Bilkent University

Department of Computer Engineering

CS319 Term Project

ReviewTool

Group 1J - Design Report

Göktuğ Çağıran, Cihan Can Kılıç, Mustafa Anıl Taşdan

Instructor: Eray Tüzün
Teaching Assistant(s): Elgun Jabrayilzade, Erdem Tuna

Progress/Final Report
May 3, 2010

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

1. Introduction 3
1.1 Purpose Of The System 3
1.2 Design Goals 3
1.3 Criteria For System Design 3
1.3.1 End User Criteria 3
1.3.1.1 Ease Of Use 3
1.3.1.2 Security 4
1.3.2 Maintenance Criteria 4
1.3.2.1 Corrective Maintenance 4
1.3.2.2 Adaptive Maintenance 4
1.3.3 Performance Criteria 4
1.3.3.1 Speed 4
1.3.3.2 Multi-Threading 4
1.4 Definitions 4
2. System Architecture 6
2.1 Subsystem Decomposition 6
2.2 Hardware/Software Mapping 7
2.3 Persistent Data Management 8
2.4 Access Control and Security 8
2.5 Global Software Controls 9
2.6 Boundary Conditions 9
2.6.1 Initialization 9
2.6.2 Termination 9
3. Subsystems 10
3.1 UserInterface Subsystem 10
3.2 Business Logic 13
3.3 Database 14
4. Low Level Design 14
4.1 Trade-offs 14
4.1.1 Space vs Speed: 14
4.1.2 Development Time vs Performance: 15
4.1.3 Functionality vs Usability: 15
4.1.4 Memory vs Maintainability: 15
4.2 Final Object Design 16
4.2.1 Class Diagram of Interface Package 17
4.2.2 Class Diagram of Business Logic 30
4.2.3 Class Diagram of Entities 43
4.3 External Packages 49

Design Report

ReviewTool

1.Introduction

1.1 Purpose Of The System

In Object Oriented Software Engineering, review of a product has a vital importance, with many types of reviews done by many types of reviewers during the life cycle of a project. These reviews can take the form of verbal reviews done over reports or documentations, or application supported reviews done over computers. GitHub for example, the most popular code repository service, has code reviewing tools in many parts of the website, with the aim of improving code and project quality.

ReviewTool is a desktop application where students and instructors can communicate, have their artifacts reviewed and peer review their group members. The purpose of the application is the provide a platform where students can upload their assignments provided by the instructors and get reviews from them. It aims to provide a singular platform for project management, review and communication.

1.2 Design Goals

The system follows client-server architecture with 3 tier architectural style. The criteria for the system design are explained below.

1.3 Criteria For System Design

1.3.1 End User Criteria

1.3.1.1 Ease Of Use

The system is designed with a user-friendly interface which requires little to no prior training for the use of the system.

1.3.1.2 Security

The system holds sensitive data which should not be shared with public without the knowledge of the users. This data includes names, emails, passwords, student id’s, artifacts and peer reviews.

1.3.2 Maintenance Criteria

1.3.2.1 Corrective Maintenance

Since bugs and errors may occur. The code should be well commented and documented, so that a programmer may easily understand the code and find solutions to the bugs and errors.

1.3.2.2 Adaptive Maintenance

Environment of the program and/or development kits and libraries used during the development may change or become deprecated. The design of the system and external libraries used during the development should be well documented and understandable so that deprecated or changed functionalities can be upgraded and fixed.

1.3.3 Performance Criteria

1.3.3.1 Speed

The system should work with minimal response times for all operations included in the system. Maximum response time is aimed to be 0.5 seconds.

1.3.3.2 Multi-Threading

Since multiple users may perform operations on the system simultaneously, the system should be able to handle multiple requests at the same time.

1.4 Definitions

Java: The programming language which the system will be written with.

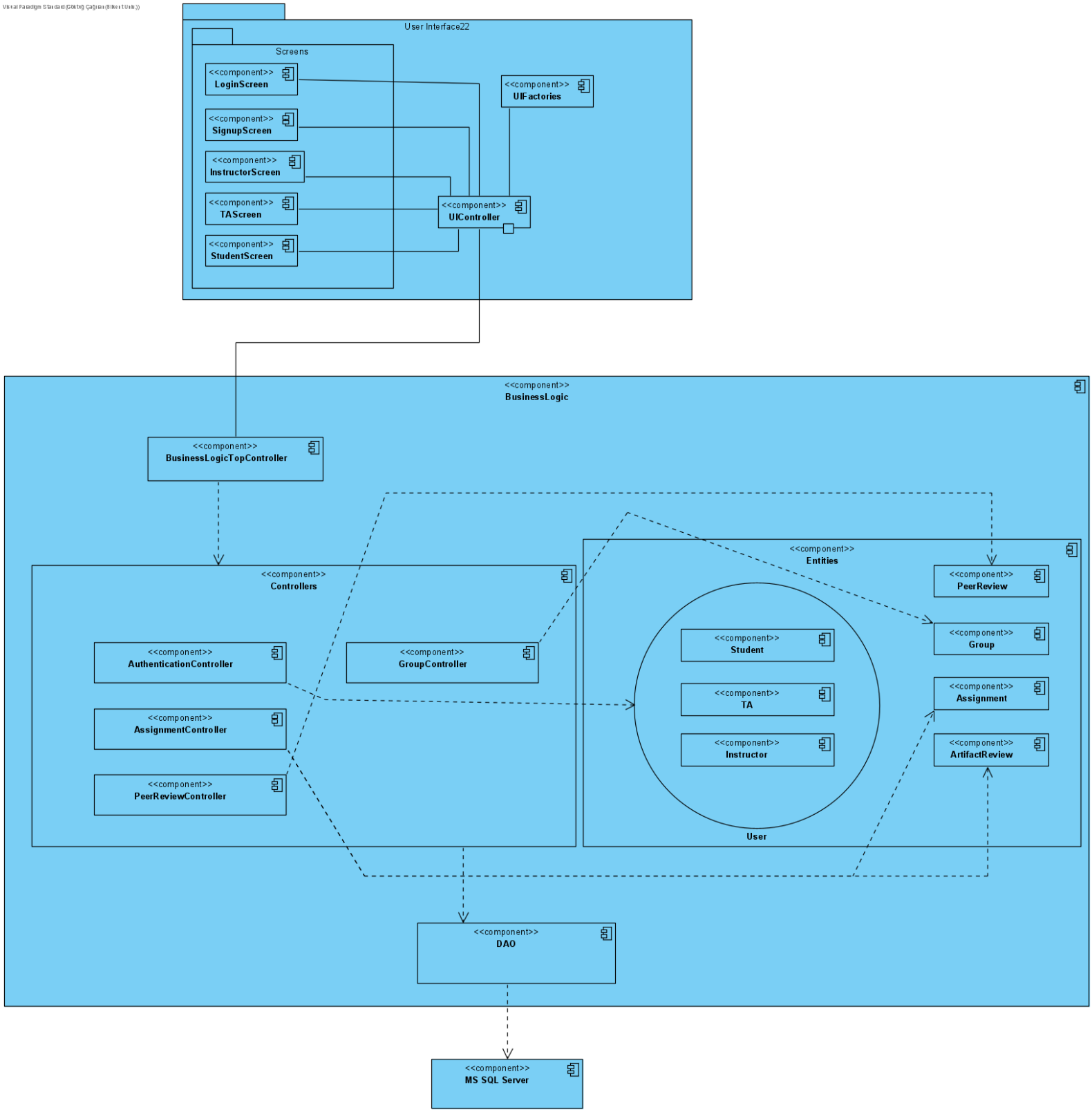
Microsoft SQL Server: SQL server service used for the database.

T-SQL: SQL language which MS SQL Server uses.

Java Runtime Environment(JRE): Java program needed to run programs written with Java language.

2.System Architecture

2.1 Subsystem Decomposition



Our system is decomposed into three subsystems to make the design easier to update and worked on. Adding new features or fixing bugs is easier when the system consists of different systems that are independent from each other. Also, by decomposing the project organization becomes a lot easier. Every system has to be worked on separately as they are independent.

Interface: First layer of our decomposition is “Interface”. In this layer system deals with mostly use. All the interaction with user is handled by Interface. It also sets boundaries for the user as forming their control panel. By taking Interface as a separate component of the project we make interface changes very easy and quick. Interface consists of LoginScreen, SignupScreen, InstructorScreen, TAScreen, StudentScreen and a UIController which controls and handles all other screens.

Business Logic: Second layer of the decomposition is Business Logic. In this layer system handles inputs of the user navigated by Interface and the data coming from database navigated by MS SQL Server. Bussiness Logic collects all this data and processes it as we need. Components that form Bussiness Logic starts with BussinessLogicTopController which communicates with Interface by sending and receiving data. Second component is Controllers. Inside Controllers we have different controllers for every class. These controllers communicate with entities and process the received data and send data to DAO which is the layer that communicates with MS SQL Server.

MS SQL Server: Third layer is the MS SQL Server. This is the layer which has the database. The database is a relational database. To communicate with the database MS SQL Server is used.

2.2 Hardware/Software Mapping

The application is a Java desktop application, capable of running

on any machine that has the Java Runtime Environment. In particular, the device must have the ability to run Java 15. It, connects to a remote database.

2.3 Persistent Data Management

The system needs to store the following types of data:

- Credentials of users: Name, e-mail, id, password
- User Information: Group name, assigned members, assigned assignments, uploaded artifacts, artifact and peer reviews.
- Artifact Data: A PDF file uploaded to a specific assignment by a group.

Even though these data are not heavy for performance purposes the data will be downloaded when needed. When a change is made by the user through the UI, the Business Logic subsystem will send the necessary queries to the MS SQL Server and update the database. When a change has been made to the database users will need to refresh the page to retrieve new data.

2.4 Access Control and Security

There are three types of accounts: Instructor, TA and Student. An Instructor account will have the most authority, will be able to create and manage groups and assignments. The TA account will only keep the ability to view all groups, compared to the Instructor. The Student account will be able to only see his/her own group and will not be able to see peer reviews made by students. All account types will be able to leave artifact reviews, while only a student account can leave peer reviews. Also only a student account can upload artifacts.

2.5 Global Software Controls

The software will utilize event-driven control. When an event is detected through the UI, the event data will be passed from the User Interface subsystem to the Logic Management subsystem. The Logic Management subsystem will execute the necessary operations, like updating the UI or requesting data from the Database Management, based on the event.

The system will follow a centralized design, for two reasons: it is easier to develop, which fits into our “low-cost” design goal, and a possible performance bottleneck is not a concern for an application of this complexity.

2.6 Boundary Conditions

2.6.1 Initialization

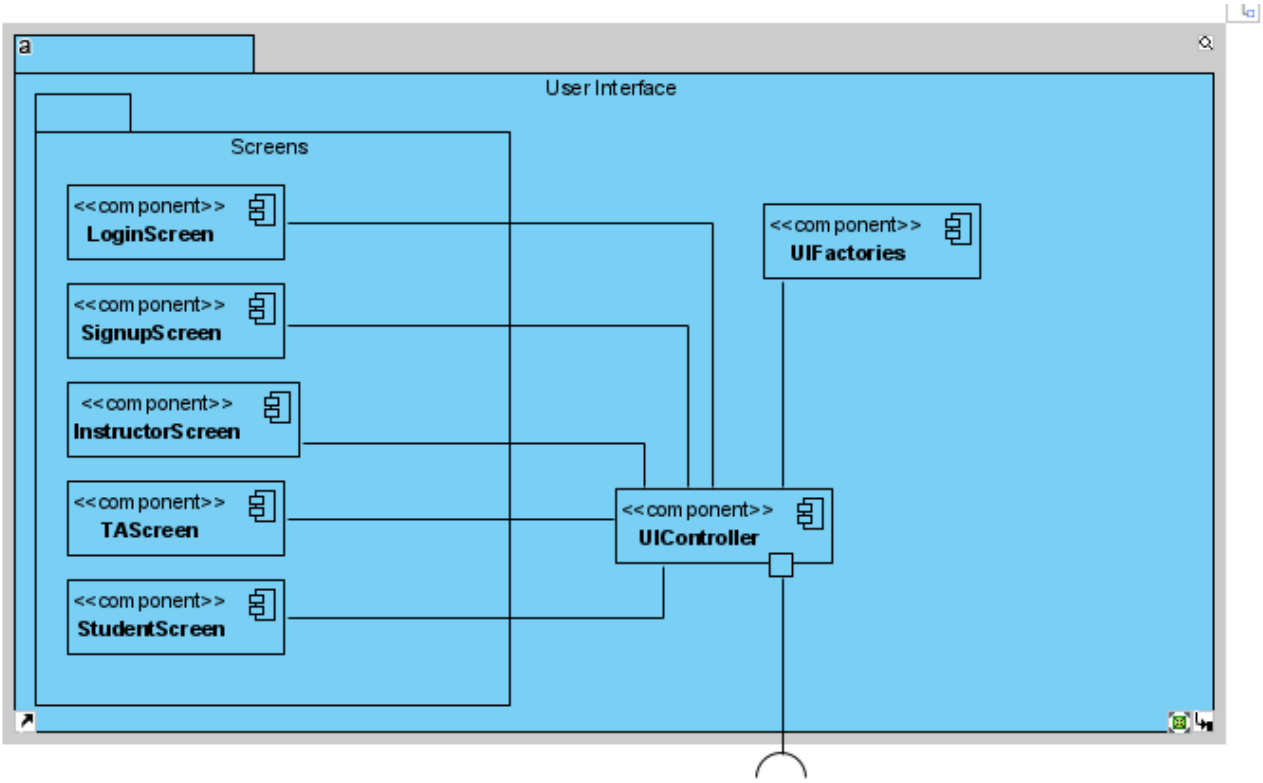
Upon startup, there will be no user data stored in the user’s computer, there will be only application code. After login, the general user and group data will be downloaded to the system. The whole system starts up at the same time.

2.6.2 Termination

The subsystems’ existence are essential for the whole program and they are not individually allowed to terminate. Upon a graceful exit by the user, the downloaded data will be erased from the system. As the updates to the database are communicated at the instant the change is made by the user, no additional transfer is necessary upon termination.

3.Subsystems

3.1 UserInterface Subsystem



The UserInterface subsystem is responsible for creation, alteration, removal and display of UI elements. It is the only subsystem an outside user can interact with, which simplifies control and enhances application security. It is responsible for getting input from the user, passing the input to the BusinessLogic subsystem and displaying data received from the BusinessLogic subsystem. The subsystem has 3 main parts:

1. Screens
2. UIController
3. UIFactories

The Screens component, which comprises of 5 classes, contains the UI objects that are shown to the user. The 5 classes are:

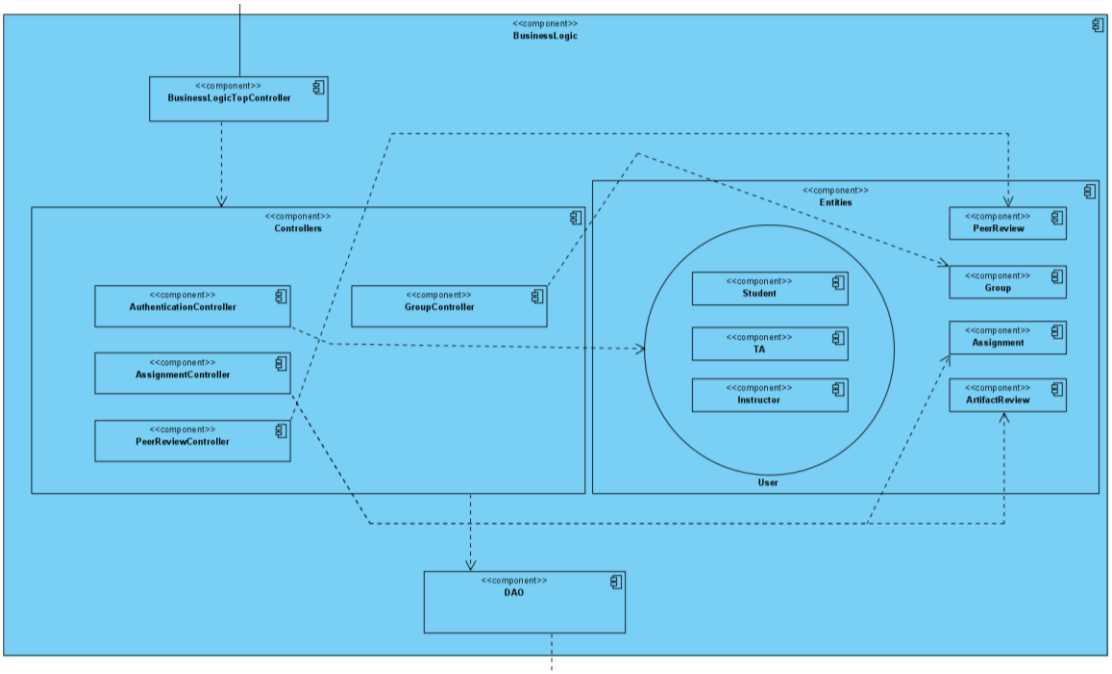
1. LoginScreen
2. SignupScreen
3. InstructorScreen
4. TAScreen
5. StudentScreen

Upon startup of the system, the LoginScreen class will be visible and the other screens will be shown according to the actions of the user and data received from the database. The active screen will be controlled by the UIController component.

The UIController component is both the heart of the subsystem, and is a Facade component that provides an interface for other subsystems trying to access the UserInterface subsystem. All data that a Screen needs will be requested from BusinessLogic through the UIController, and all data received will pass through the UIController. The UI controller contains all functions that is present in a Screen. The UIController is also responsible for: Switching Screens, Refreshing UI upon a change in BusinessLogic and creating and controlling UIFactories.

The UIFactories component is designed with the aim of applying Factory Method pattern to complex UI object creation. The application consists of many buttons or panels that follow the same pattern. For example, an AddMember button consists of a Label, an Icon and an EventHandler. The same is true for a LoginButton, a RemoveButton and else. Similarly, a GroupPanel consists of a groupName Label, various buttons for member addition, removal, alteration and so on. An AssignmentPanel has the same buttons. The UIFactories component aims to simplify object creation by encapsulating it with classes and providing polymorphism so that the panels and buttons can be easily created at runtime, depending on data received from BusinessLogic and the database.

3.2 Business Logic



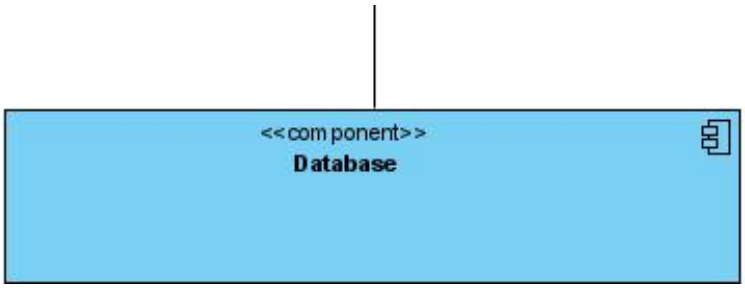
BusinessLogic handles all processes on data received and data that will be sent. It also controls all classes and their attributes with some boundaries. So mostly operational logics of the system is done here. Controllers part has four components.

1. AuthenticationController: Controls login, signup, user operations. It provides validation of users and accounts by communicating to database.
2. AssignmentController: Controls activities related to assignments. Collects assignment data and processes them onto classes to make possible some functionalities.
3. PeerReviewController: Controls activities about peer reviewing system. Collects and processes peer reviews and control entities.
4. GroupController: Controls activities about groups.

These controllers are controlled by BusinessLogicTopController. Facade design pattern is used here.

Then there is the entitiy part which consists of PeerReview, Group, Assignment and ArtifactReview. All of these are models for separate entities which are controlled by Controllers. Another component in Entities is User which holds common attributes of all user types. Under User there are independent user types which are Student, TA, Instructor. These User types model different users of the platform. Final Part of BusinessLogic is DAO which handles the data flow between BusinessLogic and MS SQL Server. It navigates processed data from Controllers and at the same time gets data from MS SQL Server and navigates it to Controllers.

3.3 Database



MS SQL Server is used for database. The database is stored in a remote location. Java JDBC is used to communicate with MS SQL Server. Queries are made with T-SQL.

4.Low Level Design

4.1 Trade-offs

4.1.1 Space vs Speed:

Speed vs space has always been very crucial in programming. This is a web-based application that is constantly communicating to a database. However, memory needs of the application aren't very large. So, speed is more important for this application therefore speed is definitely a trade-off with space in this implementation.

4.1.2 Development Time vs Performance:

Development time is more important for us as we are on a deadline so we are mostly trying to use things that we are familiar things as learning a something new takes more time. Also, as we are making a simple application, performance changes won't affect the over-all system too much. So, in this example development time is a trade-off with performance.

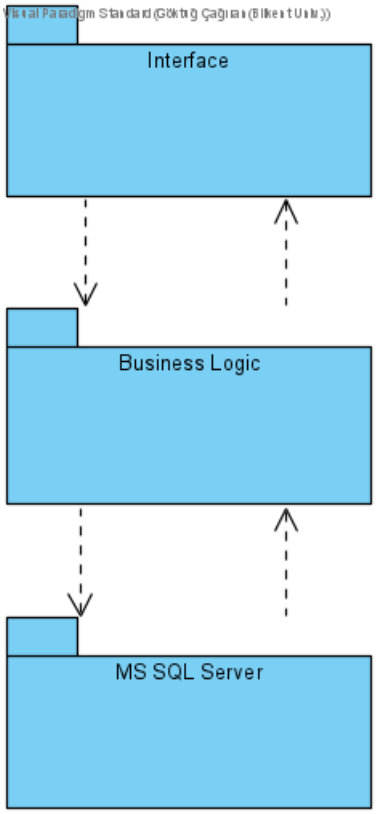
4.1.3 Functionality vs Usability:

The user profiles of this application are instructors, students and TAs so there will be lots of people using it. Therefore, it should be easy and fast to use the application. As long as all of the main functionalities of the system work well, usability is a trade-off with functionality. Satisfying main functionalities with an easy-to-use interface is our goal.

4.1.4 Memory vs Maintainability:

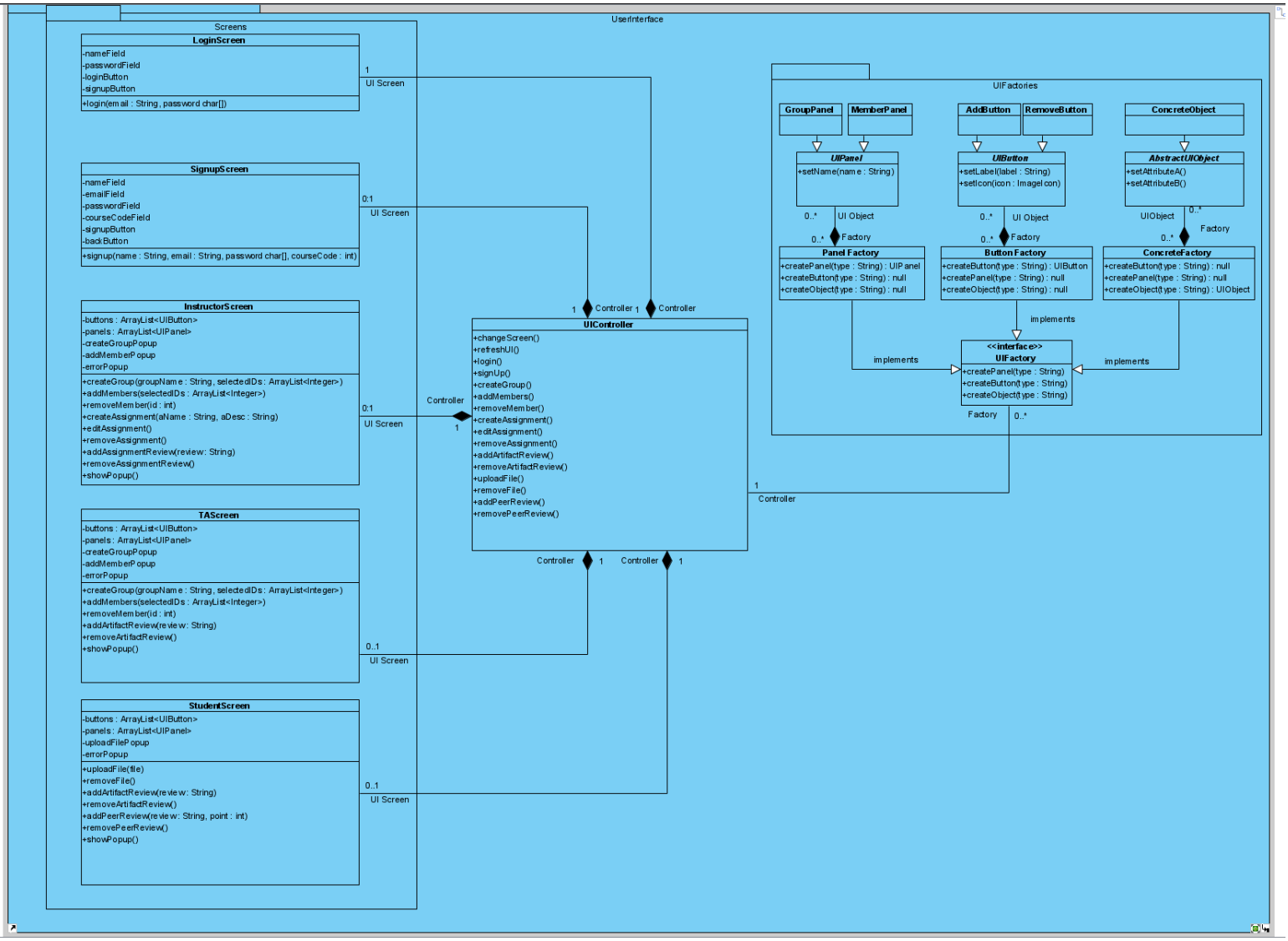
We are trying to use as little memory as possible, but there are some classes with common attributes and sometimes we have to store same data in different classes. However, this is happening because of maintainability reasons. The hierarchy built is very optimal to changes so for this application maintainability is a trade-off with memory.

4.2 Final Object Design



There are 3 main subsystems in the program. The first subsystem is Interface subsystem. This package includes interfaces and classes related to UI components of the program. Through this subsystem user can interact with the program. Interface subsystem can only communicate with Business Logic subsystem. Second subsystem is the Business Logic subsystem. This subsystem includes interfaces and classes related to the back-end of the program. Controllers, entities and database access object are parts of this subsystem. This package can communicate with Interface subsystem and MS SQL Server.
Third subsystem is the MS SQL Server. This is a service provided by Microsoft. It enables to communicate with relational databases.

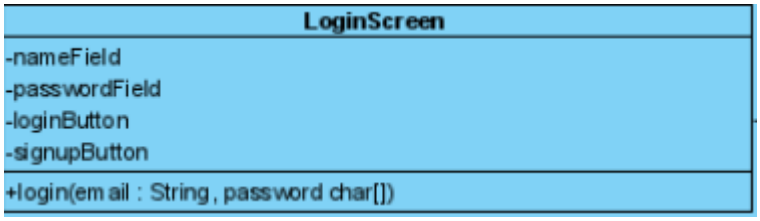
4.2.1 Class Diagram of Interface Package



- **UserInterface Subsystem Classes**

a) Screens Package

i. LoginScreen Class



Attributes:

JTextField nameField: Field for user to enter their E-Mail.

JPasswordField passwordField: Field for user to enter their password.

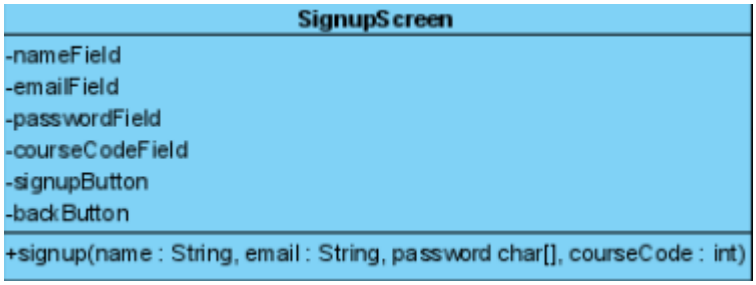
JButton loginButton: Button for user to submit credentials.

JButton signupButton: Button for user to switch to Signup Screen.

Operations:

public static boolean login(email: String,password: char[]): Sends entered info to UIController class.

ii. SignupScreen Class



Attributes:

JTextField nameField: Field for user to enter their name.

JTextField emailField: Field for user to enter their email.

JPasswordField passwordField: Field for user to enter their password.

JTextField courseCodeField: Field for user to enter their course code.

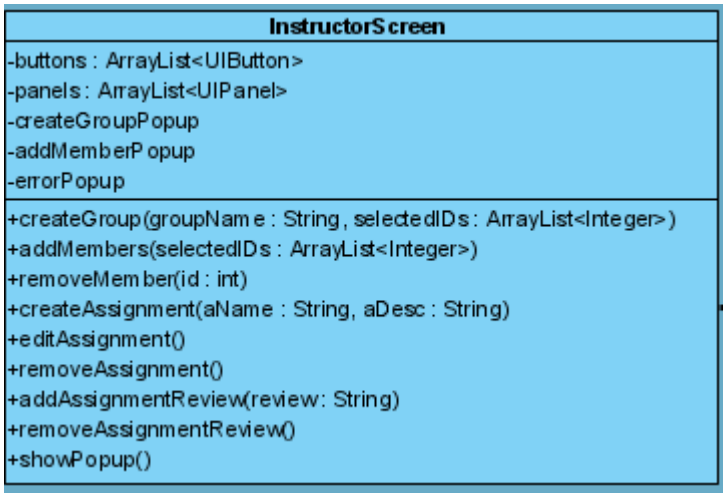
JButton signupButton: Button for user to submit form.

JButton backButton: Button for user to return to Login Screen.

Operations:

public static boolean signup(name:String,email:String,password:char[],courseCode:int): Sends entered info to UIController class.

iii. InstructorScreen Class



Attributes:

ArrayList<UIButton> buttons: The buttons created by ButtonFactory in the UIFactories package.

ArrayList<UIPanel> panels: The panels created by PanelFactory in the UIFactories package.

JDialog createGroupPopup: A popup that shows unassigned students and a field to enter a group name for.

JDialog addMemberPopup: A popup that shows unassigned students.

JDialog errorPopup: A popup that shows a customize-able error message.

Operations:

public void createGroup(String groupName,ArrayList<Integer> selectedIDs): Sends entered info to UIController class.

public void addMembers(ArrayList<Integer> selectedIDs): Sends entered info to UIController class.

public void removeMember(int id): Sends entered info to UIController class.

public void createAssignment(String aName,String aDesc): Sends entered info to UIController class.

public void editAssignment(String aName,String aDesc): Sends entered info to UIController class.

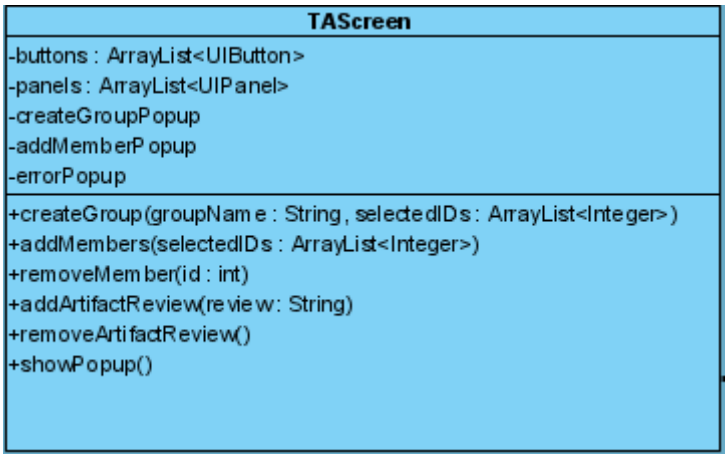
public void removeAssignment(String aName): Sends entered info to UIController class.

public void addAssignmentReview(String review): Sends entered info to UIController class.

public void removeAssignmentReview(String review): Sends entered info to UIController class.

public void showPopup(String type,String message): Shows the desired popup.

iv. TAScreen Class



Attributes:

ArrayList<UIButton> buttons: The buttons created by ButtonFactory in the UIFactories package.

ArrayList<UIPanel> panels: The panels created by PanelFactory in the UIFactories package.

JDialog createGroupPopup: A popup that shows unassigned students and a field to enter a group name for.

JDialog addMemberPopup: A popup that shows unassigned students.

JDialog errorPopup: A popup that shows a customize-able error message.

Operations:

public void createGroup(String groupName,ArrayList<Integer> selectedIDs): Sends entered

info to UIController class.

public void addMembers(ArrayList<Integer> selectedIDs): Sends entered info to UIController class.

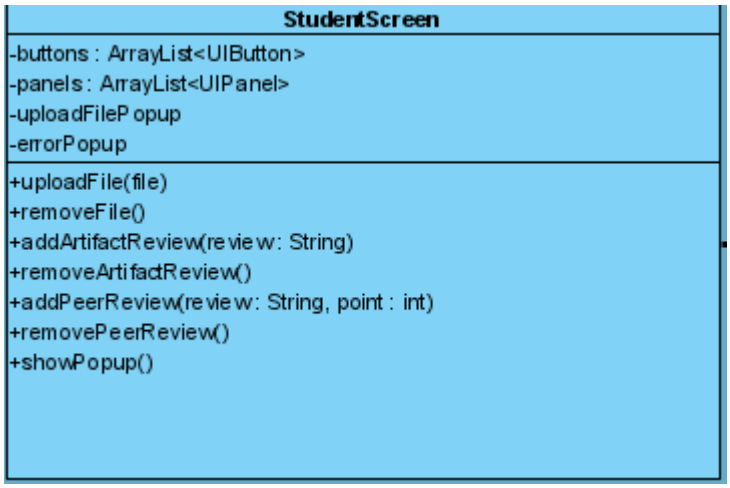
public void removeMember(int id): Sends entered info to UIController class.

public void addArtifactReview(String review): Sends entered info to UIController class.

public void removeArtifactReview(String review): Sends entered info to UIController class.

public void showPopup(String type,String message): Shows the desired popup.

v. StudentScreen Class



Attributes:

ArrayList<UIButton> buttons: The buttons created by ButtonFactory in the UIFactories package.

ArrayList<UIPanel> panels: The panels created by

PanelFactory in the UIFactories package.

JDialog errorPopup: A popup that shows a customize-able error message.

JDialog uploadFilePopup: A popup that shows a file chooser for artifact uploading.

Operations:

public void uploadFile(): Sends entered info to UIController class.

public void removeFile(): Sends entered info to UIController class.

public void addArtifactReview(String review): Sends entered info to UIController class.

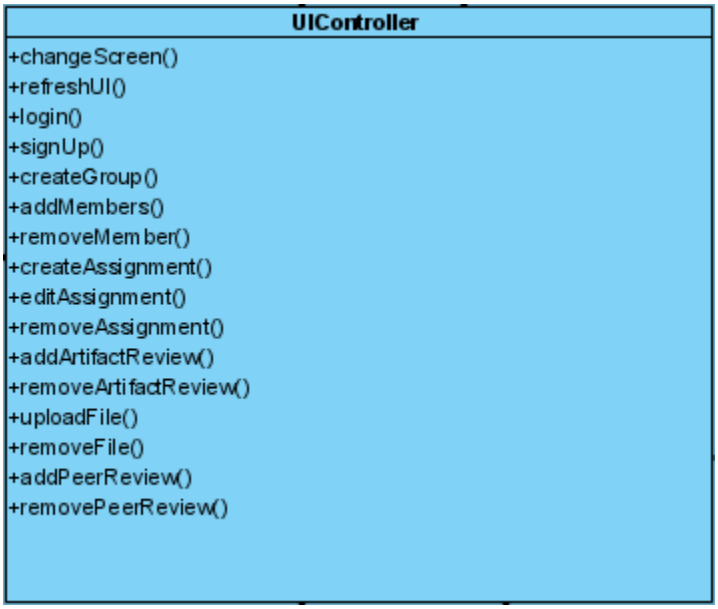
public void removeArtifactReview(String review): Sends entered info to UIController class.

public void addPeerReview(String review,int point): Sends entered info to UIController class.

public void removePeerReview(): Sends entered info to UIController class.

public void showPopup(String type,String message): Shows the desired popup.

vi. **UIController Class**



Operations:

public void changeScreen(String screen): Sets the desired screen visible for the User.

public void refreshUI(): Redraws the currently visible UI screen when there is a change in the BusinessManagement subsystem.

public boolean login(String email,char[] password): Sends the info to the BusinessManagement subsystem for User authentication.

public boolean signup(String name,String email,char[] password,int courseCode): Sends the info to the BusinessManagement subsystem for User creation.

public void createGroup(String groupName,ArrayList<Integer> selectedIDs): Sends the group name and selected ids' of students to the BusinessManagement subsystem for group creation.

public void addMembers(ArrayList<Integer> selectedIDs): Sends the selected ids' of students to the BusinessManagement subsystem for member addition.

public void removeMember(int id): Sends the id of a student to the BusinessManagement subsystem for member removal.

public void createAssignment(String aName,String aDesc): Sends the name and description of an assignment to the BusinessManagement subsystem for assignment creation.

public void editAssignment(String aName,String aDesc): Sends the name and description of an assignment to the BusinessManagement subsystem for assignment alteration.

public void removeAssignment(String aName): Sends the name of an assignment to the BusinessManagement subsystem for assignment removal.

public void uploadFile(): Sends the file data chosen by the User using the file chooser to the BusinessManagement subsystem for artifact submission.

public void removeFile(): Sends the chosen artifact data to the BusinessManagement subsystem for artifact removal.

public void addArtifactReview(String review): Sends the

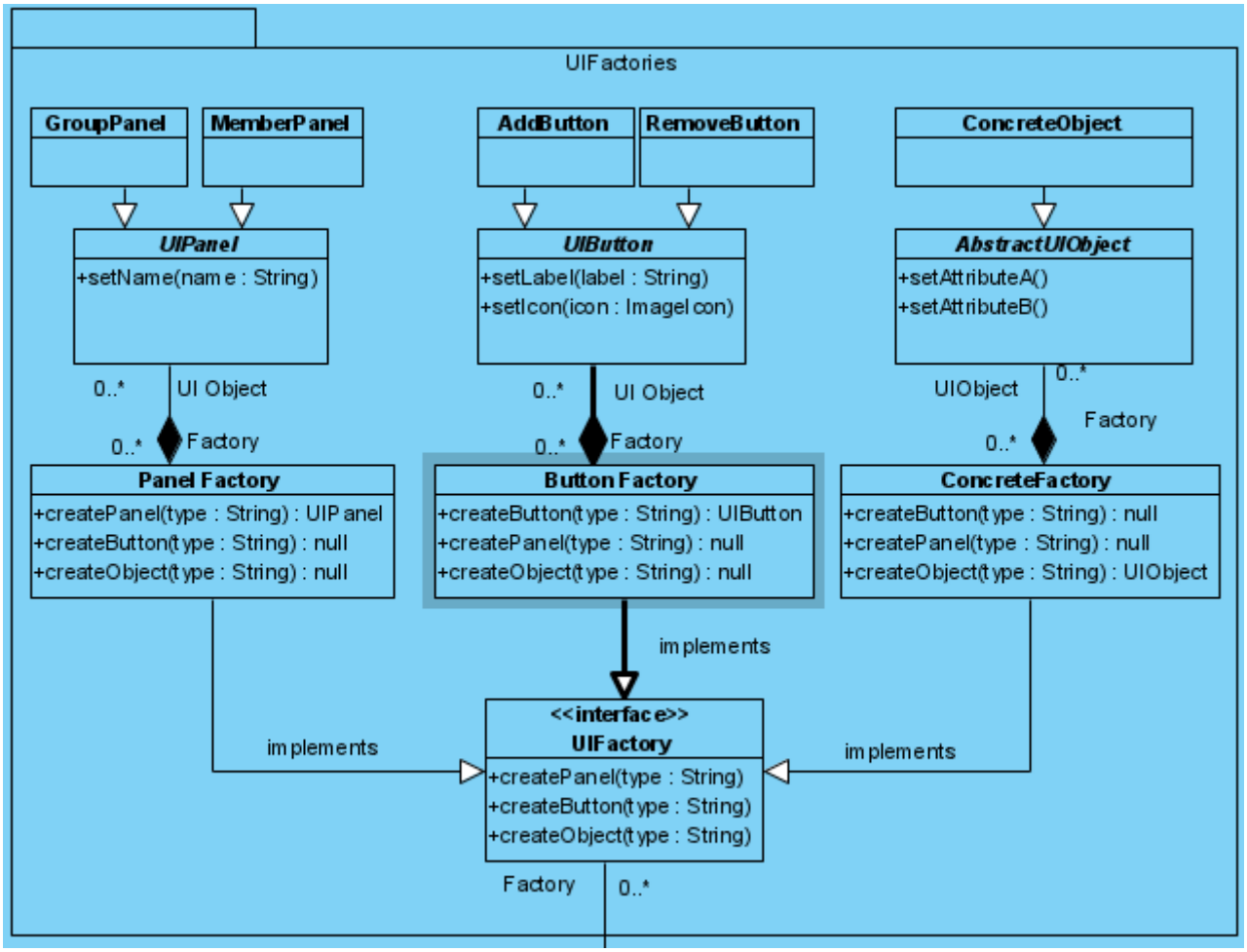
review made on an artifact to the BusinessManagement subsystem for addition to an artifact.

public void removeArtifactReview(String review): Sends the review made on an artifact to the BusinessManagement subsystem for removal from an artifact.

public void addPeerReview(String review,int point): Sends entered review and point to the BusinessManagement subsystem for addition to a Student object.

public void removePeerReview(String review): Sends entered review name to the BusinessManagement subsystem for removal from a Student object.

b)UIFactories Package



The structure of the UIFactories package is more suitable to document at once, rather than documenting attributes and operations of individual classes. Since a User Interface is ever changing with different object types being added, the UIFactories component will see new factory additions in each iteration, which is the reason we decided to implement a Factory Method pattern in the first place. The documentation will be done over an arbitrary UIObject object, which is only created for

documentation purposes, it will not be in implementation.

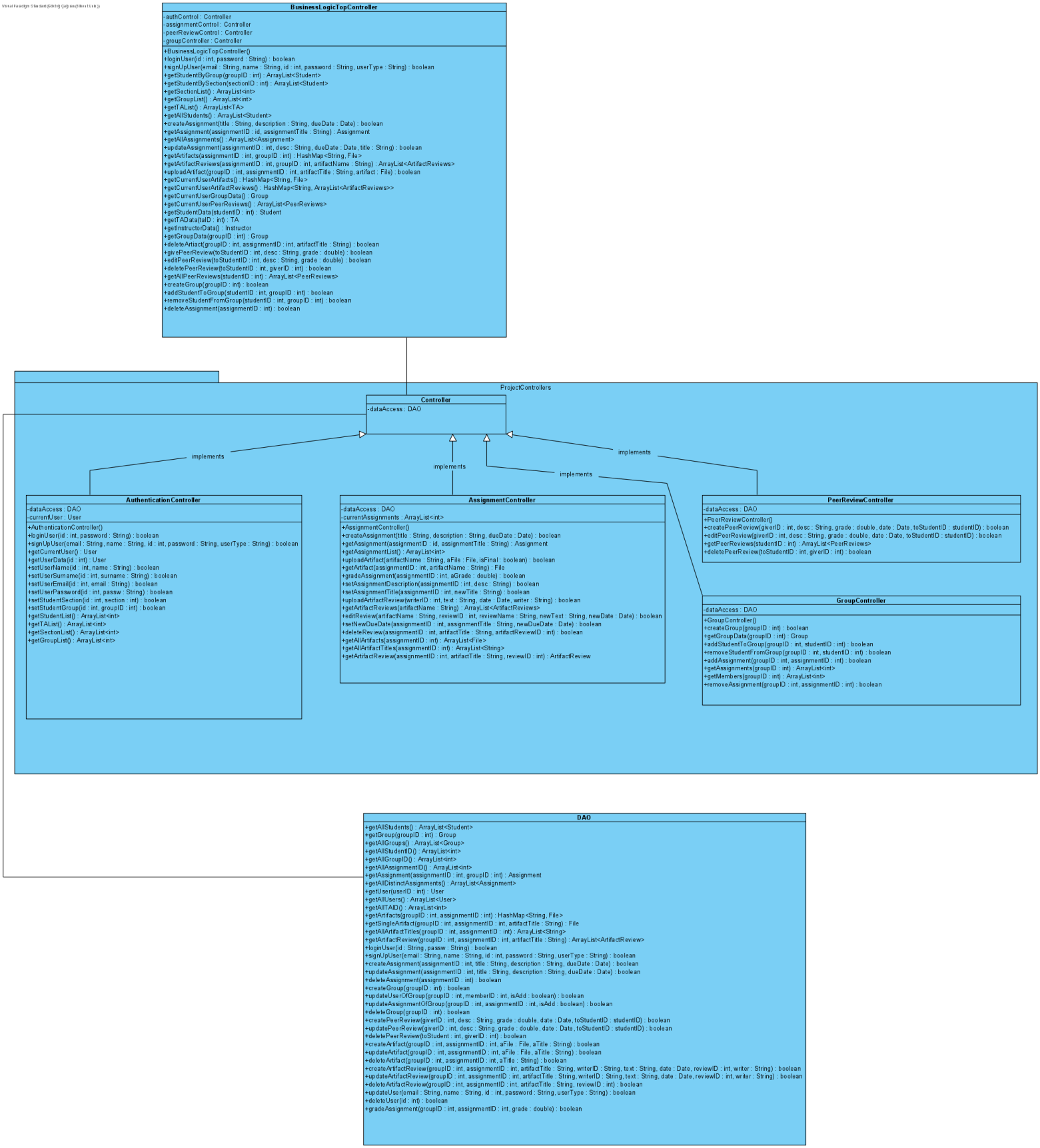
Interface UIFactory: This is a Java interface that factories have to implement. When a UIFactory is going to be instantiated in another class, this interface will provide polymorphism for runtime instantiation.

Class ConcreteFactory: This is the concrete factory that creates a UIObject. Since it implements the UIFactory interface, it has to implement creation methods found in other factories, but those implementations will be simply returning null.

abstract Class AbstractUIObject: This is an abstract class that concrete UIObjects have to extend. This class allows ConcreteFactory to return a single type of UIObject after creation, since all UIObjects will extend this class.

Class UIObject: These are the classes that have complex UI attributes and implementations such as Labels, Icons, Layouts, Margins etc. The ConcreteFactory classes create and return these objects.

4.2.2 Class Diagram of Business Logic



- **Controller Classes**

Name	AuthenticationController
Description	This class is responsible for login and sign up purposes and authenticates user about the thing it can do.
Properties	• dataAccess: DAO • currentUser: User
Methods	• loginUser(id: int, password:String): returns boolean, takes the input user entered as login credentials and checks them with current users. • signUpUser(email: String, name: String, id:int, password:String,userType:String): returns Boolean. Takes the input user entered as sign up information and creates an account. • getCurrentUser(): returns User class which is logged in at the moments. • getUserData(id: int): returns an instance of User class specified with the id number of the use. • setUserName(id: int, name:string) returns a Boolean value that shows if the new name is set or not. Changes the name of the user specified with id. • setUserSurname(id: int, surname:String) returns a boolean value to show if the surname is set or not. Changes the surname of the user specified with id.

	• setUserEmail(id: int , email: String) returns a boolean value to show if the new email is set or not. Changes the email of the user specified with id. • setUserPassword(id: int, password:String) returns a boolean value to show if the new email is set or not. Changes the password of the user specified with id. • setStudentSection(id: int, section: int) returns a boolean value to show if the new section is set or not. Changes the section of the user specified with id. • setStudentGroup(id: int, groupID: int) returns a boolean value to show if the new group is set or not. Changes the group of the user specified with id. • getStudentList() returns an array of integers filled with student ids. • getTAList() returns an array of integers filled with TA ids. • getSectionList() returns an array of integers filled with section informations. • getGroupList() returns an array of integers filled with group informations.

Name	AssignmentController
Description	This class is a controller class for assignments.
Properties	• dataAccess: DAO • currentAssignments: ArrayList<int>
Methods	• createAssignment(title: String, description: String, dueDate:Date) returns boolean to show if creation was successful. This method creates a new assignment and adds it to the currentAssignments with given information. • getAssignment(assignmentIDassignmentID: int, assignmentTitle: String) returns an assignment from currentAssignments whose id and title is specified with parameters. • uploadArtifact(name: String, aFile: File, isFinal: boolean) returns Boolean to show if upload was successful. Uploads an artifact to the system. • getArtifact(assignmentID: int, artifactName: String) returns a file from the currentAssignments list with specified assignment attributes. • gradeAssignment(assignmentID: int, aGrade: double) uploads a grade for an assignment uploaded by a group. • setAssignmentDescription(assignmentID: int, desc:String) sets the description of a specified assignment with id. • setAssignmentTitle(assignmentID: int, newTitle:String) sets the title of a specified assignment with id.

	• uploadArtifactReview(writerID: int, text:String,date:Date, writer:String) uploads an artifact review to the database specified with information given with parameters. • getArtifactReviews(artifactName:String) returns array list of ArtifactReviews specified with given parameter. • editReview(artifactName:String ,reviewed:int, reviewName:String, newText:String, newDate:Date) edits an existing review specified with information given in parameters. • setNewDueDate(assignmentID:int, assignmenTitle:String, newDueDate:Date) sets a new due date for an assignment specified in parameters. • deleteReview(assignmentID:int, artifactTitle:String, artifactReviewID:int) deletes a review done to an artifact specified with parameters. • getAllArtifacts(assignmentID:int) returns an array list of files which are uploaded as artifacts to the specified assignment. • getAllArtifactTitles(assignmentID:int) returns an array list of strings containing artifact titles of an assignment specified with parameters. • getArtifactReview(assignmentID:int, artifactTitle:String, reviewed:int) returns an ArtifactReview class instance containing a peer review that is specified with parameters.

Name	PeerReviewController
Description	This class is a controller class for peer review functionality.
Properties	• dataAccess: DAO
Methods	• createPeerReview(giverID:int, desc:String, grade:double,date:Date,toStudentID:int) checks the information given in parameters and creates a new PeerReview • editPeerReview(giverID:int, desc:String, grade:double,date:Date,toStudentID:studentID) edits an existing peer review specified with given parameters. • getPeerReviews(studentID:int) returns an array list containing ReerReviews. Returns all the peer reviews a student got. • deletePeerReview(toStudentID:int, giverID:int) deletes a peer review that is specified with given information in parameters.

Name	GroupController
Description	This class is a controller class for groups.
Properties	• dataAccess: DAO
Methods	• createGroup(groupID: int) creates a new group with given groupID • getGroupData(groupID:int) returns a the instance of the group with given groupID. • addStudentToGroup(groupID:int, studentID:int) adds the student with given id to the group with given id.

	• removeStudentFromGroup(groupID:int, studentID:int) removes the student with given id from the group with given id. • addAssignment(groupID:int, assignmentID:int) adds an assignment to the specified group. • getAssignments(groupID:int) returns an array list of assignment ids of the group specified with groupID. • getMembers(groupID:int) returns an array list of student ids of the selected group. • removeAssignment(groupID:int, assignmentID:int) remove the specified assignment from specified group

Name	BusinessLogicTopController
Description	This is working like a master controller class that has instances of other controllers and uses them to navigate data.
Properties	• authControl: Controller • assignmentControl: Controller • peerReviewControl: Controller
Methods	• loginUser(id:int, password:String) uses authControl to perform a log in. • signUpUser(email:String, name:String, id:int, password:String, userType:String) uses authControl to perform a signup.

	• getStudentByGroup(groupID:int) returns an array list of students in specified group • getStudentBySection(sectionID:int) returns an array list of students in specified section. • getSectionList() returns array list of sections. • getGroupList() returns array list of groups. • getTAList() returns array list of TAs. • getAllStudents() returns array list of all students. • createAssignment(title:String, description:String, dueDate:Date) creates a new assignment by using assignmentController. • getAssignment(assignmentID:int, assignmnetTitle:String) retuens the specified assignment. • getAllAssignments() returns the array list of assignments. • updateAssignment(assignmentID:int, desc:String, dueDate:Date, title:String) updates the specified assignment with given values. • getArtifacts(assignmentID:int, groupID:int) returns a hashmap containing all artifacts of the specified group. • getArtifactReviews(assignmentID:int, groupID:int, artifactName:String) returns array list of artifact reviews of a group's specified artifact.

	• uploadArtifact(groupID:int, assignmentID:int, artifactTitle:String, artifact:File) uploads an artifact to a specified group. • getCurrentUserArtifacts() returns hashmap containing artifacts of the current users. • getCurrentUserArtifactReviews() returns hash map containing arraylist of reviews. • getCurrentUserGroupData() returns group class of current user. • getCurrentUserPeerReviews() returns array list of PeerReviews of current users. • getStudentData(studentID:int) returns Student class instance of specified student. • getTAData(taID:int) returns TA class instance of specified TA. • getInstructorData() returns Instructor class instance of the instructor. • getGroupData(groupID:int) returns Group class instance of specified Group. • deleteArtifact(groupID:int, assignmentID:int, artifactTitle:String) deletes a spesific artifact uplaoded from specified group. • givePeerReview(toStudentID:int, desc:String, grade:double) gives a peer review to a specified student.

	• editPeerReview(toStudentID:int, desc:String, grade:double) edits an existing peer review with specified information. • deletePeerReview(toStudentID:int,giverID:int) deletes a specified peer review. • getAllPeerReviews(studentID:int) returns all peer reviews of the specified student. • createGroup(groupID:int) creates a new group. • addStudentToGroup(studentID:int, groupID:int) adds a student to the specified group. • removeStudentFromGroup(studentID:int, groupID:int) removes a student from the specified group. • deleteAssignment(assignmentID:int) deletes an assignment.

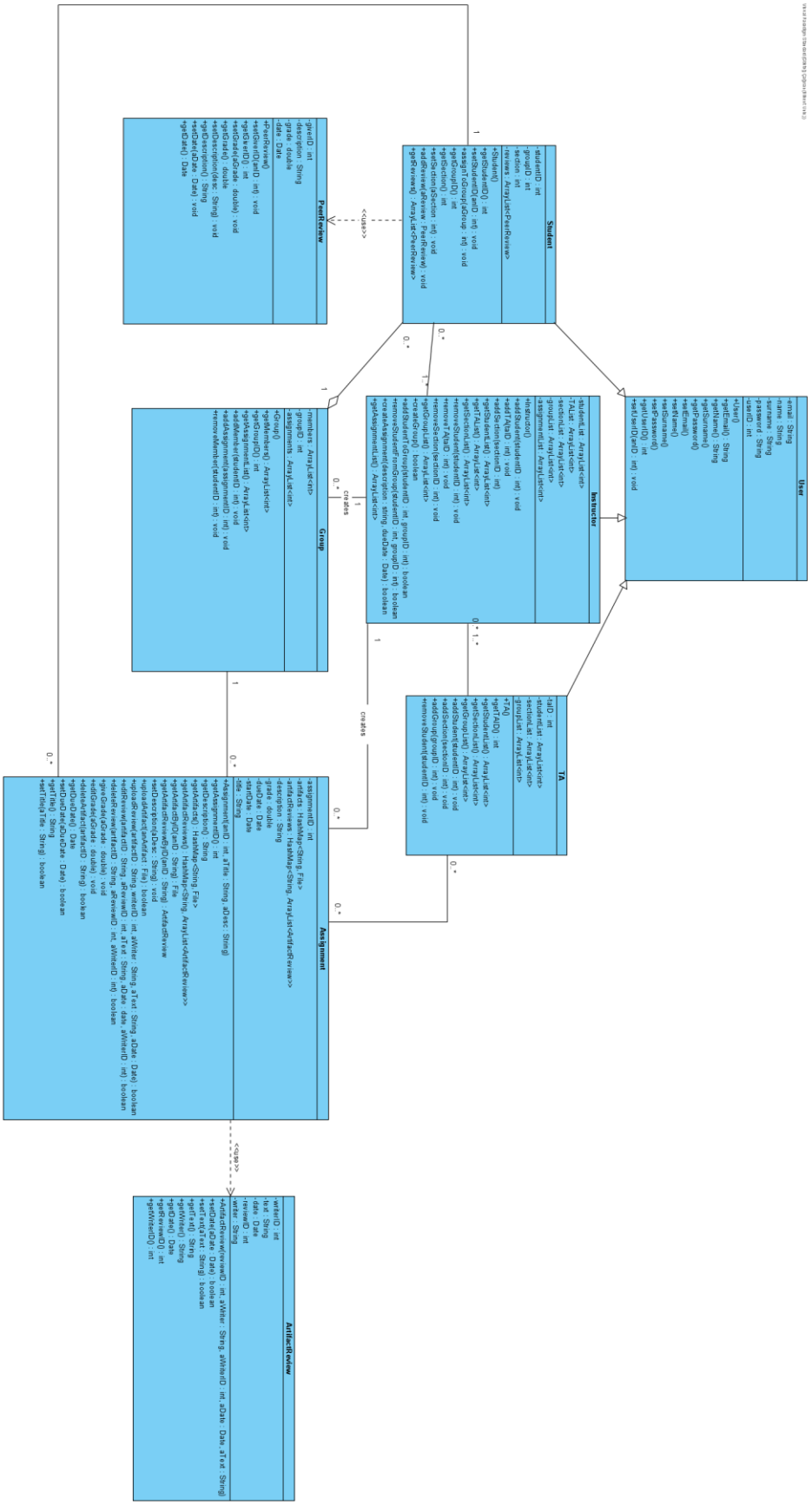
- **DAO Class**

Name	DAO
Description	Class that takes the data from database and navigates to the system.
Properties	•
Methods	• getGroup(groupID:int) returns a group class instance specified with groupID. • getAssignment(assignmentID:int) returns a specific assignment. • getUser(id:int) returns an instance of user class with specified id. • getArtifacts(groupID:int, assignmentID:int) returns a hash map of artifacts of specified group and assignment. • getSingleArtifact(groupID:int, assignmentID:int) returns the file • getArtifactReview(groupID:int, assignmentID:int, artifactTitle:int) gets a specific review. • loginUser(id:String, password:String) performs log in actions • signUpUser(email:String, name:String, id:int, password:String, userType:String) performs sign up operations. • createAssignment(assignmentID:int, title:String, desc:String, dueDate:Date) creates a new assignment on the database

	• updateAssignment(assignmentID:int, title:String, desc:String, dueDate:Date) updates an existing assignment. • deleteAssignment(assignmentID:int) deletes specified assignment from the database. • createGroup(groupID:int) creates a new group with specified id on database. • updateUserOfGroup(groupID:int, memberID:int, isAdd:boolean) updates the members of a group. • updateAssignmentOfGroup(groupID:int, assignmentID:int, isAdd:boolean) updates an existing assignmnet of a group. • deleteGroup(groupID:int) deletes a group from the database and the system. • createPeerReview(giverID:int, desc:String, grade:double, date:Date, toStudentID:int) creates a peer review on database. • updatePeerReview(giverID:int, desc:String, grade:double, date:Date, toStudentID:int) updates an existing peer review with given specifications on parameters. • deletePeerReview(toStudent:int, giverID:int) deletes an existing peer review from the database. • createArtifact(groupID:int, assignmentID:int, aFile:File, aTitle:String) creates a new artifact on database. • updateArtifact(groupID:int, assignmentID:int, aFile:File, aTitle:String) updates an existing artifact.

	• deleteArtifact(groupID:int, assignmentID:int, aTitle:String) deletes an artifact from the database. • createArtifactReview(groupID:int, assignmentID:int, artifactTitle:String, writerID:String, text:String, date:Date, reviewID:int, writer:String) creates a new artifact review on database • updateArtifactReview(groupID:int, assignmentID:int, artifactTitle:String, writerID:String, text:String, date:Date, reviewID:int, writer:String) updates an existing articaft review • deleteArtifactReview(groupID:int, assignmentID:int, artifactTitle:String,reviewID:int) deletes an artifact review with specified information. • updateUser(email:String, name:String, id:int, password:String, userType:String) updates information of a user with specified information. • deleteUser(id:int) deletes an existing user profile • gradeAssignment(groupID:int, assignmentID:int, grade:double) gives grade to an assignment. • getAll… methods return array list of the specified item in "…"

4.2.3 Class Diagram of Entities



- **Entity Classes**

Name	User
Description	User class contains the common properties between student, instructor and TA classes so it is a typical model for system users. Class contains the personal information of a user.
Properties	• email: String • name: String • surname: String • password: String • userID: int
Methods	• Getters: to return class properties • Setters: to make changes on class properties

Name	Student
Description	Student is the first type of a user. It contains group information, personal information and peer reviews. Instructor can work on this class.
Properties	• studentID: String • groupID: int • section: int • reviews: ArrayList<PeerReview>
Methods	• Getters: to return class properties • Setters: to make changes on class properties • assignToGroup(group: int): method works as a setter for groupID

	• addReview(review: PeerReview): to add an instance of PeerReview to reviews array list • getReviews(): returns an array list of PeerReview instances

Name	Instructor
Description	Model for a class instructor. Another type of user class. Contains student list, TA list and section list. This class has permission to work on TA and student classes, can create, change or delete a group; can upload, change or delete an assignment.
Properties	• studentList: ArrayList<int> • TAList: ArrayList<int> • sectionList: ArrayList<int> • groupList: ArrayList<int> • assignmentList: ArrayList<int>
Methods	• addStudent(studentID: int), addTA(taID: int) and addSection(sectioned:int) methods are used to add integers to array lists that instructor class holds • Getters: to return class properties • Remove methods remove an element specified by the parameter from the array list specified on method signature. • createGroup() creates a new group. • addStudentToGroup(studentID:int, groupID:int) adds the student whose information is passed with parameters to the specified group.

	• createAssignment(desc:String, dueDate:Date) creates new assignment with given information on parametes.

Name	TA
Description	Final type of user class. Contains personal information, student list, section list and group list. Has limited permission to work on student, assignment and group classes compared to an instructor.
Properties	• taID: int • studentList: ArrayList<int> • sectionList: ArrayList<int> • groupList: ArrayList<int>
Methods	• Getters: to return class properties • addStudent(studentID: int), addSection(sectioned:int), addGroup(groupID: int): add an integer value to the specified array list • removeStudent(studentID: int): removes an integer from the studentList array list

Name	PeerReview
Description	Model for the review that will be done by students to other students. It contains the information of writer and date of the review and review itself which is composed of description and grade. This class will be stored in student classes.
Properties	• giverID: int • description: String

	• grade: double • date: Date
Methods	• Getters: to return class properties • Setters: to make changes on class properties

Name	Group
Description	Model for the student groups that will be formed by instructors. Instructor and TA classes can both create or change a group class. It contains the id, members and the assignments of the group.
Properties	• Members: ArrayList<int> • groupID: int • assignment: ArrayLisy<int>
Methods	• Getters: to return class properties • addMember(studentID: int), addAssignment(assignmentID: int) adds an integer value to the lists stored as class poperty • removeMember(studentID: int): removes an integer value passed by parameters from the members array list

Name	Assignment
Description	This class will be held in Group class. It can be created or changed by both TA and Instructor classes. Class contains assignment id, artifacts that are uploaded by groups and artifact reviews.
Properties	• assignmentID: int • artifacts: HashMap<String, File>

	• artifactReviews: HashMap<String, ArrayList<ArtifactReview>> • description: String • grade:double • dueDate:Date • startDate:Date • title:String
Methods	• Getters: to return class properties. • getArtifactByID(id: String): finds a matching string from the artifacts hash map and returns a File • getArtifactReviewByID(id: String): finds a matching string from the artifactReviews hash map and returns an ArtifactReview • setDescription(desc: String): sets or changes description • uploadArtifact(art: File): returns boolean value to show if upload was successful. This method adds a file to the artifacts hash map • uploadReview(artifactID: String, writerID: int, text: String, date: Date):): returns boolean value to show if upload was successful. This method adds a Review to the artifactReviews hash map • deleteReview(artifactID. String, reviewID: int, writerID:int): deletes the review from the artifactReviews hash map specified with method parameters • giveGrade(aGrade:double) gives a value to grade property • editGrade(aGrade:double) edits grade property • deleteArtifact(artifactID:int) deletes an artifact

Name	ArtifactReview
Description	This class will be held in assignment class. These reviews will be created by instructor and TA classes. It holds the information of review writer id, date of review, review id and the review itself as a text.
Properties	• writer: int • text: String • date: Date • reviewID: int
Methods	• Getters: to return class properties • Setters: to make changes on class properties

4.3 External Packages

- **org.netbeans.lib.awtextra:** This package created by NetBeans contains extra buttons, panels and layouts that we use in in the UserInterface subsystem.
- **java.util:** This package contains various utility classes for Java.
- **java.sql:** This package is used to connect to databases with Java.
- **javax.swing:** This Java package contains all the necessary classes for building a Swing UI.
- **java.awt.event:** This Java package contains event listeners and handlers that we use to capture User interaction with the system.