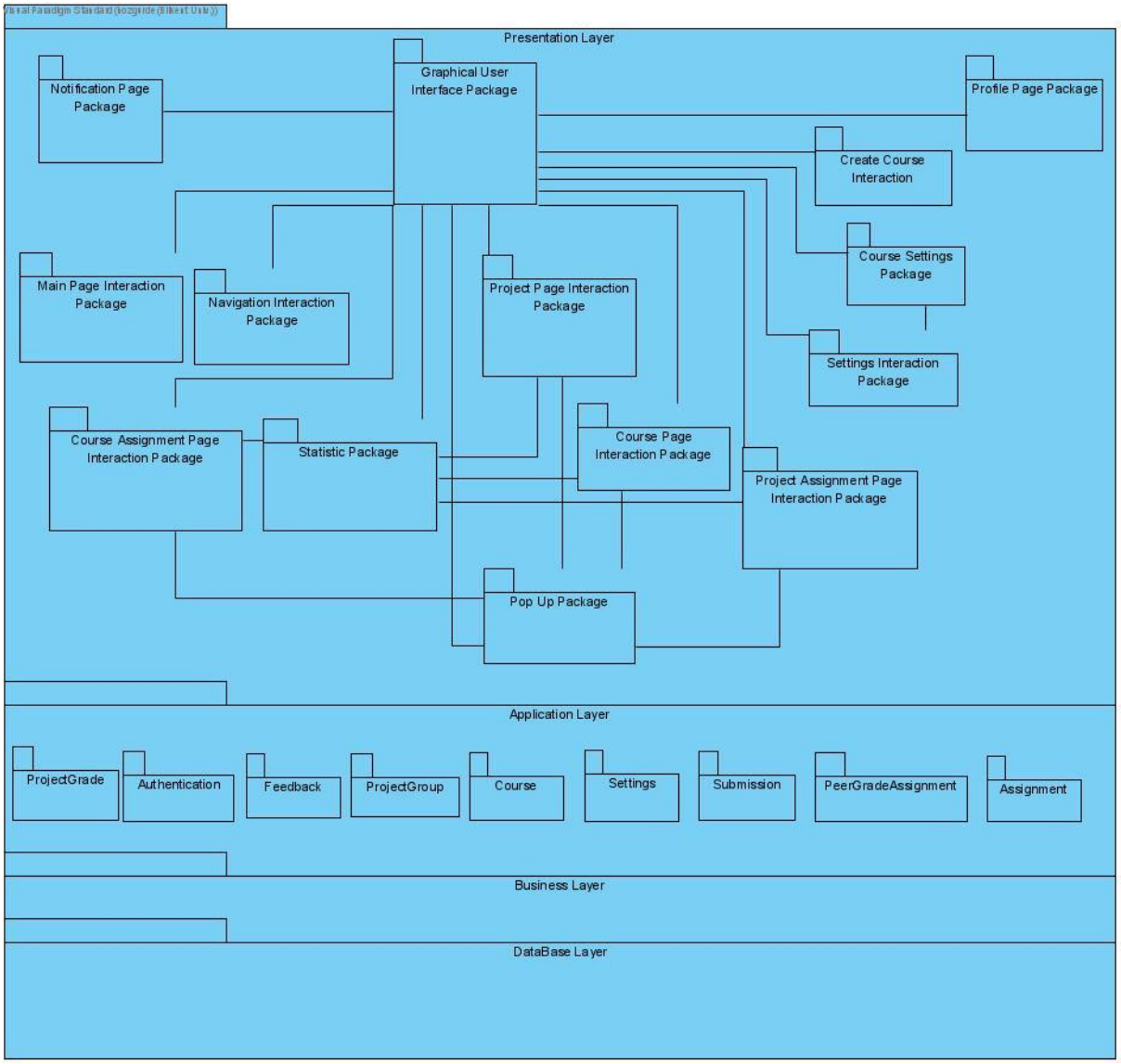
This system must be reusable. This is because we have limited time and in order not to decrease some other thing towards cost. In order to make the system reusable, we are going to use pre designed user-interfaces, and libraries for places which are not object-oriented.

1.2.3 Maintainability

This system must be maintainable because it will be used by all the CS students in Bilkent (It can also be applied to other departments). In order to do this, we will design this subsystem with low coupling, so a change in one system does not have much effect on the other subsystem. Also we are going to use widely used technologies which are CSS, React, and .NET so changing the implementation according to needs won't require learning new libraries.

2. High Level Design

2.1 Subsystem Decomposition



For subsystem decomposition we used multitier architecture. We've chosen this design because it was best fit for our needs.

We used four layers four our subsystem decomposition. Three of them are for backend implementation and one of them is for front-end implementation.

In the front-end implementation, we will use a presentation layer to visualize our application to the end-user. In this layer all the visualization and validation regarding the front end will take place. With the data we get from the application layer, we will visualize those data in here. With the data we get from the user, we will

send that data to the application layer to manipulate the database with proper logic that we don't care about in this layer.

In the back-end implementation we will use three layers.

Application Layer will be the layer between the presentational layer and the business layer. We will implement a REST API in this layer to transfer data between the presentational layer and the business layer. This layer will provide several end-points to interact with the presentational layer. It will send the data that is requested by the front-end via getting it from the business layer and it will pass the data that is sent by the front-end to the business layer to manipulate the database with proper logic that we don't care about in this layer. This layer will be responsible for communication between the backend and frontend as well as whether the requests are valid.

Business layer will be the heart of our system. It will take the actions that are coming from the application layer and manipulate the database accordingly with some validations before. Here we will provide functions that manipulate the database if they have valid parameters. This layer will secure the manipulation of the database.

Database layer will hold all data regarding users, assignments, courses and projects. Database will be only manipulated via business layer.

2.2 Hardware / Software Mapping

BilHub can work on any browser which supports react.js. In order to use this system the device which tries to access BilHub must have internet connection. Even though this system works on every react.js supported browser, its user interface will be designed according to PC view, so in order to use this system smoother, it will be better to use it in a PC environment.

2.3 Persistent Data Management

BilHub is going to use an online database in order to store and alter the data, and communicate with them. We are going to use SQL to implement the database. Communication with the database is handled inside Business Layer which is connected to both Application Layer and Database Layer. All the necessary data will be kept in SQL so we won't use any other storage system for data management.

2.4 Access Control and Security

In BilHub, there are 3 types of users and admins. Users cannot manipulate databases directly, they can only do this by using the features of the system. Each user has a different level of access and manipulation to the database. For example, while instructors can create courses, students do not have authority to do this, and while TA's can add students to courses, students cannot add other students to courses. Admin is not a direct user of the system. It has the most access over the database because it can access the database directly in order to add instructors to the system.

In Bilhub, there is a security system in order to prevent students from universities other than Bilkent from joining the system. While one student is trying to enter the system, the email and password of that student is asked and after they are entered, the system checks whether or not this email belongs to a Bilkent student. If it is, the student successfully joins the system, if not, the student is warned with a message that says students not from the Bilkent is not acceptable to the system.

2.5 Boundary Conditions

2.5.1 Initialization

BilHub will be a web application. Therefore, it will not require any installation and will be initialized when the user goes into the main page of BilHub on their favorite browser and logs in using their Bilkent email address and BilHub password. Users will also have the choice to use 2-step verification. In that case, they will also need to enter the code that is sent to their phones in order to make the initialization completed.

2.5.2 Termination

BilHub will be terminated when the user signs out from the menu. Additionally, if the user closes his/her browser or the tab in which the BilHub is open, is inactive or there is a problem with Internet connection, his/her session will be automatically terminated in 15 minutes due to security and privacy reasons.

When the program is terminated, the unsaved data gets automatically saved.

2.5.3 Failure

If BilHub crashes because of an issue related to performance or design, it will show a pop-up saying "Ups, we are having problems. Please

log-in again.”and it will be terminated. When the user closes that pop-up, s/he will be taken to the log-in page. In the case of that kind of an error, unsaved data will be lost and the system will continue with the latest previously saved data.

If there is a problem because of a bug such as a student is present in two groups or one group has more than one assignment submitted by different students, the instructor can step in and choose a group for the student or delete one of the submissions, respectively.

If there is a problem while trying to save the data and data cannot be saved, the user will be notified of the situation and s/he can take necessary precautions and try again later.

If an already existing user in the system tries to sign in again, the system shows an error that “The given email already exists”. It is also the same in course registration. For example if the instructor or TA try to add an existing student to the course, the system shows error for that student that “The student already exists.”. However, as sometimes TA or instructor upload files with the mails of students, if one student already exists, it does not completely crash, it only shows errors that include already existing mails of students and add other students to the course.

If the instructor tries to add a mail which does not exist in the system, s/he gets an error about it and cannot add that into the course. If s/he faces this issue while trying to add students via file, the system gives error for that student(or students) and adds others which have not any problems.

If a student's connection to the database is lost, for example if s/he wants to submit an assignment however cannot submit, there will be a pop up which asks for sending both to the admin and instructor of that course a message about it. By doing this, we are preventing students from losing unnecessary points because the due date of assignment is passed.

3. Low Level Design

3.1 Design Trade-offs

3.1.1 Functionality vs Usability

We prefer usability over functionality. Even though this system has some complex functionalities, it will offer mostly used ones as default

preferences. We are approaching this problem by using Pareto Principle (80-20 rule), as we believe that 80% of the users will use the 20% of functionalities.

3.1.2 Cost vs Robustness

Even though we have a limited time, we prefer robustness over cost. This is because bugs in a program may lead to unexpected dangerous results. For example, some connection problems may lead to loss of data, which may result in students to lose points.

3.1.3 Cost vs Reusable

As we prefered robustness over cost, we need to prefer cost over some other thing because of limited time. As we do not have a necessity to design UI libraries ourselves, we are using libraries for this.

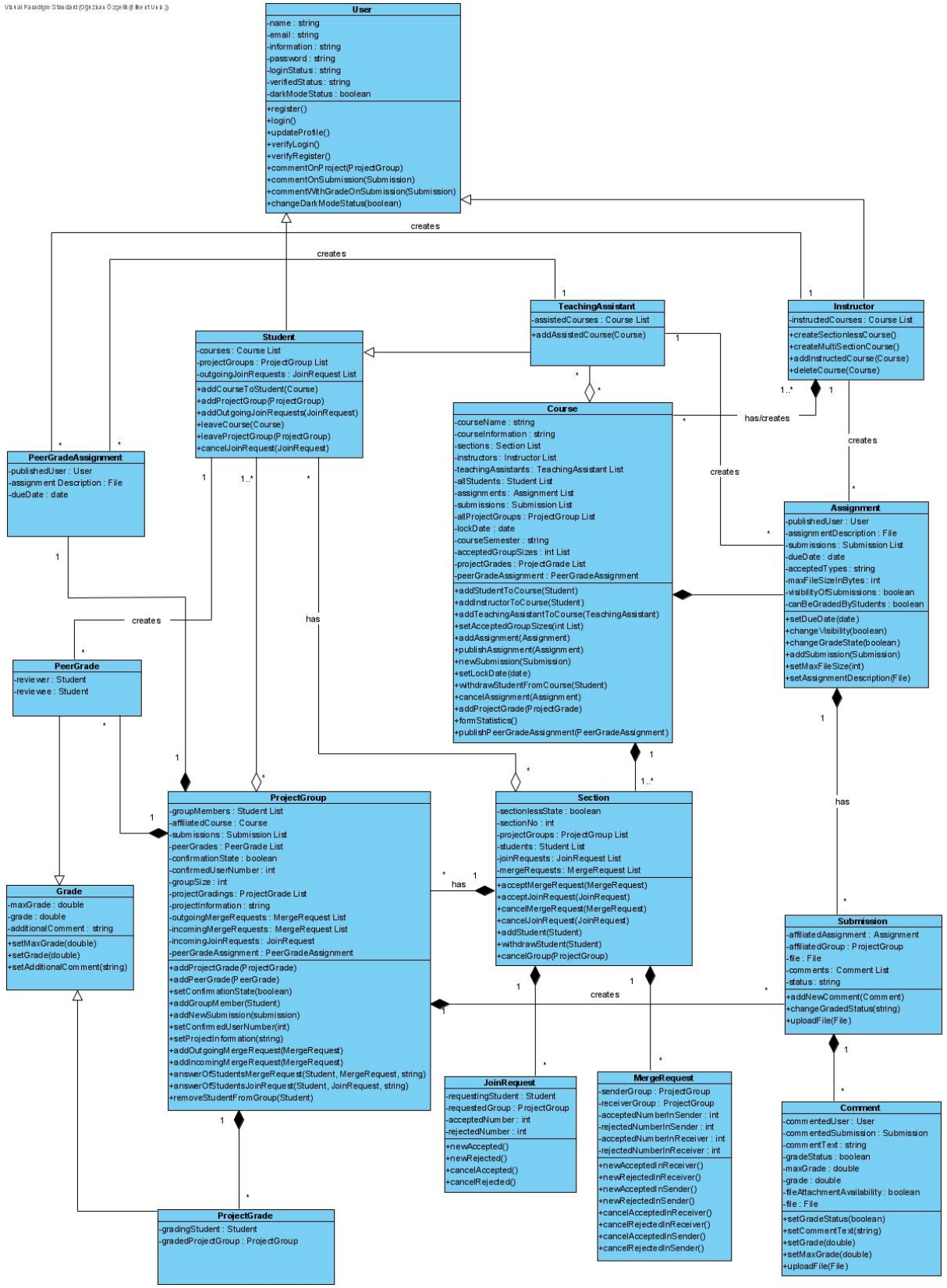
3.1.4 Efficiency vs Portability

We prefer efficiency over portability. It is because generally uploading and publishing is done on computers and also, one time jobs in courses (eg. forming courses, peer reviewing) are done on computers, so we are not going to put too much focus on how our program will look on mobile devices which allow us to focus more on view in computer browsers.

3.1.5 Security vs Usability

We will check students' mails in order to confirm that the student is in Bilkent. However, as we prefer usability over security, we will not have any confirmation other than this. We are doing this because students will use this system frequently and asking for verification in each login would be inconvenient.

3.2 Final Object Design



We did not discover any new reason to change our object design that we conducted in the analysis report.

3.3 Packages

3.3.1 React

In order to provide better user experience in user interface, we are going to use React library for implementing graphical user interface.

3.3.2 Semantic UI

We don’t want to use very casual looking user interfaces in BilHub, so we are going to use the Semantics UI framework in order to prepare fancier user interfaces which will lead to improvement in user experience.

3.3.3 Node Package Manager

To use several libraries and frameworks such as React and Semantic UI in frontend, we will use Node Package Manager (NPM) which provides a lot of user and company defined libraries and frameworks that are being used across thousands of react applications to develop powerful, secure and functional graphics.

3.3.4 .NET Core WEB API

In order to separate the back-end from the front-end we will use an application programing interface.

3.3.5 Entity Framework

In order to directly map our model into a database, we are planning to use Entity framework with a code-first approach. We believe that this will save lots of time.

3.3.6 Swagger

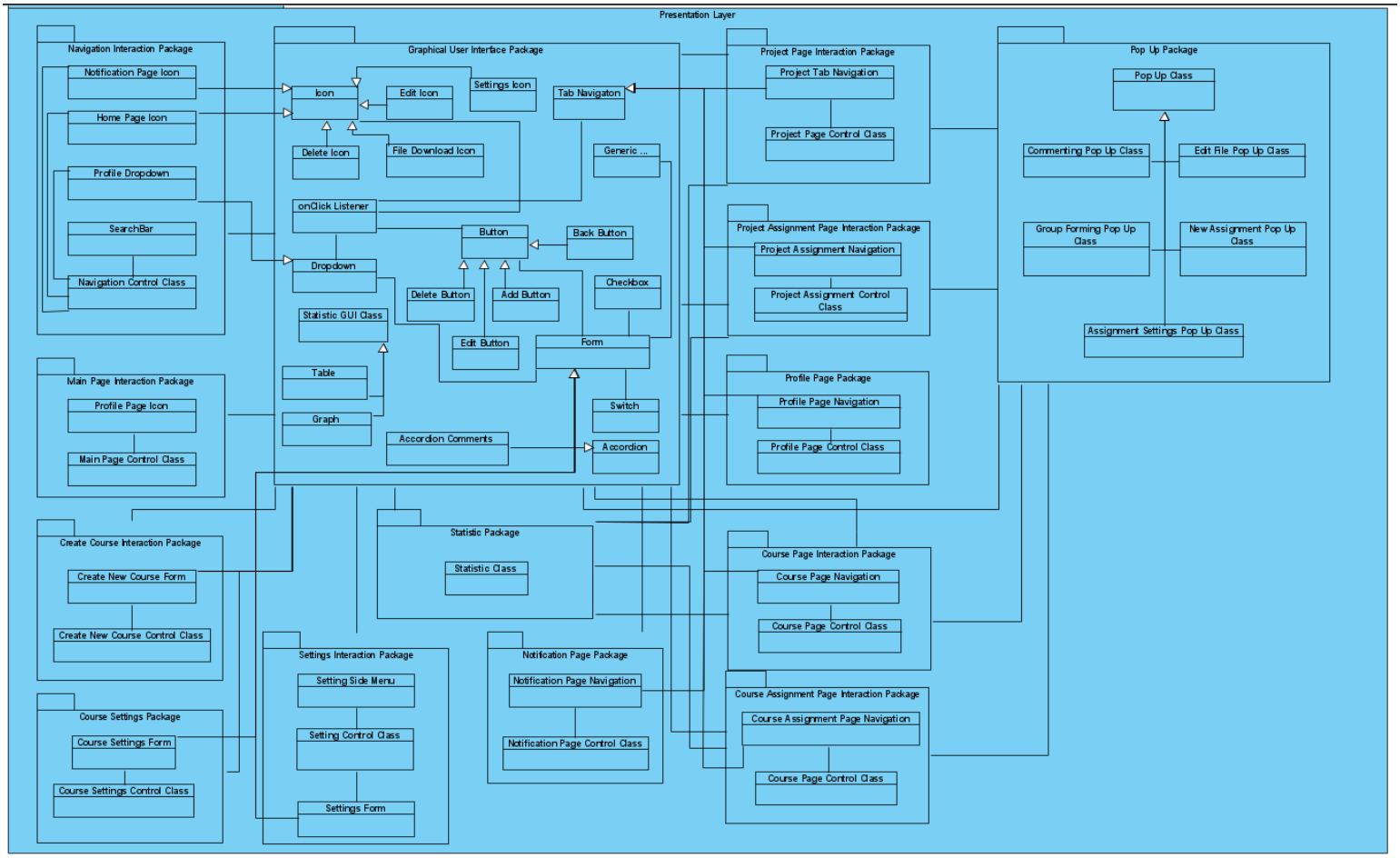
In order to provide a neat documentation system, we will use swagger

3.3.7 MySQL

To provide a powerful, fast and secure database, we will use MySQL database management system. We will connect the MySQL to the Entity Framework to establish connection between backend and database

4. Layer Analysis

4.1 Presentation Layer



This layer is responsible for the communication between users and application. The reason this layer exists is because it will be used by the users. This layer will enable users to use the website and interact with its functionalities.

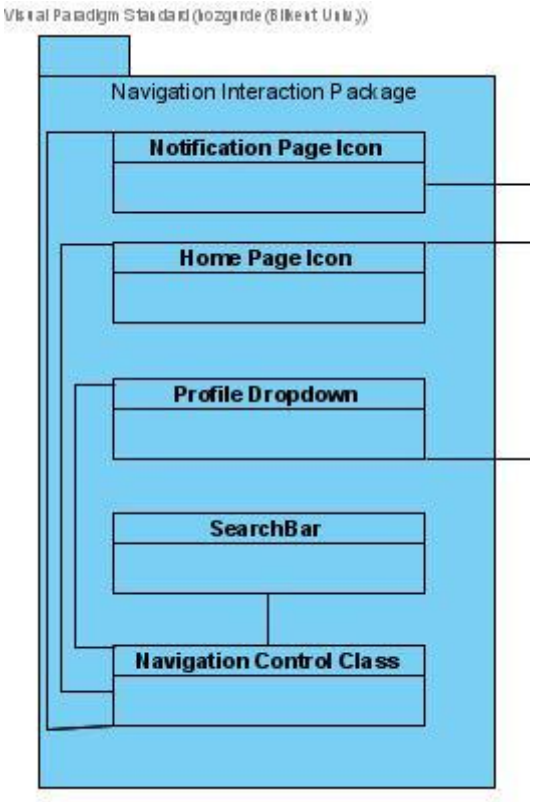
This layer is important because it will visualize REST responses from the application layer. It will also send new REST requests to add, change or delete data from the database through the application layer.

This layer restricts users ability to manipulate the backend. With this layer, we will implement basic validation approaches before sending the data to the backend, such as checking whether given mail is in valid format.

This layer is divided into fourteen packages. One of them is for generic graphical user interface components, eleven of them are for each page interaction, one of them for statistics before visualizing them and one of them is the pop-up that is used throughout several packages. Under these packages, there are several

classes. These classes are correlated with each other since they visualize the components that are in the same page.

4.1.1 Navigation Interaction Package



This package is for the navigation bar at every page of the system. It is an interactive package because it inputs from the user in order s/he to navigate between pages.

-Notification Page Icon Class: This class will inherit the generic icon and is responsible for navigating users to the notification page.

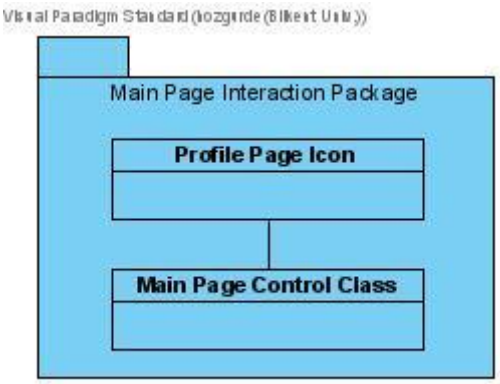
-Home Page Icon Class: This class will inherit the generic icon and is responsible for navigating users to the home page.

-Profile Dropdown Class: This class will inherit generic dropdown and is responsible for showing dropdown to the user.

-SearchBar Class: This class is for writing the place where users want to go manually.

-Navigation control class: This class will be responsible for all the different areas that are in the navigation bar.

4.1.2 Main Page Interaction Package

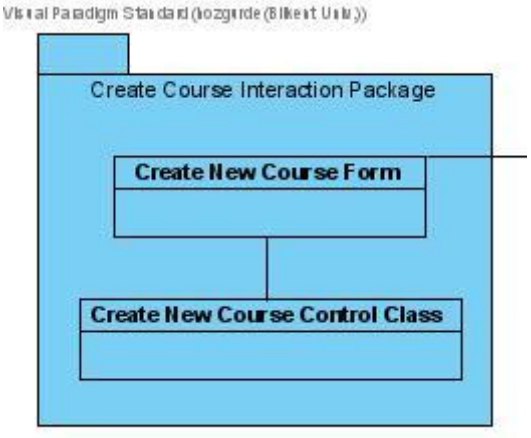


This package is for main page interaction. The user will see this page when they log in.

-Profile Page Icon Class: This class will inherit the generic icon and it will enable the user to go to his/her own profile page.

-Main Page Control Class: This class will be responsible for all the different areas that are in the main page. It will have click listeners for going to specific sections (such as via clicking the assignment name, the user will be able to go to that assignment's page). This class will aggregate a lot of components from the graphical user interface class to display information that is sent by the application layer for the main page. This class also will use the profile page icon class.

4.1.3 Create Course Interaction Package

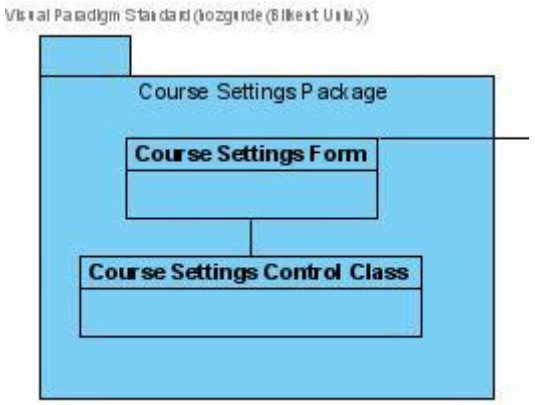


This package is for creating course interaction. The user will see this page when they are creating a course.

-Create New Course Form: This form will inherit the generic form class that is in the graphical user interface class. It will have areas for course name and course information.

-Create New Course Control Class: This class will be responsible for all the different areas that are in the create new course page. It will create a new course form to interact with the user and then send the new data to the application layer.

4.1.4 Course Settings Package

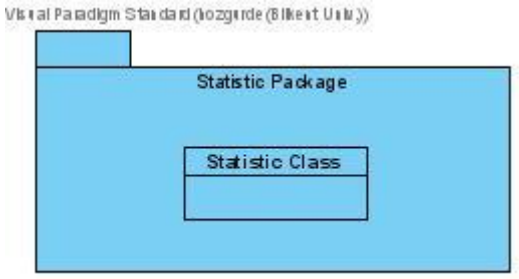


This package is for course settings page interaction. The instructors will see this page when they want to change the course settings. All the related classes will be under this package.

-Course Settings Form: This form will inherit the generic form class that is in the graphical user interface class. It will have areas for adding/removing instructors/TAs/students as well as changing the sections and groups.

-Course Settings Control Class: This class will be responsible for all the different areas that are in the course setting page. It will have a course setting form to interact with the user and then send the new data to the application layer.

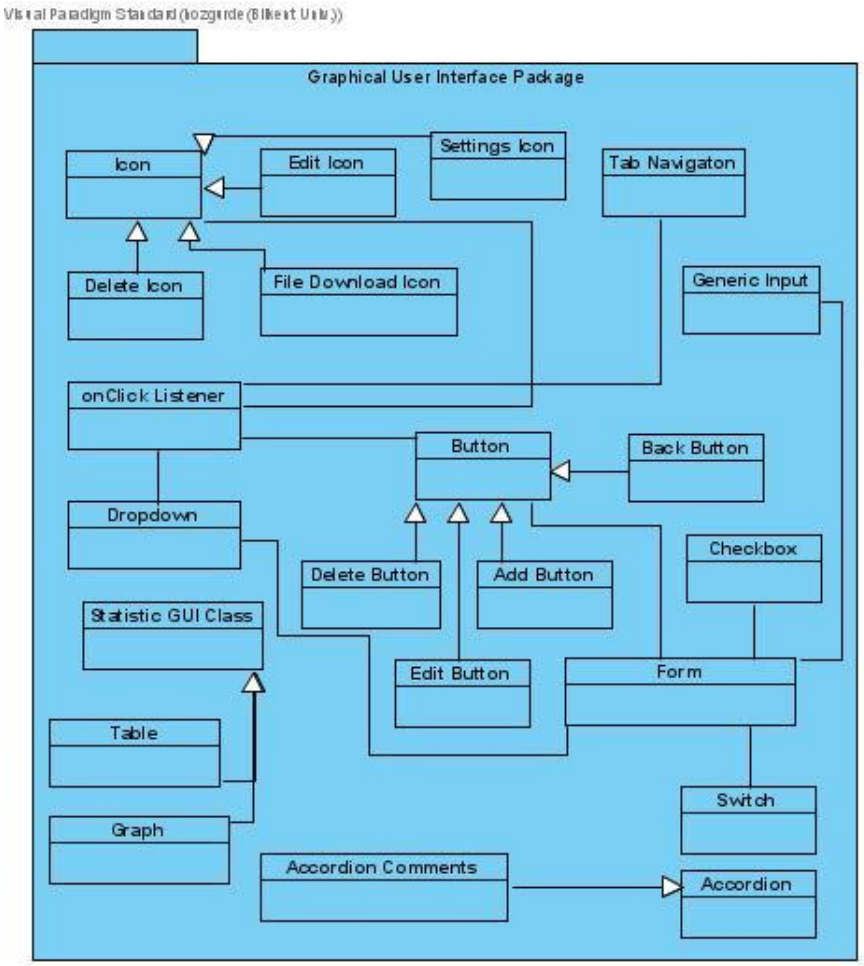
4.1.5 Statistic Package



This package will consist of one class. In the future may be with possible statistics analysis classes.

-Statistic Class: This will contain the raw statistical data without any graphics that can be used by the graphical statistic class and subclasses.

4.1.6 Graphical User Interface Package



This package is for generic graphical user interface components that are going to be used by all the other packages in the presentation layer. This package ultimately will enable users to utilize the website with different I/O components that are styled.

-Button Class: This class will be the base for all the buttons that are going to be used.

-Statistic GUI Class: This class will be the base for tables and graphs that will help users to see different and interesting data.

-Accordion Comments Class: This class will enable users to see different comments from the different types of users.

-Icon Class: This class will provide an abstract class for all the icons that are going to be used by other packages.

-onClick Listener Class: This class will listen to the clicks of the user. This class will be used by all the components that interact with the user.

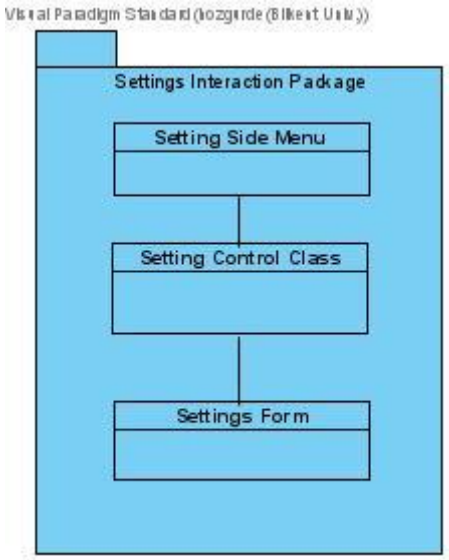
-Dropdown Class: This class will enable users to choose from different options.

-Switch/Checkbox Class: This class will enable users data about boolean values.

-Form Class: This class will enable users to give inputs about specific objects at one time.

-Tab Navigation Class: This class will enable users to see different sectional parts related to a specific page. For example, in the profile page, the user will be able to see that user's previous projects as well as instructed classes.

4.1.7 Settings Interaction Package



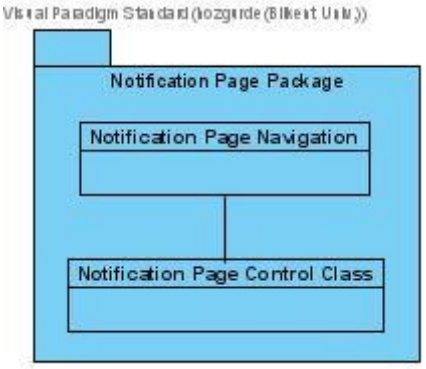
Settings interaction package is for interactions inside the settings page. Users can navigate this page by using the dropdown in the navigation bar.

-Setting Side Menu Class: This class is for showing a side menu to users in order to change settings of the system they are using.

-Setting Control Class: tThis class will be responsible for all the different areas that are in the settings page. It uses a side menu and settings form for changing settings of system and user.

-Settings Form Class: This class will inherit generic form class and is for taking necessary information from users as a form.

4.1.8 Notification Page Package

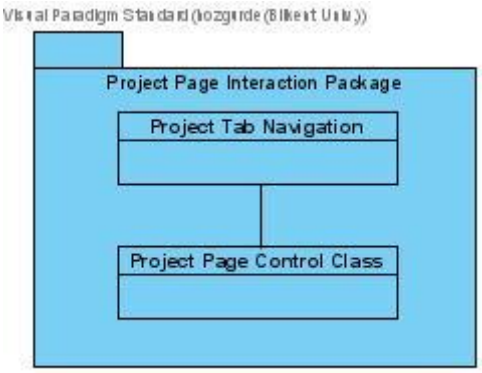


This package is for the notification page. Users can access this page using the tab icon in the navigation bar.

- Notification Page Navigation Class: This navigation class will inherit the tab section class that is in the graphical user interface class. It will contain three bars: Incoming Request, Your Requests, and New Comments. This class will use lots of icons and buttons from the graphical user interface class.

-Notification Page Control Class: This class will be responsible for all the different areas that are in the notification page. It will have a notification page navigation class to interact with the user.

4.1.9 Project Page Package

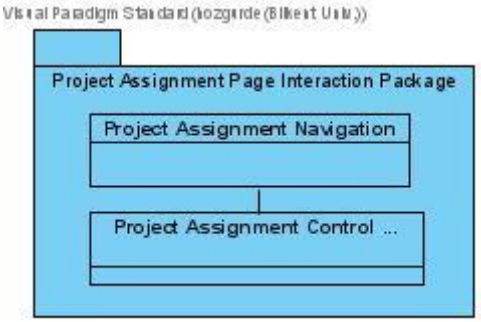


This package is for showing a project page to users and allowing them to interact with this page. Users can navigate this page by using the projects shown in their main page.

-Project Tab Navigation Class: This class will inherit tab navigation and is for navigating between subpages in the project page.

-Project Page Control Class: This class will be responsible for all the different areas that are in the project page. It will navigate between subpages and apply things done inside these subprograms.

4.1.10 Project Assignment Interaction Package

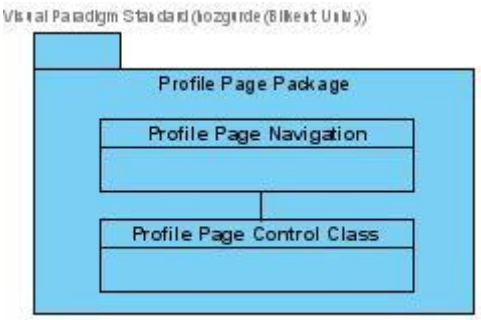


This package is for the project's assignment page interaction. The users will see this page to get information about a specific assignment that belongs to a specific project . All the related classes will be under this package.

-Project Assignment Navigation Class: This navigation class will inherit the tab section class that is in the graphical user interface class. It will contain three bars: Submission, Grades, and Comments. This class will use lots of icons and buttons from the graphical user interface class.

-Project Assignment Control Class: This class will be responsible for all the different areas that are in the project assignment page. It will have a project assignment navigation class to interact with the user as well as the left section to show information that is sent by the application layer.

4.1.11 Profile Page Package

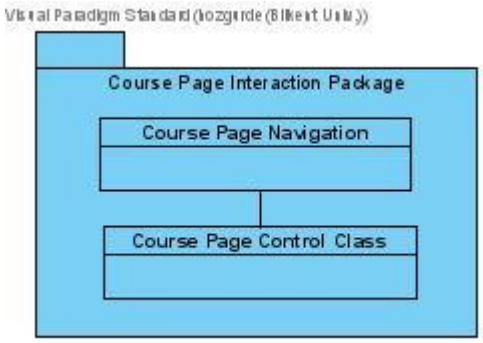


This package is for the profile page interaction. The users will see this page to get information about a user. All the related classes will be under this package.

-Profile Page Navigation Class: This navigation class will inherit the tab section class that is in the graphical user interface class. It will contain three bars: Projects and Instructed Courses. This class will use lots of icons and buttons from the graphical user interface class.

-Profile Page Control Class: This class will be responsible for all the different areas that are in the profile page. It will have a profile page navigation class to interact with the user as well as the left section to show information that is sent by the application layer.

4.1.12 Course Page Interaction Package

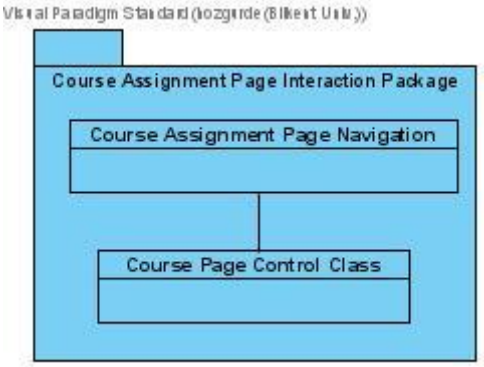


This package is for the course page interaction. The users will see this page to get information about the course and interact with it. All the related classes will be under this package.

-Course Page Navigation Class: This navigation class will inherit the tab section class that is in the graphical user interface class. It will contain three bars: Groups,Statistics and Assignments. This class will use lots of icons and buttons from the graphical user interface class.

-Course Page Control Class: This class will be responsible for all the different areas that are in the course page. It has a bar at left which shows the information of the class. It will have a course page navigation class to interact with the user.

4.1.13 Course Assignment Page Interaction Package

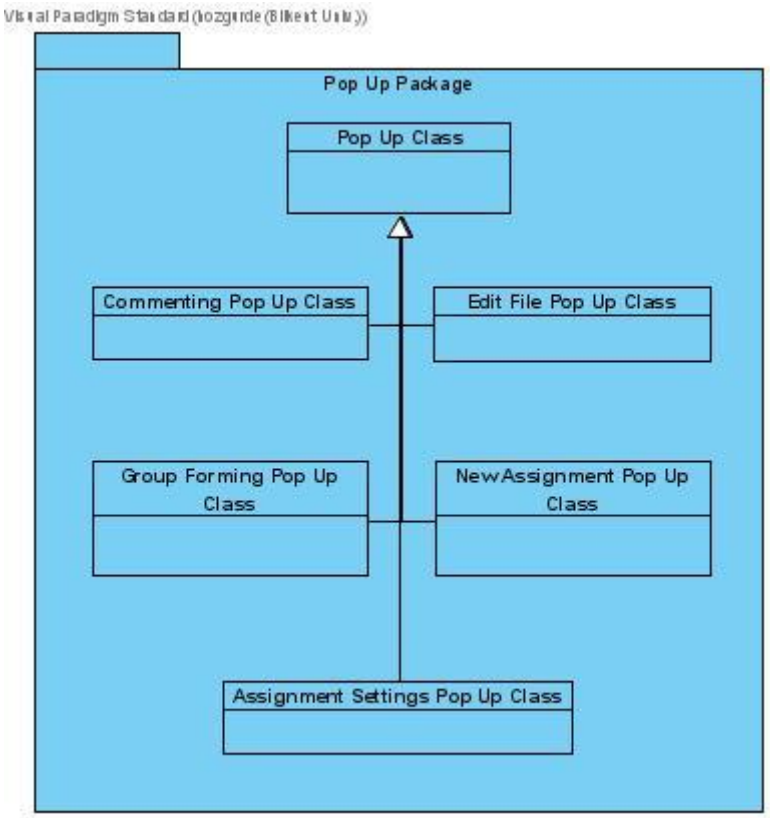


This package is for showing the course assignment page to users and allowing them to interact with this page. Users can navigate this page by using the assignments shown in their course page.

-Course Assignment Page Navigation Class: This class will inherit tab navigation and is for navigating between subpages in the course assignment page.

-Course Assignment Page Control Class: This class will be responsible for all the different areas that are in the course assignment page. It will have a course assignment page navigation class to interact with the user as well as the left section to show information of course.

4.1.14 Pop Up Package



This package will contain implementations regarding all the pop-ups that are going to be used in the pages.

-Pop Up Class: Generic class that implements common features of the customized pop ups

-Commenting Pop Up Class: A class that shows a pop up for commenting to assignments.

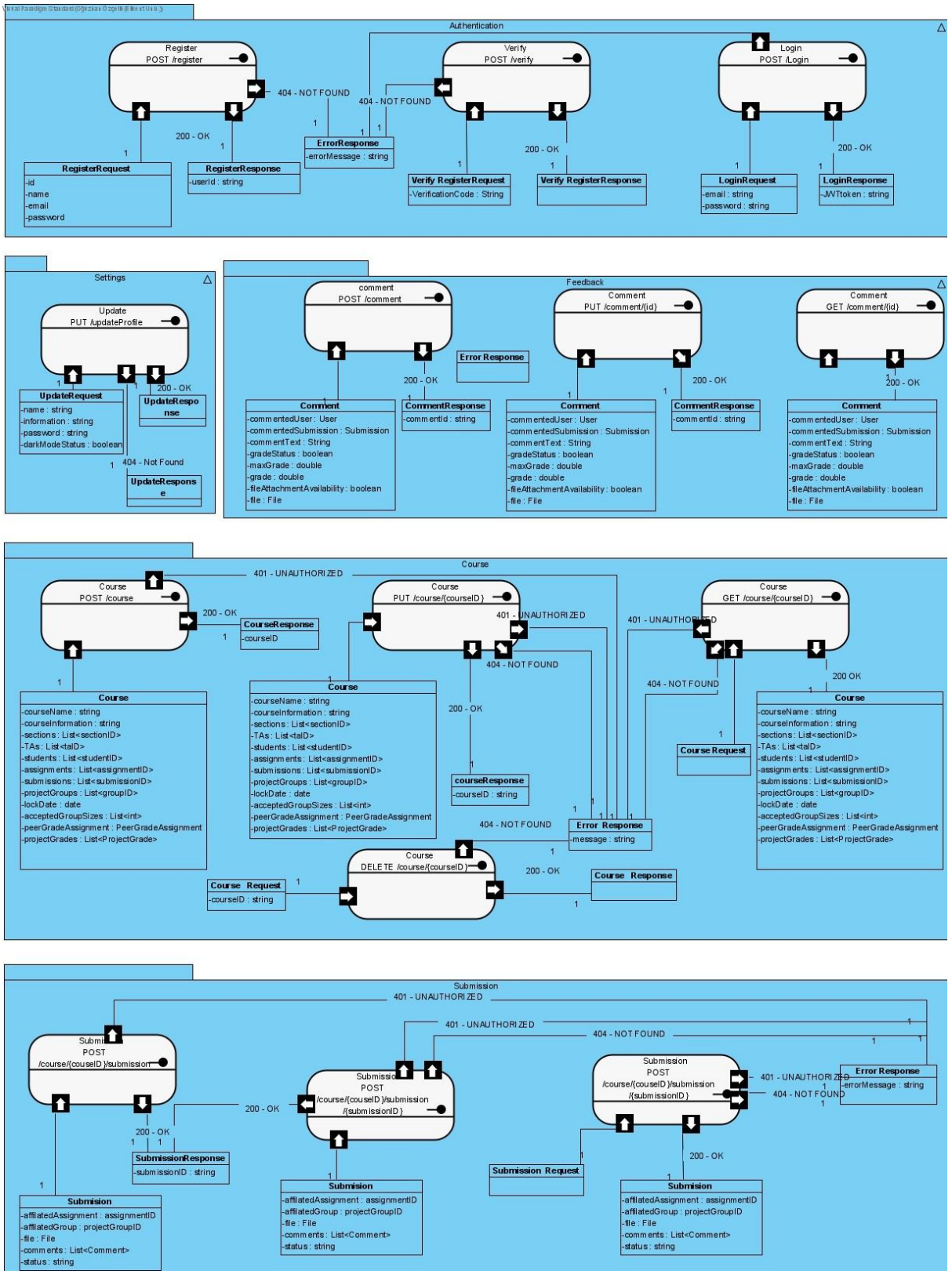
-Edit File Pop Up Class: A class that shows a pop up for editing files that are uploaded to assignments.

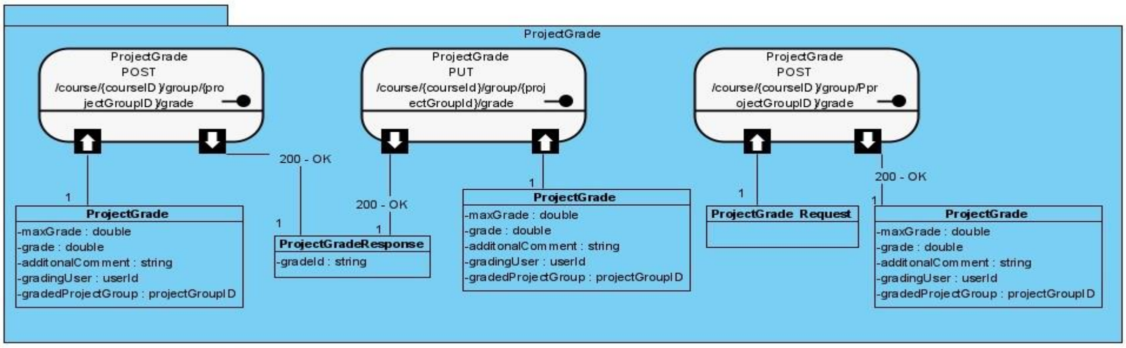
-Group Forming Pop Up Class: A class that shows a pop up for forming groups to attend or exit from the group.

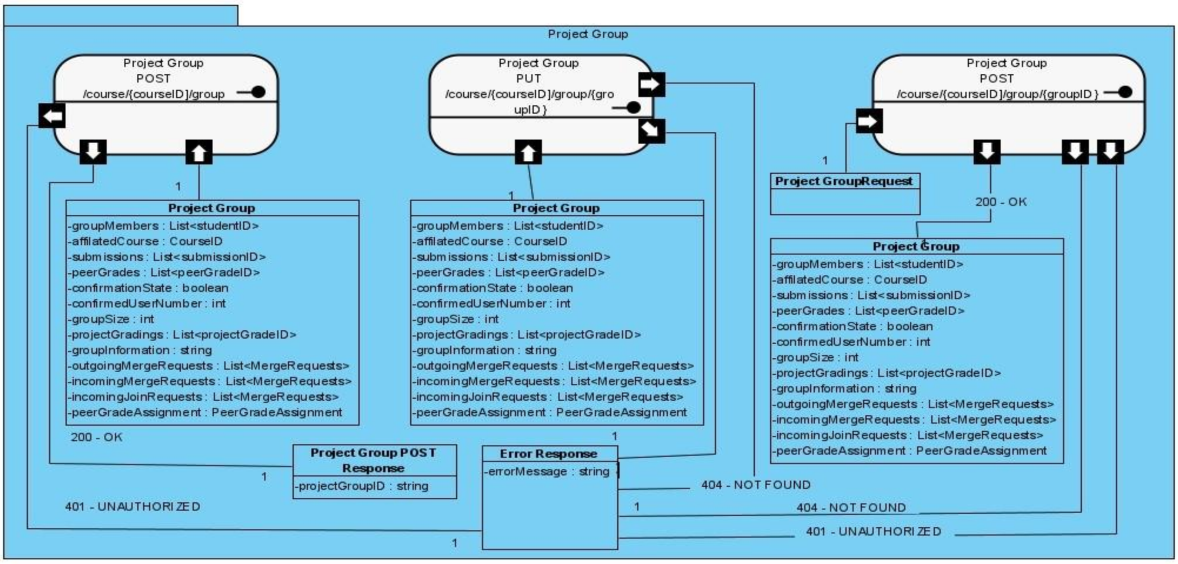
-New Assignment Pop Up Class: A class that shows a pop up for new assignment creation.

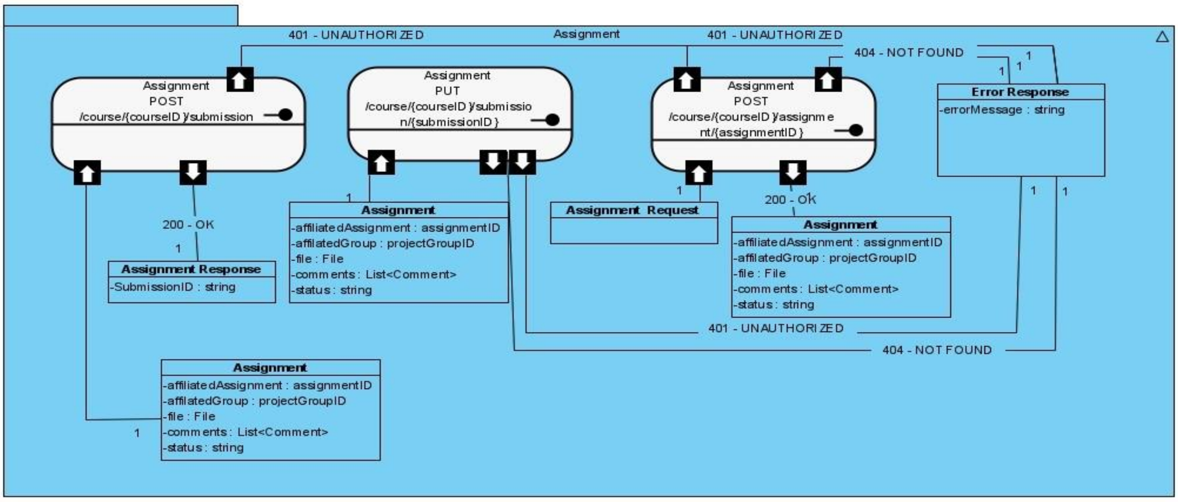
-Assignment Settings Pop Up Class: A class that shows a pop up for adjusting assignment settings.

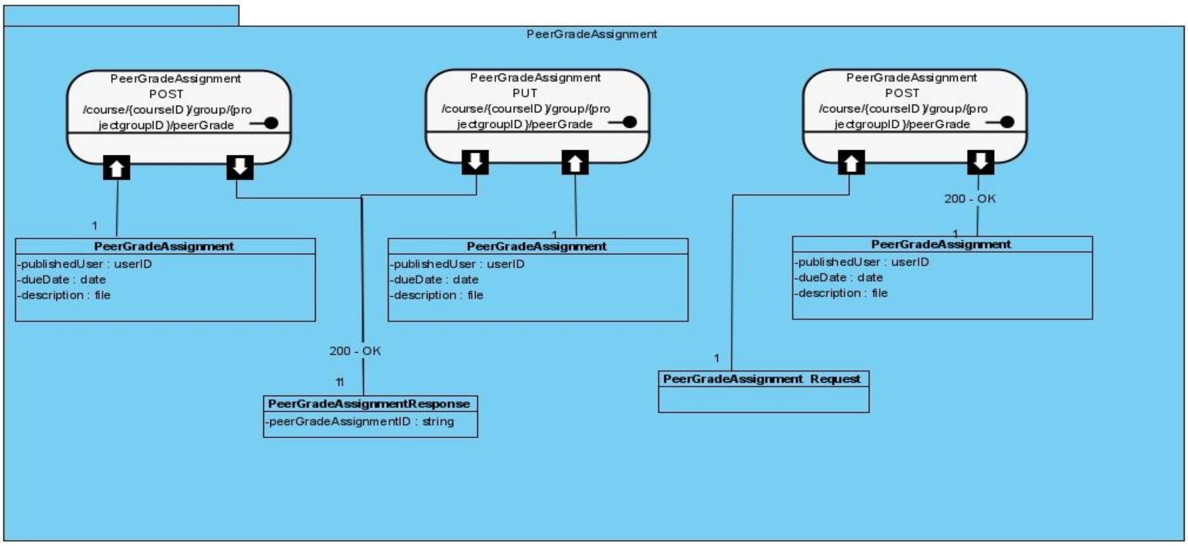
4.2 Application Layer




ProjectGrade
ProjectGrade
POST
/course/{courseID}/group/{projectGroupID}/grade
200 - OK
1
ProjectGrade
-maxGrade : double
-grade : double
-additonalComment : string
-gradingUser : userId
-gradedProjectGroup : projectGroupID
1
ProjectGradeResponse
-gradeId : string
ProjectGrade
PUT
/course/{courseId}/group/{projectGroupId}/grade
200 - OK
1
ProjectGrade
-maxGrade : double
-grade : double
-additonalComment : string
-gradingUser : userId
-gradedProjectGroup : projectGroupID
ProjectGrade
POST
/course/{courseID}/group/Pprojectgroupid}/grade
1
ProjectGrade Request
200 - OK
1
ProjectGrade
-maxGrade : double
-grade : double
-additonalComment : string
-gradingUser : userId
-gradedProjectGroup : projectGroupID

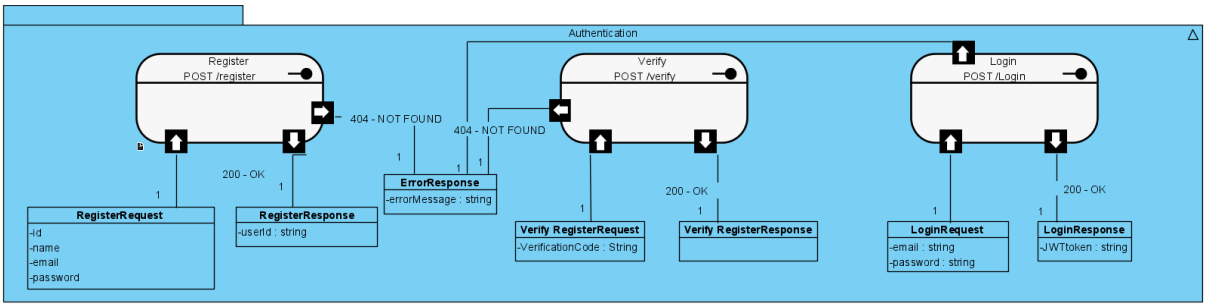

Project Group
Project Group
POST
/course/{courseID}/group
1
Project Group
-groupMembers : List<studentID>
-affilatedCourse : CourseID
-submissions : List<submissionID>
-peerGrades : List<peerGradeID>
-confirmationState : boolean
-confirmedUserNumber : int
-groupSize : int
-projectGradings : List<projectGradeID>
-groupInformation : string
-outgoingMergeRequests : List<MergeRequests>
-incomingMergeRequests : List<MergeRequests>
-incomingJoinRequests : List<MergeRequests>
-peerGradeAssignment : PeerGradeAssignment
200 - OK
Project Group POST Response
-projectGroupID : string
401 - UNAUTHORIZED
Project Group
PUT
/course/{courseID}/group/{groupID}
Project Group
-groupMembers : List<studentID>
-affilatedCourse : CourseID
-submissions : List<submissionID>
-peerGrades : List<peerGradeID>
-confirmationState : boolean
-confirmedUserNumber : int
-groupSize : int
-projectGradings : List<projectGradeID>
-groupInformation : string
-outgoingMergeRequests : List<MergeRequests>
-incomingMergeRequests : List<MergeRequests>
-incomingJoinRequests : List<MergeRequests>
-peerGradeAssignment : PeerGradeAssignment
Error Response
-errorMessage : string
404 - NOT FOUND
404 - NOT FOUND
401 - UNAUTHORIZED
Project Group
POST
/course/{courseID}/group/{groupID}
Project GroupRequest
200 - OK
Project Group
-groupMembers : List<studentID>
-affilatedCourse : CourseID
-submissions : List<submissionID>
-peerGrades : List<peerGradeID>
-confirmationState : boolean
-confirmedUserNumber : int
-groupSize : int
-projectGradings : List<projectGradeID>
-groupInformation : string
-outgoingMergeRequests : List<MergeRequests>
-incomingMergeRequests : List<MergeRequests>
-incomingJoinRequests : List<MergeRequests>
-peerGradeAssignment : PeerGradeAssignment


Assignment
401 - UNAUTHORIZED
401 - UNAUTHORIZED
404 - NOT FOUND
Assignment
POST
/course/{courseID}/submission
200 - OK
Assignment Response
-SubmissionID : string
Assignment
PUT
/course/{courseID}/submission/{submissionID}
Assignment
-affiliatedAssignment : assignmentID
-affilatedGroup : projectGroupID
-file : File
-comments : List<Comment>
-status : string
Assignment
POST
/course/{courseID}/assignment/{assignmentID}
Assignment Request
200 - OK
Assignment
-affiliatedAssignment : assignmentID
-affilatedGroup : projectGroupID
-file : File
-comments : List<Comment>
-status : string
Error Response
-errorMessage : string
401 - UNAUTHORIZED
404 - NOT FOUND
Assignment
-affiliatedAssignment : assignmentID
-affilatedGroup : projectGroupID
-file : File
-comments : List<Comment>
-status : string


PeerGradeAssignment
PeerGradeAssignment
POST
/course/{courseID}/group/{projectgroupID}/peerGrade
PeerGradeAssignment
-publishedUser : userID
-dueDate : date
-description : file
PeerGradeAssignment
PUT
/course/{courseID}/group/{projectgroupID}/peerGrade
PeerGradeAssignment
-publishedUser : userID
-dueDate : date
-description : file
200 - OK
PeerGradeAssignmentResponse
-peerGradeAssignmentID : string
PeerGradeAssignment
POST
/course/{courseID}/group/{projectgroupID}/peerGrade
200 - OK
PeerGradeAssignment
-publishedUser : userID
-dueDate : date
-description : file
PeerGradeAssignment Request

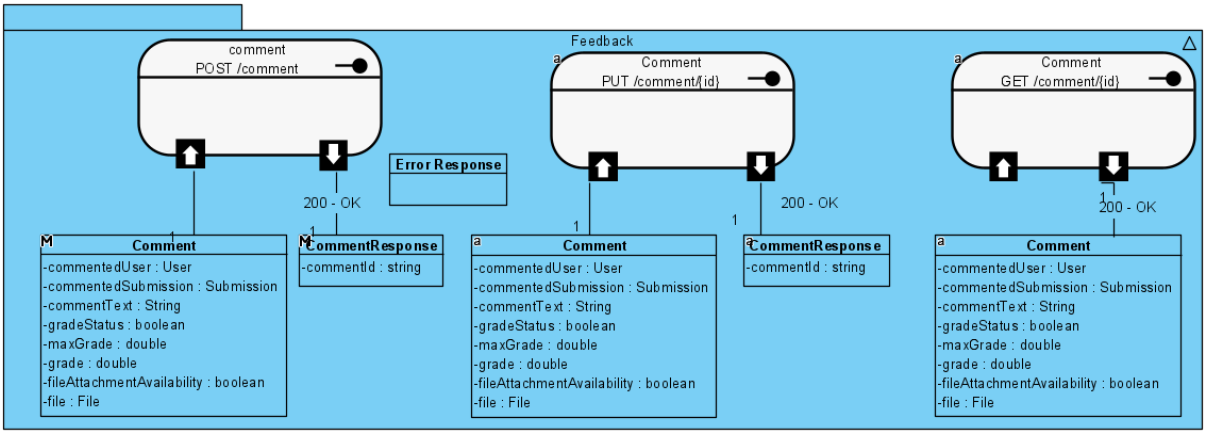
In order to communicate with the data layer we designed an Application programmer interface in order to make abstractions that will make our developers work without in both front-end and back-end.

Authentication



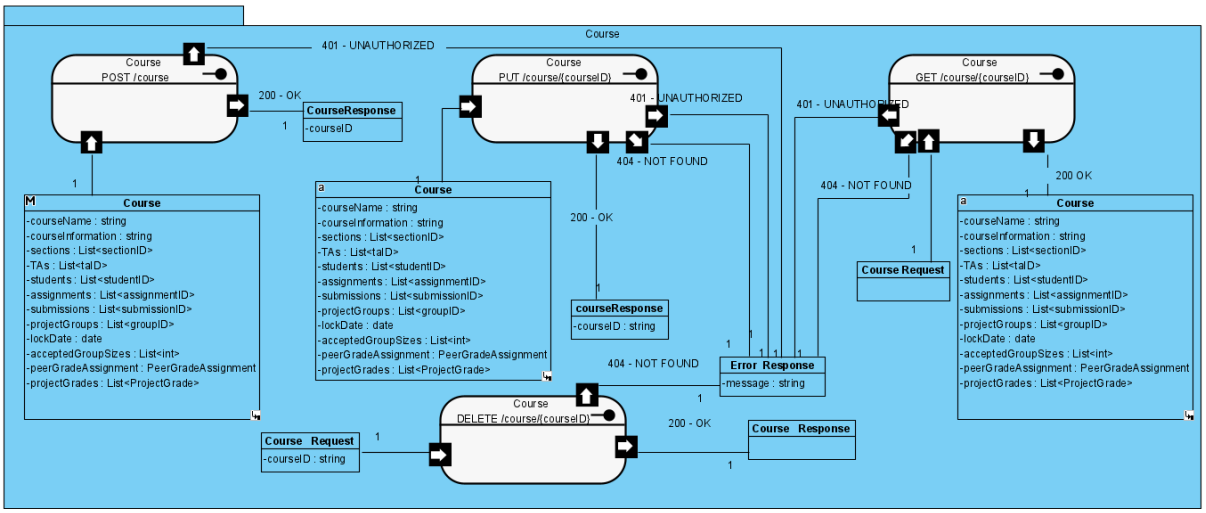
Find above the endpoints for authentication related REST services. Once the user registers the system, a new and unique ID is registered into the database for that user. In order to log in to the system user needs to enter the correct credentials thereafter the user will reach a JWT token that s/he will use in order to authorize to use other endpoints of the API.

Feedback System



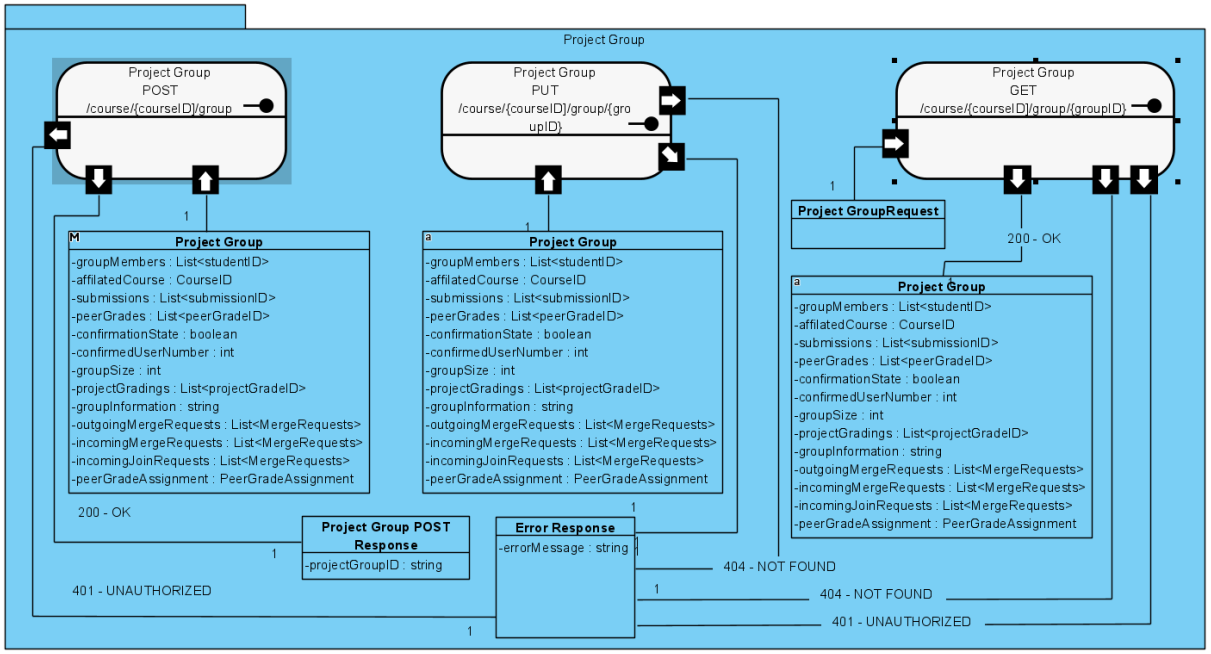
When a user uses the POST/comment endpoint with the parameters that is wanted from him/her a new comment will be registered to the database and a new unique commentID is created. When a user uses the PUT/comment endpoint with the parameters that s/he wanted to change and is able to change, comment in database with unique commentID will be changed accordingly. All the users will be able to see comments by using the GET/comment endpoint.

Course



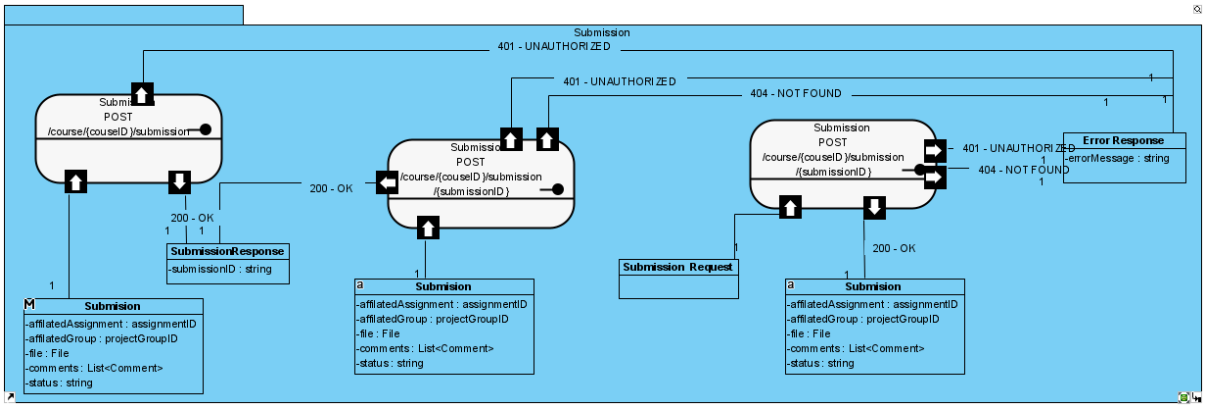
When an instructor uses the POST /course endpoint with the parameters s/he wants to initiate the course object with, a new course object will be registered into the database and a new and unique courseID will be created. If the user is not authorized to use the endpoint an error message will be returned with related response code. POST PUT and DELETE endpoints will only be available for instructors but GET endpoint will also be available to student users.

Project Group



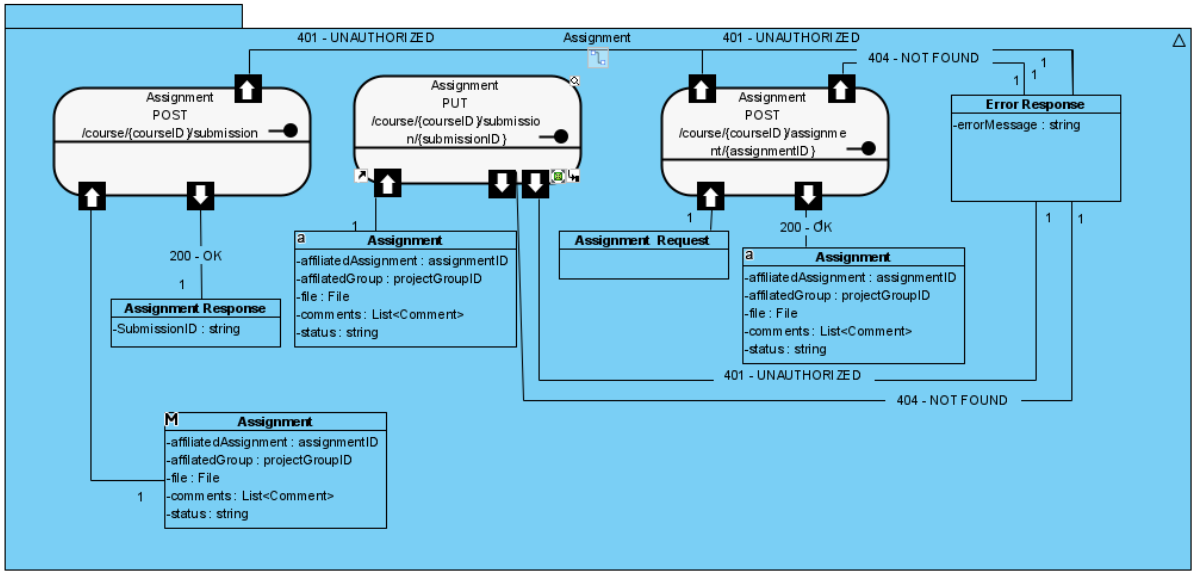
When a student uses the POST / course/{courseID}/group endpoint, a new project group object will be registered into the database and a new and unique projectGroupID will be created. This happens when a new student is added to the course. If the user is not authorized to use the endpoint an error message will be returned with related response code. POST, PUT and GET endpoints will be available to both instructor and student users.

Submission



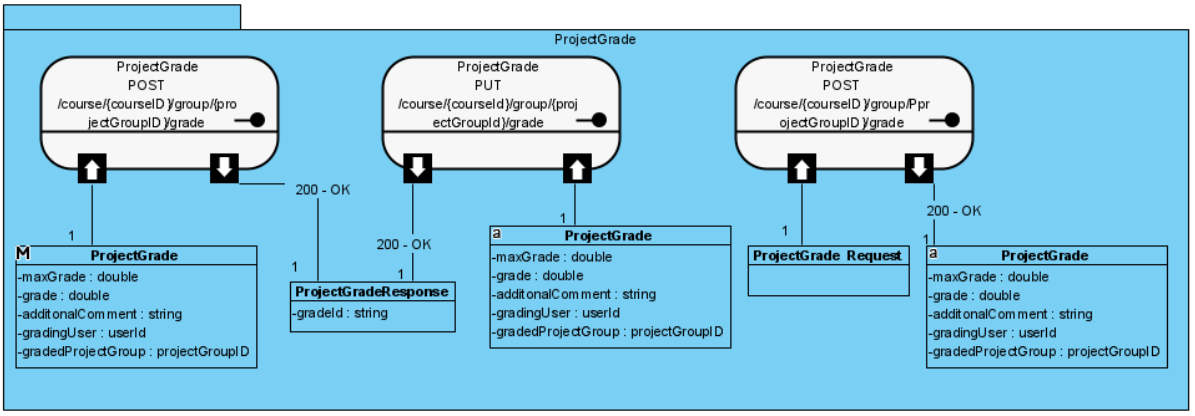
When a student uses the POST / course/{courseID}/submission endpoint, a new Submission object will be registered into the database and a new and unique submissionID will be created. If the user is not authorized to use the endpoint an error message will be returned with related response code. POST endpoints will only be available for students.

Assignment



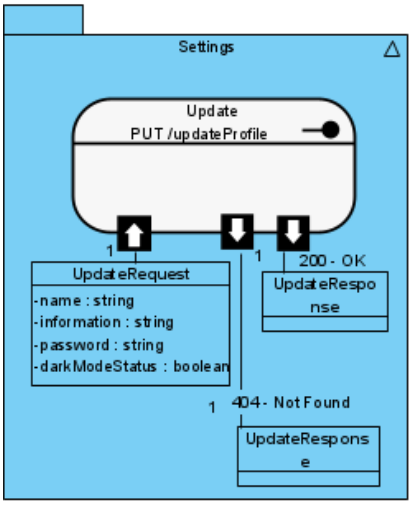
When an instructor uses the POST / course/{courseID}/assignment endpoint, a new Assignment object will be registered into the database and a new and unique assignmentID will be created. If the user is not authorized to use the endpoint an error message will be returned with related response code. POST and PUT endpoints will only be available for instructors but GET endpoint will also be available to student users.

ProjectGrade



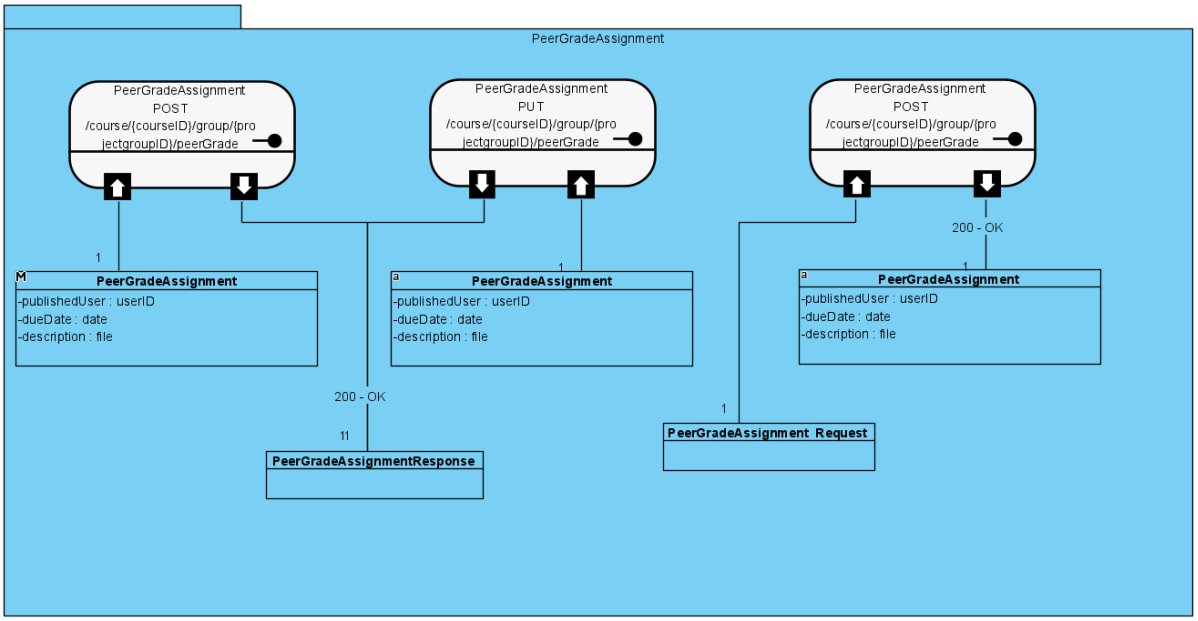
When a user uses the POST / course/{courseID}/group/{projectGroupID} endpoint, a new ProjectGrade object will be registered into the database and a new and unique projectGradeID will be created. POST, PUT and GET endpoints will only be available for instructors and student users.

Settings



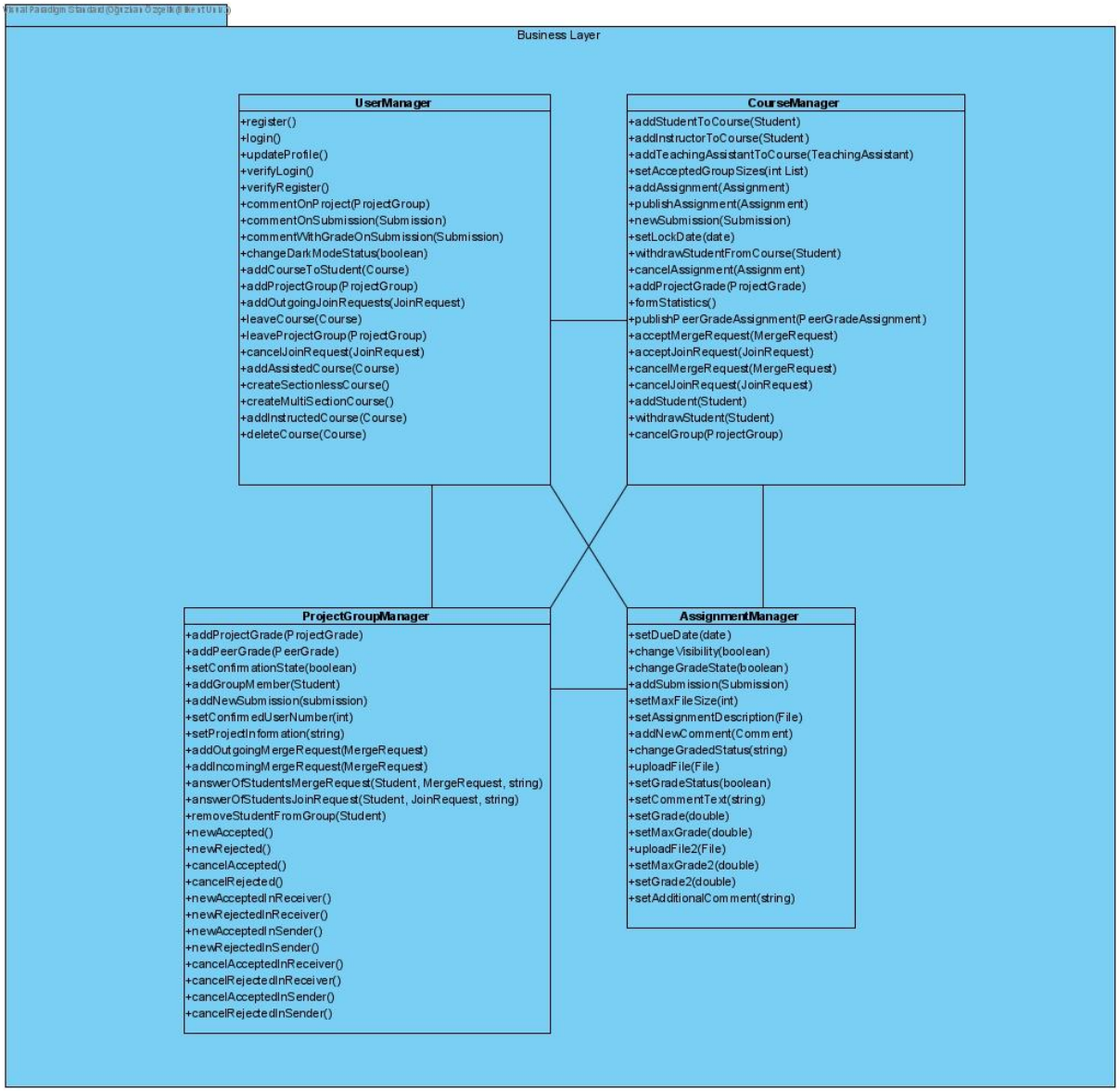
When a user uses the PUT /updateProfile endpoint, s/he can change settings. PUT endpoint will be available for both instructors and student users.

Peer Grade Assignment



When an instructor uses the POST /course/{courseID} /group/ {projectGroupID}/peerGrade endpoint, a new PeerGradeAssignment object will be registered into the database. POST and PUT endpoints will only be available for instructors but GET endpoint will also be available to student users.

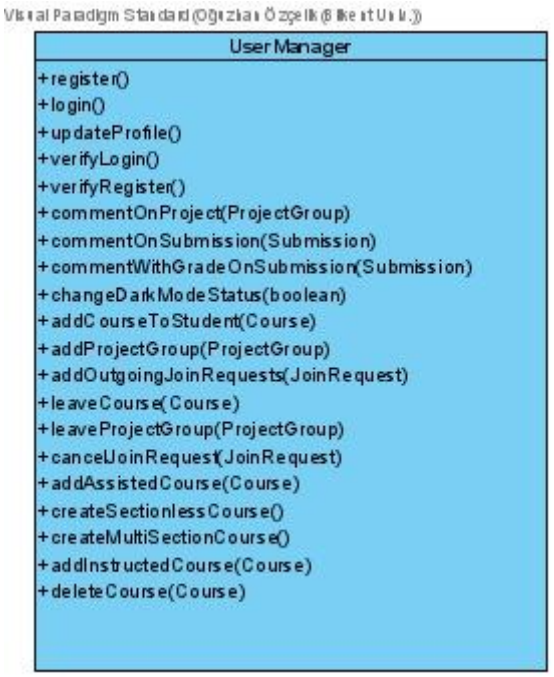
4.3 Business Layer



This layer is responsible for the communication between database and application. The reason this layer exists is because, whenever we need to change the database structure (may happen due to technical or choice reasons), we will only need to modify this layer and other layers will not be affected from the changes, which will make things easier for developers and also, debugging will be easier.

This layer is divided into four classes each are responsible for their corresponding database classes. Also, these layers' classes act like a subsystem. These classes are correlated with each other since one change/addition in a database class may affect other classes (e.g: whenever a student is enrolled in a course, both course database and student database will be affected).
Since this layer is responsible for only communication, it does not have its own attributes. Each four classes have its own methods. These methods are explained below.

4.3.1 UserManager Class



- register()
  This method is responsible for registration of users into the system.
- login()
  This method enables the user to log in to the system.
- updateProfile()
  This is for updating the user's profile in the system
- verifyLogin()
  This method is a helper of login() method. It provides the verification of login.
- verifyRegister()

This method is a helper of register() method. It provides the verification of registration.

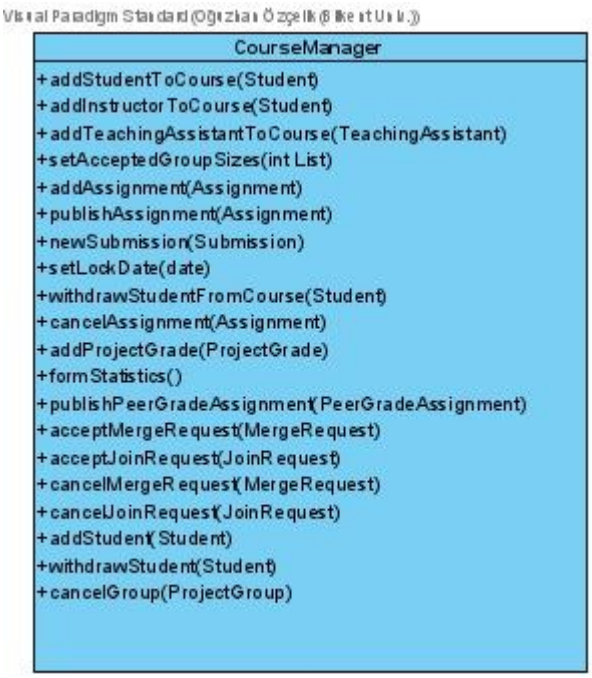
- commentOnProject(ProjectGroup)
  This method is responsible for a user to comment on a project
- commentOnSubmission(Submission)
  This method is for making a comment on a submission.
- commentWithGradeOnSubmission(Submission)
  With this method, a user can make a comment on submission while also providing a grade for this submission
- changeDarkModeStatus(boolean)
  If boolean is true, it changes the view to dark mode, otherwise light mode. The reason this method exists is that regardless of the browser or device the user is using, dark mode status is kept.

- addCourseToStudent(Course)
  Whenever a student is enrolled in a course, this method enables to change the user table to include newly added course
- addProjectGroup(ProjectGroup)
  This method is for students. It changes the user table to include his/her new project group
- addOutgoingJoinRequest(JoinRequest)
  Whenever a student requests to join another group, a new JoinRequest object is included in the user table.
- leaveCourse(Course)
  Whenever a student drops/withdraws a course, the database is updated with this method.
- leaveProjectGroup(ProjectGroup)
  Whenever a student leaves a project group, the database is updated with this method.
- cancelJoinRequest(JoinRequest)
  If a student cancels his/her join request, the database is updated with this method.
- addAssistedCourse(Course)

This method is for TAs. With this method, the database is updated with the information of the new TA of the corresponding course.

- createSectionlessCourse()
  This method is for Instructors. Whenever an instructor chooses to open a sectionless course, this method is being called to make necessary updates in the database.
- createMultiSectionCourse()
  This method is for Instructors. Whenever an instructor chooses to open a multi section course, this method is being called to make necessary updates in the database.
- addInstructedCourse(Course)
  This method is for Instructors. Whenever an instructor is added to a course by another instructor (because only one of them creates the course object; other one must be added later), this method does the necessary changes in the database.
- deleteCourse(Course)
  This method is for Instructors. If instructor decides to delete the course (this may happen due to the mistake in the setup phase. e.g: naming the course wrong) this method is being used.

4.3.2 CourseManager Class

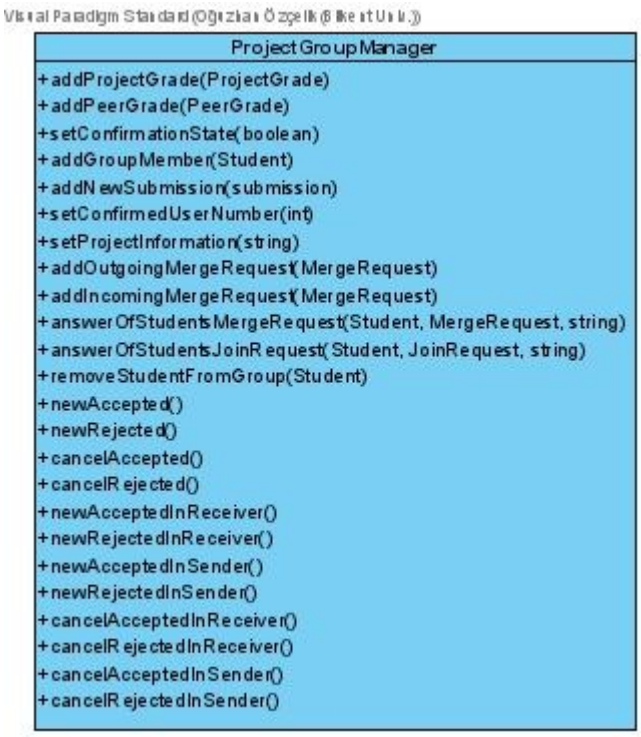


- addStudentToCourse(Student)
  When a student is enrolled in the course, this method makes the changes in the course database table.
- addInstructorToCourse(Instructor)
  When an instructor is added to the course, this method makes the changes in the course database table.
- addTeachingAssistantToCourse(TeachingAssistant)
  When a TA is added to the course, this method makes the changes in the course database table.
- setAcceptedGroupSizes(int List)
  Instructor chooses the accepted group size in projects. With this method, the database is updated.
- addAssignment(Assignment)
  When an instructor or TA adds a new assignment, this method does the necessary changes in the database.
- publishAssignment(Assignment)

When an instructor or TA makes the assignment visible (for students to see), this method is being called.

- newSubmission(Submission)
  If a student/project group submits a submission to an assignment given in this course, this method is called.
- setLockDate(date)
  When an instructor or TA decides when is the last time groups can be formed, this method is called.
- withdrawStudentFromCourse(Student)
  When a student leaves the course, this method is called.
- cancelAssignment(Assignment)
  When an assignment is being cancelled, this method is called.
- addProjectGrade(ProjectGrade)
  ProjectGrade is a class in our system and whenever a user grades a project, this method is called to make the proper changes in the corresponding course.
- formStatistics()
  When Instructors or TAs decide to show the statistics of the course, this method is being called.
- publishPeerGradeAssignment(PeerGradeAssignment)
  When a user makes a peer grade, this method is being called for the course table to make proper changes.
- acceptMergeRequest(MergeRequest)
  This method is for accepting the merge requests of project groups.
- acceptJoinRequest(JoinRequest)
  This method is for accepting the join request of a student to a project group.
- cancelMergeRequest(MergeRequest)
  This method is for cancelling the merge requests of project groups.
- cancelJoinRequest(JoinRequest)
  This method is for cancelling the join request of a student to a project group.

4.3.3 ProjectGroupManager Class



- addProjectGrade(ProjectGrade)
  This method is for adding a project grade for this project group.
- addPeerGrade(PeerGrade)
  When a student of this project group makes a peer grading, this method updates the database.
- setConfirmationState(boolean)
  If a group is finalized, this method is being called.
- addGroupMember(Student)
  This method adds students to the group.
- addNewSubmission(Submission)
  When a member of this group makes a submission for an assignment, this method does the necessary changes in the database.
- setConfirmedUserNumber(int)
  When a member candidate confirms to be in this group, the number is increased and changes in the database.
- setProjectInformation(string)

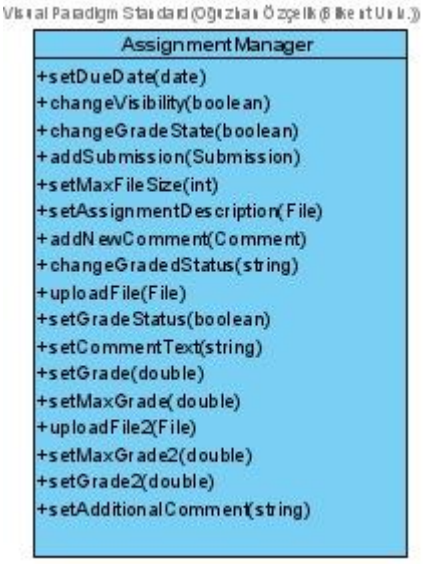
Every project has a description/information and this method changes this information in the database.

- addOutgoingMergeRequest(MergeRequest)
  When a group decides to merge with another group, this method makes a request object and updates the database.
- addIncomingMergeRequest(MergeRequest)
  When another group wants to merge with this group, this method is invoked.
- answerOfStudentsMergeRequest(Student,MergeRequest,string)
  When a student decides whether he/she wants the merge or not, this method is called.
- answerOfStudentsJoinRequest(Student,JoinRequest,string)
  When a student decides whether he/she wants the other student to join or not, this method is called.
- removeStudentFromGroup(Student)
  This method removes a student from the group and updates the database.
- newAccepted()
  When a join request is accepted, this information is updated in the JoinRequest object.
- newRejected()
  When a join request is rejected, this information is updated in the JoinRequest object.
- cancelAccepted()
  When a join request accepted status is cancelled, this information is updated in the JoinRequest object.
- cancelRejected()
  When a join request rejected status is cancelled, this information is updated in the JoinRequest object.
- newAcceptedInReceiver()
  When a member from the group that has received the merge request accepts it, this information is added to the votes.
- newRejectedInReceiver()
  When a member from the group that has received the merge request rejects it, this information is added to the votes.
- newAcceptedInSender()

When a member from the group that has sent the merge request accepts it, this information is added to the votes.

- newRejectedInSender()
  When a member from the group that has received the merge request rejects it, this information is added to the votes.
- cancelAcceptedInReceiver()
  When a member from the group that has received the merge request cancels his/her acceptance, this information is added to the votes.
- cancelRejectedInReceiver()
  When a member from the group that has received the merge request cancels his/her rejection, this information is added to the votes.
- cancelAcceptedInSender()
  When a member from the group that has sent the merge request cancels his/her acceptance, this information is added to the votes.
- cancelRejectedInSender()
  When a member from the group that has received the merge request cancels his/her rejection, this information is added to the votes.

4.3.4 AssignmentManager Class



- setDueDate(date)

This method is for setting the due date for this assignment.

- changeVisibility(boolean)

  This method is for changing the visibility of an assignment by the students.

- changeGradeState(boolean)

  This method is for changing the assignment's status in terms of being gradable.

- addSubmission(Submission)

  This method is for submitting the assignment.

- setMaxFileSize(int)

  This method is for setting a maximum file size for the assignment.

- setAssignmentDescription(File)

  This method is called when the instructor or the TA changes the assignment description.

- addNewComment(Comment)

  This method is called when a comment is added to the assignment.

- changeGradedStatus(string)

  This method is called when an assignment gets graded by the instructor or the TA or its grade is deleted.

- uploadFile(File)

  This method is called when a student uploads a file in order to make a submission for an assignment.

- setGradeStatus(string)

  This is for changing whether the submission is a graded submission or not.

- setCommentText(string)

  This method is called when a user sets a comment as text for a submission.

- setGrade(double)

  This method is called when a user wants to enter a grade to a submission.

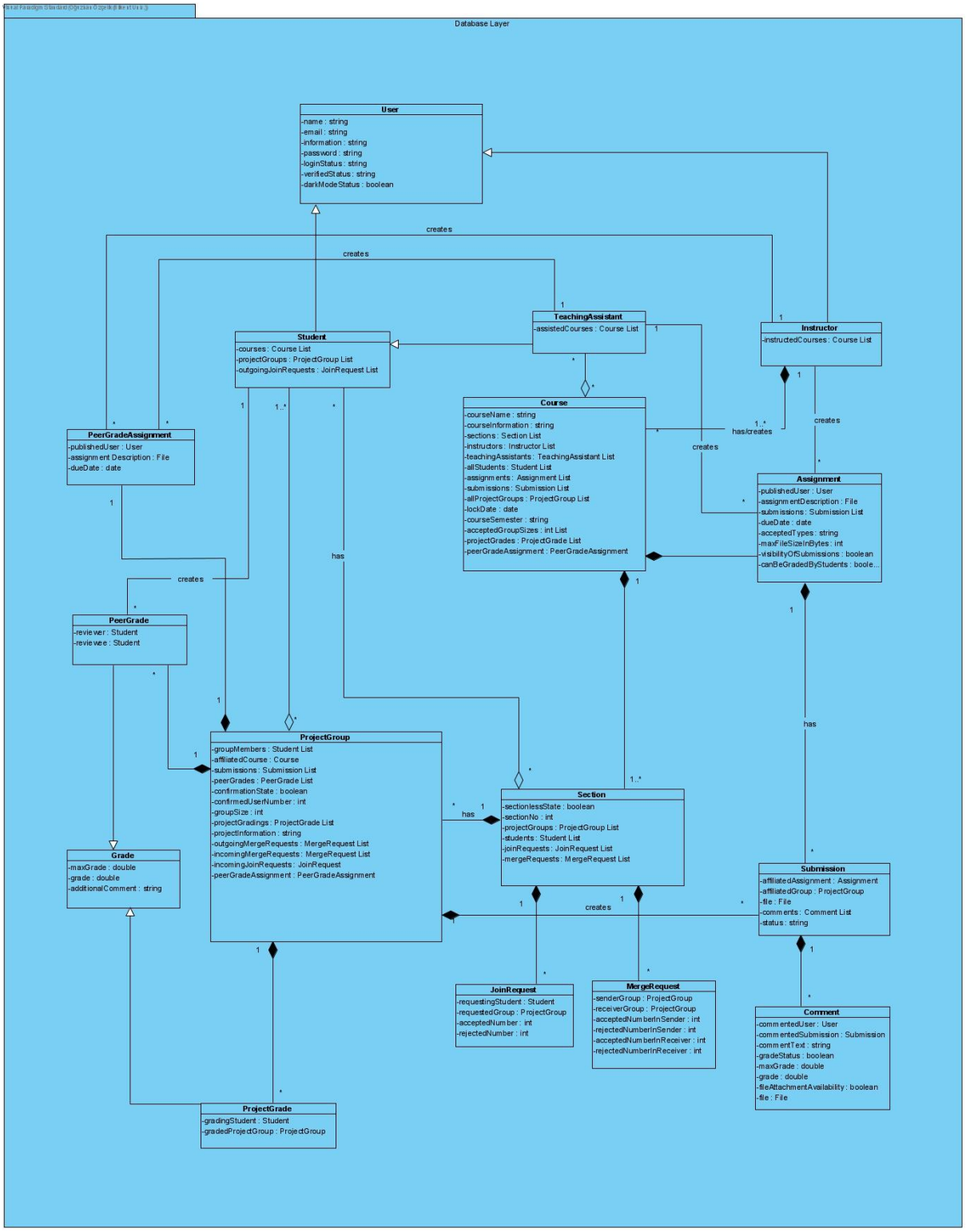
- setMaxGrade(double)

  This method is for setting a maximum grade for the assignment.

- setAdditionalComment(string)

  This method is called when the user sets additional comments for assignment as feedback.

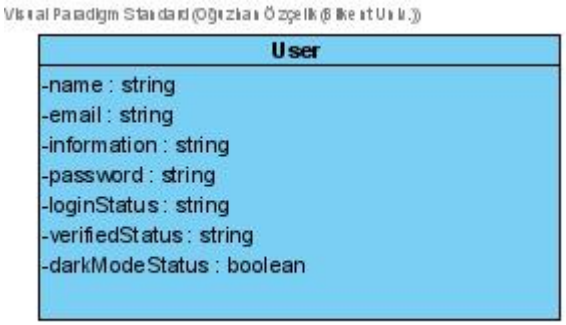
4.4 Database Layer



This layer is responsible for the database. All these classes act as a subsystem where the objects interact with each other. Each class has its attributes and these are necessary attributes which will be an important factor in the structure of SQL

Database tables. Whenever a change occurs, the Business Layer interacts with the Database Layer and makes the necessary changes in the database objects as well as database tables.
To make implementation easier, User class is being used as a parent class for Student, TeachingAssistant and Instructor class. The attributes of these classes are explained below:

4.4.1 User Class



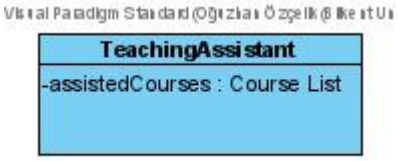
- string name: Name of the user
- string email: Email of the user
- string information: Information belonging to the user
- string password: Password of the user
- string loginStatus: Login status of the user.
- string verifiedStatus: Status of the user account being verified by the system or not
- boolean darkModeStatus: True, if the user uses dark mode; False, if the user uses regular mode.

4.4.2 Student Class

- List<Course> courses: List of the courses that the student is enrolled in
- List<ProjectGroup> projectGroups: List of the project groups that the student is in

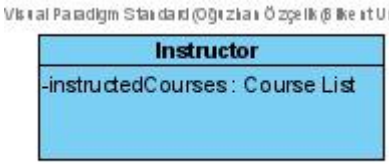
- List<JoinRequest> outgoingJoinRequests: List of the join requests that the student has sent

4.4.3 TeachingAssistant Class



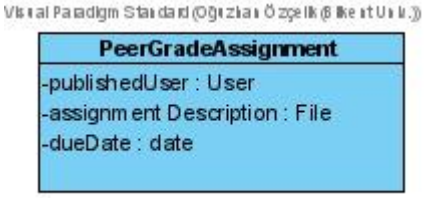
- List<Course> assistedCourses: List of the courses that the TA is assisting

4.4.4 Instructor Class



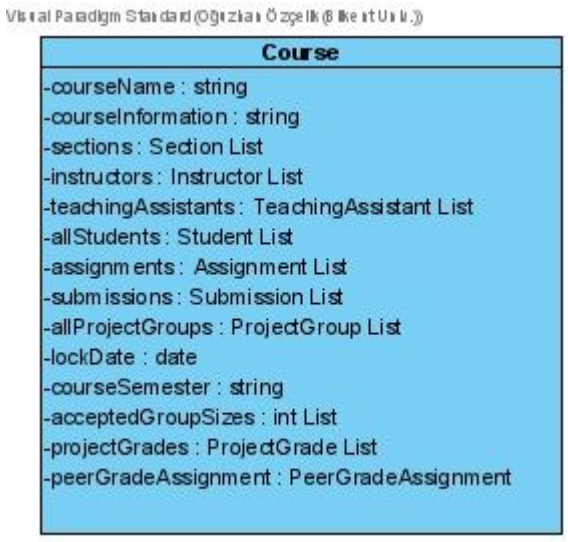
- List<Course> instructedCourses: List of the courses that instructor is instructed

4.4.5 PeerGradeAssignment Class



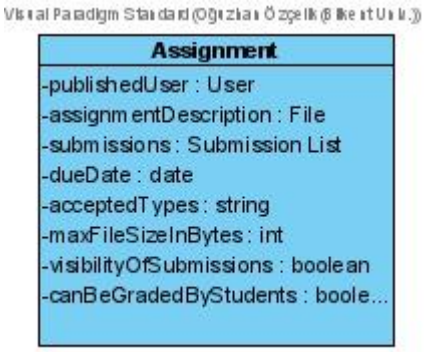
- User publishedUser: The user who has published the assignment
- File assignmentDescription: The file containing assignment description
- date dueDate: Due date of the assignment

4.4.6 Course Class



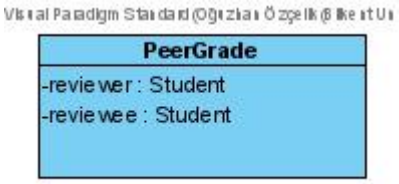
- string courseName: Name of the course
- string courseInformation: Information about the course
- List<Section> sections: List of the sections in the course
- List<Instructor> instructors: Instructors that are teaching the course
- List<TeachingAssistans> teachingAssistants: List of the TA's that are assisting the course
- List<Student> allStudents: List of the students enrolled in the course
- List<Assignment> assignments: List of the assignments in the course
- List<Submission> submissions: List of the submissions that are submitted in the course
- List<ProjectGroup> allProjectGroups: List of the project groups in the course
- date lockDate: The date that the groups for the course will be locked and cannot be changed anymore
- string courseSemester: The semester in which the course is taught
- List<int> acceptedGroupSizes: The list of the accepted group sizes
- List<ProjectGrade> projectGrades: List of the project grades
- PeerGradeAssignment peerGradeAssignment: Peer grade assignment of the course

4.4.7 Assignment Class



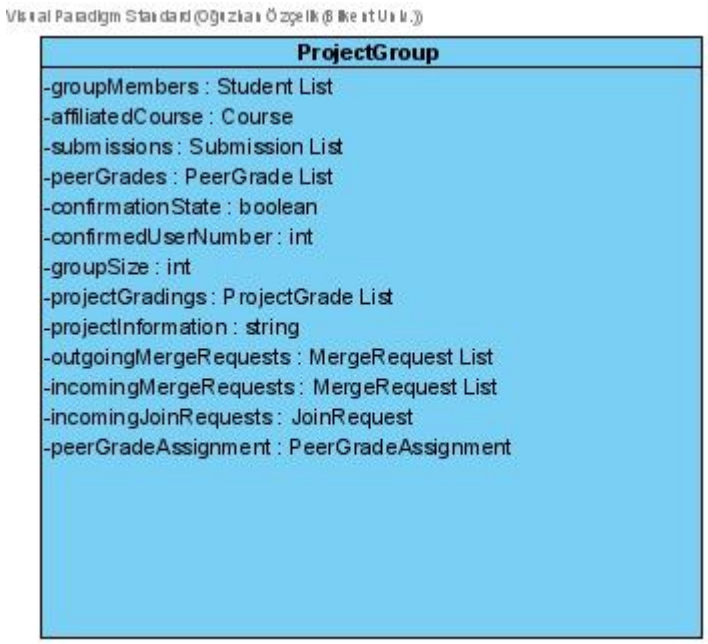
- User publishedUser: TA or Instructor that published the assignment
- File assignmentDescription: .txt file which has the description of assignment
- List<Submission> submissions: submissions of this assignment
- Date dueDate: Due date of the assignment
- string acceptedTypes: accepted file types of this assignment (might be word document or pdf document etc.)
- int maxFileSizeinBytes: Maximum file size for assignment represented as bytes
- boolean visibilityOfSubmissions: Shows visibility of submission
- boolean canBeGradedByStudents: Shows the grading of assignment by students

4.4.8 PeerGrade Class



- Student reviewer: Student who graded
- Student reviewee: Student who was reviewed

4.4.9 ProjectGroup Class



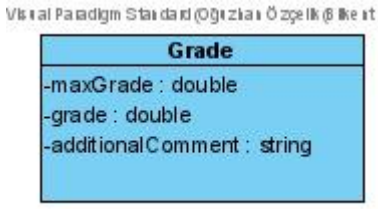
- List<Student> groupMembers: List of all members in project group
- Course affiliatedCourse: Course which the group is formed in
- List<Submission> submissions: List of submissions that are submitted in the lifetime of project group
- List<PeerGrade> peerGrades: List of peer grades that are submitted in the lifetime of project group
- boolean confirmationState: Indicates whether the group is confirmed/finalized or not.
- int confirmedUserNumber: number of users that confirmed this group.
- int groupSize: Size of the group
- List<ProjectGrade> projectGradings: All the grades that are given by other users
- string projectInformation: Information of project submitted by students in that project
- List<MergeRequest> outgoingMergeRequests: All the merge request sent by the members of this group
- List<MergeRequest> incomingMergeRequests: All the merge requests sent to this group
- List<JoinRequest> incomingJoinRequests: All the join requests sent to this group

- PeerGradeAssignment peerGradeAssignment: Peer grade assignment posted to this group by TA or instructor

4.4.10 Section Class

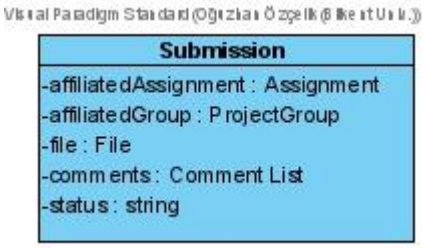
- boolean sectionlessState: True, if the course of the section is sectionless; False, if the course of the section is not sectionless
- int sectionNo: Number of the section
- List<ProjectGroup> projectGroups: List of the project groups in the section
- List<Students> students: Students that are in the section
- List<JoinRequest> joinRequests: List of the join requests that are created within the section
- List<MergeRequest> mergeRequests: List of the group merge requests that are created within the section

4.4.11 Grade Class



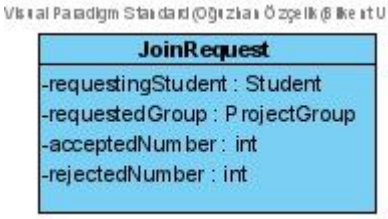
- double maxGrade: Maximum possible grade
- double grade: Numeric grade that is given
- string additionalComment: Additional comments about the grade

4.4.12 Submission Class



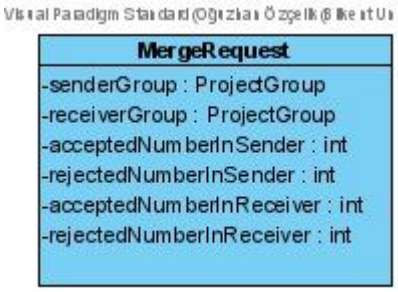
- Assignment affiliatedAssignment: Assignment that the submission is affiliated with
- ProjectGroup affiliatedGroup: The project group that the submission belongs to
- File file: Submitted file
- List<Comment> comments: List of the comments that are made as feedback for the submission
- string status: Status of the assignment in terms of being submitted or not

4.4.13 JoinRequest Class



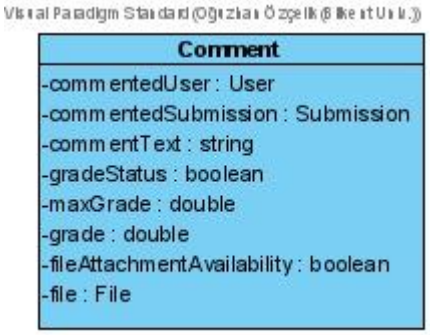
- Student requestingStudent: Student who sent the join request
- ProjectGroup requestedGroup: Project group which student sent join request to
- int acceptedNumber: Number of students accepted the join request
- int rejectedNumber: Number of students rejected the join request

4.4.14 MergeRequest Class



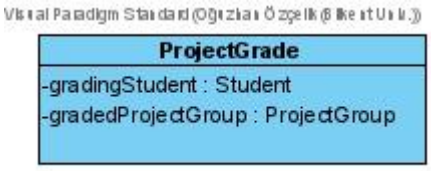
- ProjectGroup senderGroup: Project group which sent the merge request
- ProjectGroup receiverGroup: Project group which received the merge request
- int acceptedNumberInSender: Number of students who accepted the request in sender group
- int rejectedNumberInSender: Number of students who rejected the request in sender group
- int acceptedNumberInReceiver: Number of students who accepted the request in receiver group
- int rejectedNumberInReceiver: Number of students who rejected the request in receiver group

4.4.15 Comment Class



- User commentedUser: User that the comment is being made for
- Submission commentedSubmission: Submission that is affiliated with the comment
- string commentText: Text of the comment
- boolean gradeStatus: True, if the comment contains a grade; False, if not
- double maxGrade: Maximum possible grade
- double grade: Numerical grade that is given
- boolean fileAttachmentAvailability: Information about the availability of the file attachment. True, if attaching the file is possible; False, if not
- File file: File of the comment

4.4.16 ProjectGrade Class



- Student gradingStudent: Student who is grading the project
- ProjectGroup gradedProjectGroup: Project group which is graded by student

5. Glossary & References

Glossary

REST: Representational State Transfer

Application Programming Interface

TA: Teaching Assistant

Pseudogroup: Group formation before finalizing groups. There isn't any restriction on forming pseudogroups.

Sectionless: When instructor wants to open a course in which group formation between sections are allowed, instructor opens sectionless course.

References

Object-Oriented Software Engineering, Using UML, Patterns, and Java, Third Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2010, ISBN: 0-13-606125-7.

Visual Paradigm [Computer Software]. (Version 16.2.).