Bilkent University

Department of Computer Engineering

CS 319 Term Project

ReviewTool

Group 1J Analysis Report – Iteration 2

Mustafa Anıl TAŞDAN, Göktuğ ÇAĞIRAN, Cihan Can KILIÇ

Instructor: Eray TÜZÜN

Teaching Assistant(s): Erdem TUNA, Elgun JABRAYILZADE

Table of Contents

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. [1] 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. [2]

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 Changes over Iteration 1

- Changes over the format of the report
- Reworked introduction
- Added overview
- Reworked functional requirements
- Reworked non-functional requirements
- Added scenarios
- Reworked use-cases
- Added object model
- Reworked dynamic model

1.3 Definitions and Abbreviations

TA: Teaching Assistant

UI: User Interface

JVM: Java Virtual Machine

MVP: Minimum Viable Product

1.4 References

1)Bruegge, B. (2010). Organizing Client and Project Reviews. In A. H. Dutoit (Ed.), *Object-Oriented Software Engineering Using UML, Patterns, and Java* (3rd Ed., p. 113). Prentice Hall.

2) *GitHub features: Intuitive code review tools*. GitHub. (n.d.). https://github.com/features/code-review/.

3)Techopedia. (2011, November 2). *What is a Minimum Viable Product (MVP)? - Definition from Techopedia*. Techopedia.com. https://www.techopedia.com/definition/27809/minimum-viable-product-mvp.

2. Proposed System

2.1 Overview

The main features of the application are: group creation, assignment creation, artifact submission, artifact reviewing and peer reviewing. These features are necessary for achieving an MVP[3], with extra features to be considered after MVP is achieved, as the development team is smaller than normal.

2.2 Functional Requirements

Depending on the login information, the application will switch into one of three states: Instructor, Student or TA.

2.2.1 Booting up the Application

The application boots to the login screen. The user can either input their credentials to login or press the sign-up button to switch to the sign-up screen.

2.2.2 The Login Screen

The user has to input their e-mail/student ID and their password. Pressing the login button will have the application switch to one of the three dashboards, student, TA or instructor dashboard, assuming the credentials were valid. Pressing the sign-up button will have the application switch to the sign-up screen.

2.2.3 The Sign-up Screen

The user has to enter their:

- Name-Surname
- E-Mail
- Password
- Course Code

to send a sign-up request. The user can use a button to return to the login screen.

2.2.4 Instructor Feature Requirements

Using the dashboard available to them, an instructor can:

- Manage groups
- Manage assignments
- View general progress of the groups
- Review artifacts uploaded by the groups

2.2.4.1 Managing Groups

Initially, the groups screen will contain no groups. The instructor has to create a group and add participants to the group. The instructor also has the ability to remove participants. The list of groups will be shown to the instructor in a compact view, clicking on a particular group will display the information of the group and the ability to manage the group. The information to be shown is:

- Information of each member
- The artifacts uploaded by the group
- The peer reviews submitted by the students.

2.2.4.2 Managing Assignments

The instructor can add or remove assignments for each group to fulfill. The assignments added here will show up in the dashboards of students. The instructor can enter a title and description for the assignment, and the assignment may require a file upload from the groups. The instructor can also add a due date for the assignment.

2.2.4.3 Viewing General Progress

The instructor can see the progress of each group in a compact view in the progress screen.

2.2.4.4 Reviewing Artifacts

The instructor can add reviews to an artifact uploaded by a group. When a student uploads an artifact, it goes through an internal review by the students, where another student has to approve the artifact for submission. When approved, instructors and TAs can see the submission and add their own reviews/comments, which will be visible to every participant of the group.

2.2.5 Student Feature Requirements

Using the dashboard available to them, a student can:

- View the details of their own group
- Upload artifacts
- Review artifacts
- Review peers

It is important that a student can only see the information of their own group and not any other group.

2.2.5.1 Viewing his/her Group

The student will be able to view the information of his group-mates from this screen. The student is able to upload/download files from this screen. The student also submits his peer reviews from this screen, but the button will be blocked until the instructor allows peer reviews to be submitted.

2.2.5.2 Uploading Artifacts

The student can upload an artifact file to the system, where it will be stored. Initially, the artifact will only be visible to fellow group members, where it will be pending for review. After another student approves the artifact, it will be submitted, visible to everyone with access to the group.

2.2.5.3 Reviewing Artifacts

In addition to the necessary artifact review discussed in the previous step, students can add optional reviews to their artifacts for group members to see, just like an Instructor or TA can do.

2.2.5.4 Reviewing Peers

When peer reviewing feature is enabled by the instructor, the student can submit peer reviews for their group members. These reviews have the form of a text and point based system. A student cannot see reviews made on them by others.

2.2.6 Teaching Assistant Feature Requirements

A TA account has the same abilities as an Instructor account, with the differences being:

- No ability to create/manage assignments
- No ability to enable peer reviews

The TA will be able to view/review artifacts. The account type is primarily for notifying the students of the reviewer's "rank", as in whether an Instructor, Student or TA did the review.

2.3 Non-Functional Requirements

2.3.1 Usability

As the system aims to solve the communications complexity between Instructors, TAs and Students, it should be simple, easy to understand and easy to use. If a Student has to contact course managers throughout the semester with questions about application functionality it would be against the purpose of the system.

2.3.2 Security

As the system will contain vital personal data, such as name, e-mail, password; and vital course data, such as artifacts, artifacts reviews, peer reviews, it is important that the application will be secure. Security means both security against externalities, and security considerations regarding internal access, such as a student not being able to view other groups' data.

2.3.3 Extensibility

As mentioned before, the application has a smaller set of features then desired, for reasons beyond our control. Because of this, it is important to design the application in a modular and extensible form, so that if desired, the application could be enhanced in the future.

2.3.4 Reliability and Compatibility

As mentioned in the Usability section, the application will be used by many types of users in the course. As such, it is important that the application is reliable with minimum downtime. It is also important that the application is able to run on many types of devices, which is one of the reasons we chose Java as the programming language to build the application, as the JVM can run on any machine.

3 System Models

3.1 Scenarios

Scenario names	groupCreation, memberManagement
Participating actor instances	alice: Instructor or TA
Flow of events	1. Alice logs in to her dashboard. 2. Alice clicks the “Create New Group” button. 3. Alice enters the group name and selects initial students. 4. Alice sees the updated “Groups” screen. 5. Alice clicks the “View Group” button. 6. Alice clicks the “Remove Member” button. 7. Alice clicks the “Add New Members” button. 8. Alice selects the students to add.

Scenario names	assignmentCreation, assignmentManagement
Participating actor instances	bob: Instructor
Flow of events	1. Bob logs in to his dashboard. 2. Bob clicks the “Create New Assignment” button. 3. Bob enters the assignment name and description. 4. Bob sees the updated “Assignments” screen. 5. Bob clicks the “View Assignment” button. 6. Bob clicks the “Edit Assignment” button and changes assignment details. 7. Bob clicks the “Remove Assignment” button.

Scenario names	artifactSubmission, artifactRemoval
Participating actor instances	charlie: Student, frank: Student
Flow of events	1. Charlie logs in to his dashboard. 2. Charlie clicks the “Assignments” button. 3. Charlie clicks the “View Assignment” button. 4. Charlie clicks the “Submit Assignment” button and selects a file. 5. Charlie sees his uploaded file. 6. Frank reviews and approves the submission. 7. Charlie sees Frank’s review. 8. Charlie clicks the “Remove Assignment” button.

Scenario names	artifactReviewSubmission, artifactReviewRemoval
Participating actor instances	david: Instructor, TA or Student
Flow of events	1. David logs in to his dashboard. 2. David clicks the “Groups” button. (Instructor and TA only) 3. David clicks the “View Group” button. (Instructor and TA only) 4. David clicks the “Assignments” button. 5. David clicks the “View Assignment” button. 6. David clicks the “Add Review” button and enters his review. 7. David sees his review on the screen, along with reviews made by others. The username attached to the review has a specific color indicating account type. 8. David clicks the “Remove Review” button.

Scenario names	peerReviewSubmission, peerReviewRemoval
Participating actor instances	eve: Student
Flow of events	1. Eve logs in to her dashboard. 2. Eve clicks the "Submit Peer Review" button. 3. Eve selects a fellow student with "Add Review" button. 4. Eve selects a score out of 10 and adds a text based comment. 5. Eve can remove a peer review by clicking the "Remove Review" button.

3.2 Use Case Model

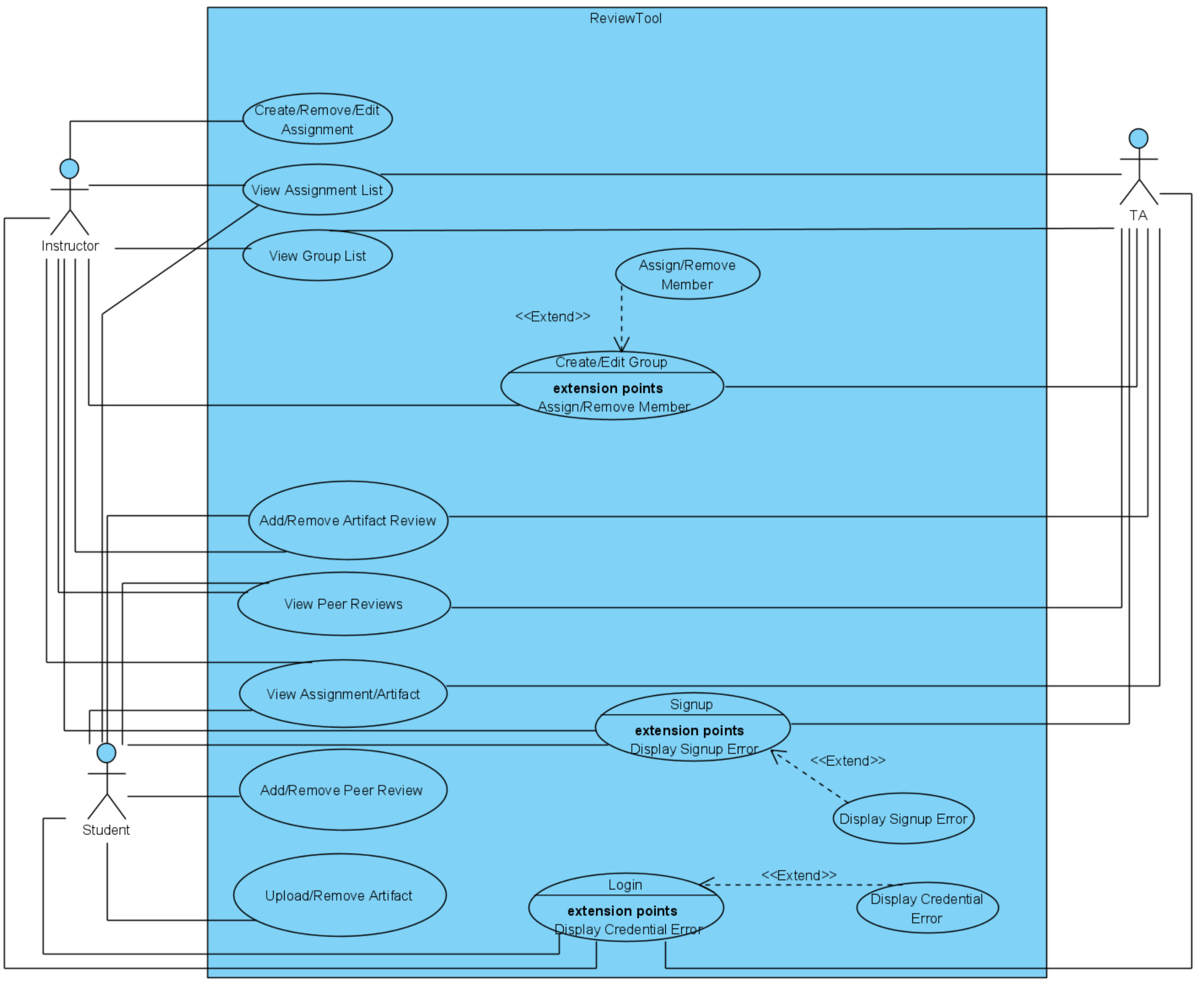


Figure 1: Use Case Diagram

Non-Trivial Use Cases

Use Case #1	Login
Participating actors	Initiated by any user
Entry condition	User runs the application.
Exit conditions	User successfully logs in, OR shuts the application down, OR clicks “Sign Up” button.
Flow of events	1. User enters “E-Mail” and “Password” and clicks the “Login” button. 2. The system checks the database if the credentials are correct. 3. The UI switches to the respective dashboard if the credentials are correct. It displays an error if login fails.

Use Case #2	Sign up
Participating *actors*	Initiated by any user
Entry condition	User clicks the “Sign Up” button.
Exit conditions	The account is created, OR user clicks the “Back” button.
Flow of events	1. User switches to “Sign Up” page. 2. User enters their “Name”, “E-Mail” and “Password” and clicks the “Sign Up” button. 3. The system sends the credentials to the database. 4. The system logs the user in if account creation happens. It displays an error if the account already exists in the database.

Use Case #3	Create/Edit Group
Participating actors	Initiated by Instructor or TA
Entry condition	User clicks the “Create New Group” button.
Exit conditions	The group is created, OR User aborts the process.
Flow of events	1. User clicks the “Create New Group” button. 2. The system shows a screen with a field to enter a group name and a list of unassigned students. 3. User enters the group name and selects students. 4. The system initializes new users and group. The group appears on the “Groups” screen. 5. User can edit any group on the screen with the “Edit Group” button.

Use Case #4	Assign/Remove Member
Participating actors	Initiated by Instructor or TA
Extension-points	This use case **extends** the Create/Edit Group use case.
Flow of events	1. User clicks the “Add New Members” button. 2. The system shows a screen with a list of unassigned students. 3. User selects students. 4. The system adds the students to the group and updates the screen. 5. User can click the “Remove Member” button near any student to remove them from the group.

Use Case #5	Create/Remove/Edit Assignment
Participating actors	Initiated by Instructor
Entry condition	Instructor clicks the “Create New Assignment” button.
Exit conditions	The assignment is created, OR Instructor aborts the process
Flow of events	1. Instructor clicks the “Create New Assignment” button. 2. The system shows a screen with a field to enter an assignment name and a field to enter description. 3. Instructor fills the fields. 4. The system initializes the new assignment. The assignment appears on the “Assignments” screen. 5. Instructor can click the “Edit Assignment” button near any assignment to alter it.

Use Case #6	Add/Remove Artifact Review
Participating actors	Initiated by Instructor, TA or Student
Entry condition	User clicks the “Add Artifact Review” button.
Exit conditions	The review is added, OR user aborts the process
Flow of events	1. User clicks the “Add Review” button near the desired artifact. 2. The system shows a screen with a field to enter a review. 3. User fills the field. 4. The system adds the review to the artifact. All reviews made by various users is shown on the screen. 5. User can click the “Remove Review” button near any of their reviews to remove it.

Use Case #7	Add/Remove Peer Review
Participating actors	Initiated by Student
Entry condition	User clicks the “Add Peer Review” button.
Exit conditions	The review is added, OR user aborts the process
Flow of events	1. Student clicks the “Add Review” button near the desired group member. 2. The system shows a screen with a field to enter a review and an interface to select a score out of 10. 3. Student fills the fields. 4. The system adds the review to the user. The reviewer cannot see reviews made by others. Students cannot see reviews made to them. 5. Student can click the “Remove Review” button near his review to remove it.

Use Case #8	Upload/Remove Artifact
Participating actors	Initiated by Student
Entry condition	User clicks the “Upload Artifact” button.
Exit conditions	The artifact is uploaded, OR user aborts the process
Flow of events	1. Student clicks the “View Assignment” button. 2. Student clicks the “Upload Artifact” button. 3. Student selects a file using a file chooser. 4. The system uploads the file to the database. 5. The system shows the file as pending for review. When a student reviews the file, the system shows it as submitted. 6. Student can remove the file during “pending for review” or “submitted” states.

3.3 Object Model

3.3.1 Class Diagrams

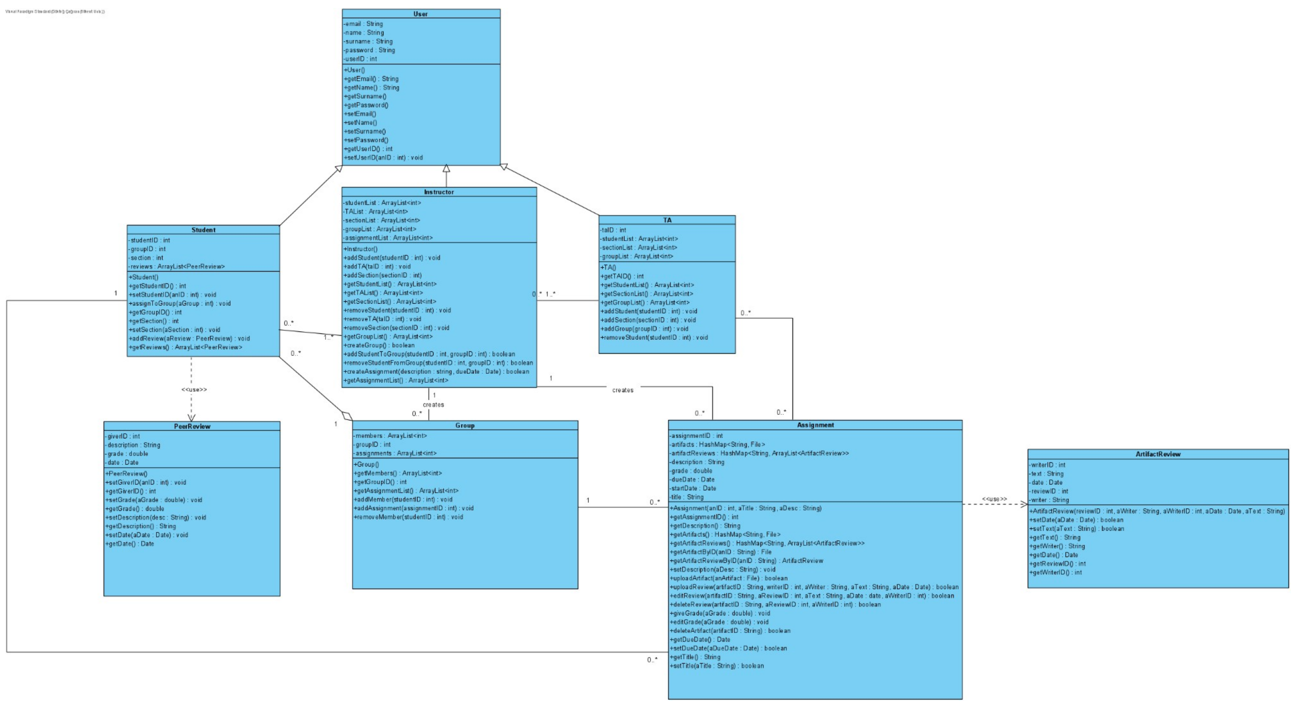

User
-email : String
-name : String
-surname : String
-password : String
-userID : int
+User()
+getEmail() : String
+getName() : String
+getSurname()
+getPassword()
+setEmail()
+setName()
+setSurname()
+setPassword()
+getUserID() : int
+setUserID(anID : int) : void
Instructor
-studentList : ArrayList<int>
-TAList : ArrayList<int>
-sectionList : ArrayList<int>
-groupList : ArrayList<int>
-assignmentList : ArrayList<int>
+Instructor()
+addStudent(studentID : int) : void
+addTA(taID : int) : void
+addSection(sectionID : int)
+getStudentList() : ArrayList<int>
+getTAList() : ArrayList<int>
+getSectionList() : ArrayList<int>
+removeStudent(studentID : int) : void
+removeTA(taID : int) : void
+removeSection(sectionID : int) : void
+getGroupList() : ArrayList<int>
+createGroup() : boolean
+addStudentToGroup(studentID : int, groupID : int) : boolean
+removeStudentFromGroup(studentID : int, groupID : int) : boolean
+createAssignment(description : string, dueDate : Date) : boolean
+getAssignmentList() : ArrayList<int>
Student
-studentID : int
-groupID : int
-section : int
-reviews : ArrayList<PeerReview>
+Student()
+getStudentID() : int
+setStudentID(anID : int) : void
+assignToGroup(aGroup : int) : void
+getGroupID() : int
+getSection() : int
+setSection(aSection : int) : void
+addReview(aReview : PeerReview) : void
+getReviews() : ArrayList<PeerReview>
TA
-taID : int
-studentList : ArrayList<int>
-sectionList : ArrayList<int>
-groupList : ArrayList<int>
+TA()
+getTAID() : int
+getStudentList() : ArrayList<int>
+getSectionList() : ArrayList<int>
+getGroupList() : ArrayList<int>
+addStudent(studentID : int) : void
+addSection(sectionID : int) : void
+addGroup(groupID : int) : void
+removeStudent(studentID : int) : void
1
0..*
1..*
0..*
0..* 1..*
0..*
<<use>>
1
creates
0..*
1
creates
0..*
0..*
PeerReview
-giverID : int
-description : String
-grade : double
-date : Date
+PeerReview()
+setGiverID(anID : int) : void
+getGiverID() : int
+setGrade(aGrade : double) : void
+getGrade() : double
+setDescription(desc : String) : void
+getDescription() : String
+setDate(aDate : Date) : void
+getDate() : Date
1
Group
-members : ArrayList<int>
-groupID : int
-assignments : ArrayList<int>
+Group()
+getMembers() : ArrayList<int>
+getGroupID() : int
+getAssignmentList() : ArrayList<int>
+addMember(studentID : int) : void
+addAssignment(assignmentID : int) : void
+removeMember(studentID : int) : void
1
0..*
Assignment
-assignmentID : int
-artifacts : HashMap<String, File>
-artifactReviews : HashMap<String, ArrayList<ArtifactReview>>
-description : String
-grade : double
-dueDate : Date
-startDate : Date
-title : String
+Assignment(anID : int, aTitle : String, aDesc : String)
+getAssignmentID() : int
+getDescription() : String
+getArtifacts() : HashMap<String, File>
+getArtifactReviews() : HashMap<String, ArrayList<ArtifactReview>>
+getArtifactByID(anID : String) : File
+getArtifactReviewByID(anID : String) : ArtifactReview
+setDescription(aDesc : String) : void
+uploadArtifact(anArtifact : File) : boolean
+uploadReview(artifactID : String, writerID : int, aWriter : String, aText : String, aDate : Date) : boolean
+editReview(artifactID : String, aReviewID : int, aText : String, aDate : date, aWriterID : int) : boolean
+deleteReview(artifactID : String, aReviewID : int, aWriterID : int) : boolean
+giveGrade(aGrade : double) : void
+editGrade(aGrade : double) : void
+deleteArtifact(artifactID : String) : boolean
+getDueDate() : Date
+setDueDate(aDueDate : Date) : boolean
+getTitle() : String
+setTitle(aTitle : String) : boolean
0..*
<<use>>
ArtifactReview
-writerID : int
-text : String
-date : Date
-reviewID : int
-writer : String
+ArtifactReview(reviewID : int, aWriter : String, aWriterID : int, aDate : Date, aText : String)
+setDate(aDate : Date) : boolean
+setText(aText : String) : boolean
+getText() : String
+getWriter() : String
+getDate() : Date
+getReviewID() : int
+getWriterID() : int

Figure 2: Application Domain Class Diagram

3.3.2 Data Dictionary

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

3.4 Dynamic Model

3.4.1 Sequence Diagrams

Create Group

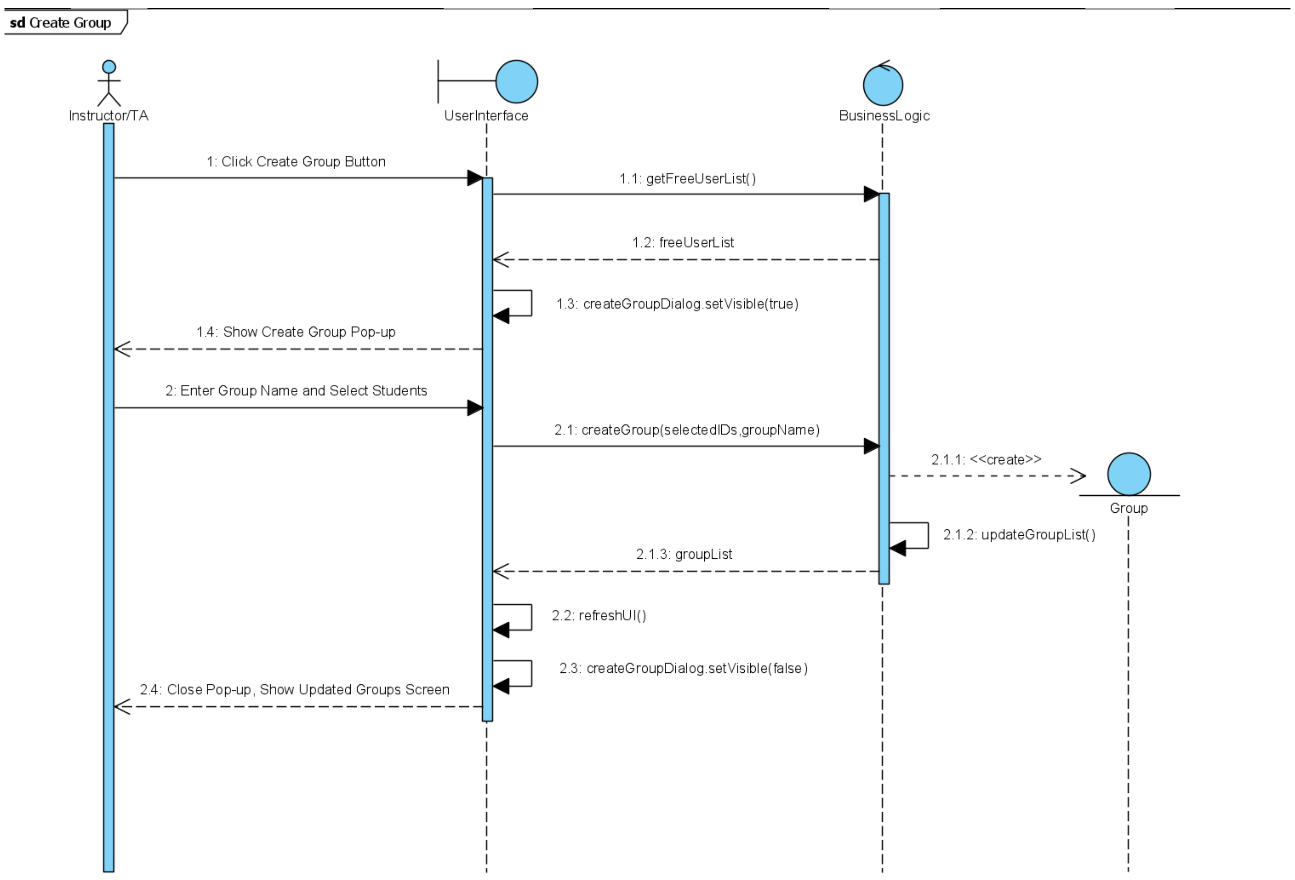

sd Create Group
Instructor/TA
UserInterface
BusinessLogic
Group
1: Click Create Group Button
1.1: getFreeUserList()
1.2: freeUserList
1.3: createGroupDialog.setVisible(true)
1.4: Show Create Group Pop-up
2: Enter Group Name and Select Students
2.1: createGroup(selectedIDs,groupName)
2.1.1: <<create>>
2.1.2: updateGroupList()
2.1.3: groupList
2.2: refreshUI()
2.3: createGroupDialog.setVisible(false)
2.4: Close Pop-up, Show Updated Groups Screen

Figure 3: Group Creation Sequence Diagram

First, the user logs in to the system, which is not shown. If the user’s account type is an Instructor or TA, the “Create Groups Button” on the “Groups” screen will be visible to the user. When the button is clicked, the boundary object UI will request the freeUserList from the control object BusinessLogic. The freeUserList will be a data structure that holds the users that are currently not assigned to a group. The UI will draw a pop-up screen showing the unassigned users. Then, the actor will enter a group name and select the users they want. BusinessLogic object will handle the creation of a new group with the selected users. In the end, the UI will be updated according to the changes.

Upload Artifact

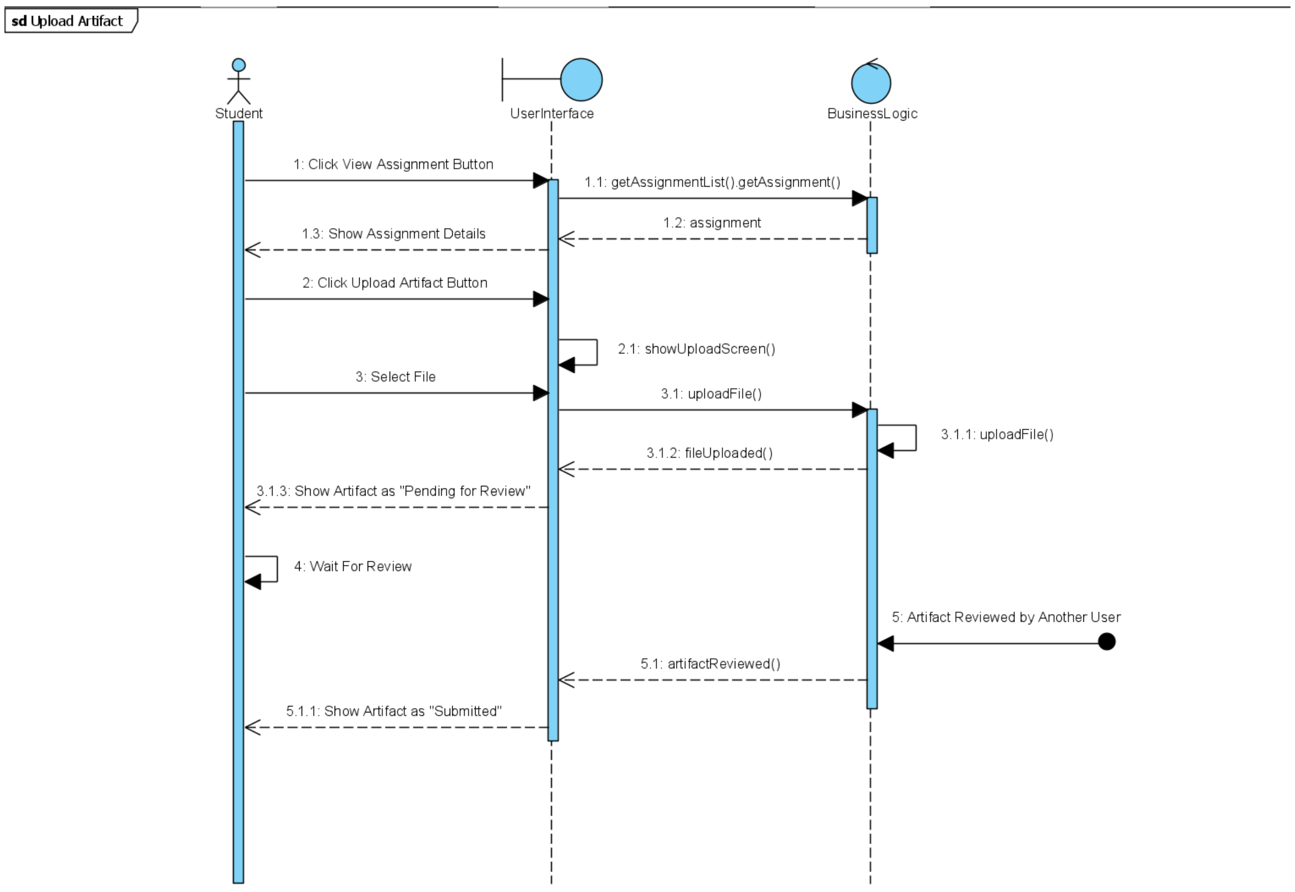

sd Upload Artifact
Student
UserInterface
BusinessLogic
1: Click View Assignment Button
1.1: getAssignmentList().getAssignment()
1.2: assignment
1.3: Show Assignment Details
2: Click Upload Artifact Button
2.1: showUploadScreen()
3: Select File
3.1: uploadFile()
3.1.1: uploadFile()
3.1.2: fileUploaded()
3.1.3: Show Artifact as "Pending for Review"
4: Wait For Review
5: Artifact Reviewed by Another User
5.1: artifactReviewed()
5.1.1: Show Artifact as "Submitted"

Figure 4: Artifact Upload Sequence Diagram

This action can only be taken by a Student account type. The student logs into the system and selects an assignment that is previously created by an Instructor account. There, the Student clicks “Upload Artifact” button, which takes them to a file chooser screen. The Student selects the file they want. The BusinessLogic object handles the creation of the artifact and upload of the file. The artifact enters a “Pending for Review” state, and waits until another Student adds a review and approves the upload. Then, the artifact enters the “Submitted” state and is visible to all Users with access to the group.

Add/View Artifact Reviews

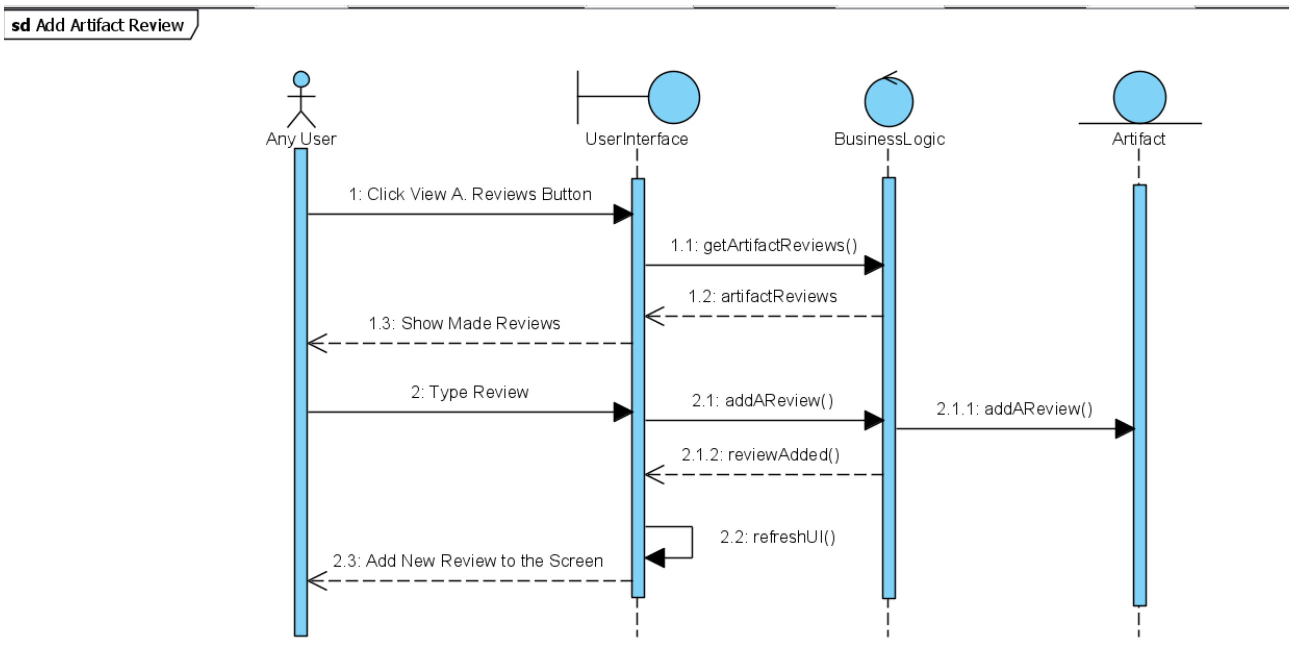


Figure 5: Add Artifact Review Sequence Diagram

This action can be taken by any User that has access to the group/artifact. Clicking the “View Artifact Reviews” button takes the User to a screen with artifact details and reviews made by other Users. The reviews will show User type, User name and review text. The User can type their own review in to the text field below and click a button, which makes the system add the new review to the artifact and refresh the UI.

3.4.2 State Diagrams

Group Object

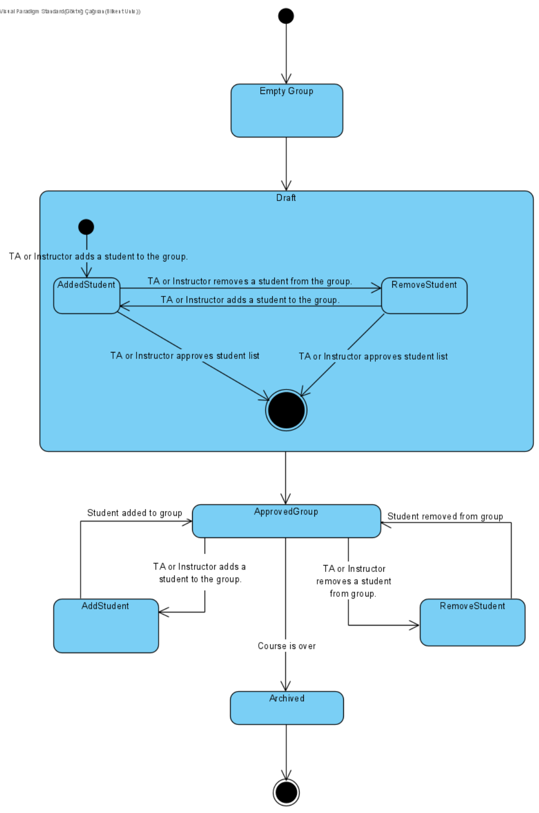

Empty Group
Draft
TA or Instructor adds a student to the group.
AddedStudent
TA or Instructor removes a student from the group.
RemoveStudent
TA or Instructor adds a student to the group.
TA or Instructor approves student list
TA or Instructor approves student list
Student added to group
ApprovedGroup
Student removed from group
TA or Instructor adds a student to the group.
TA or Instructor removes a student from group.
AddStudent
RemoveStudent
Course is over
Archived

Figure 6: Group Object State Diagram

When creating a new group initial state is empty group. Then we directly pass to draft state where TAs or instructors add a student to an empty group after this removing and adding student continues until TAs or instructors approve the student list. In this stage groups are forming for the first time. When instructor or TA approves the group, state of the group changes to approved group where TAs or instructors still can add or remove students to apply changes to the groups. At the end when the course is over group is archived.

Peer Review Object

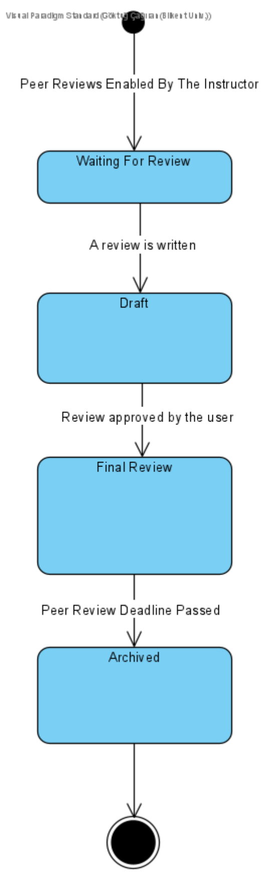
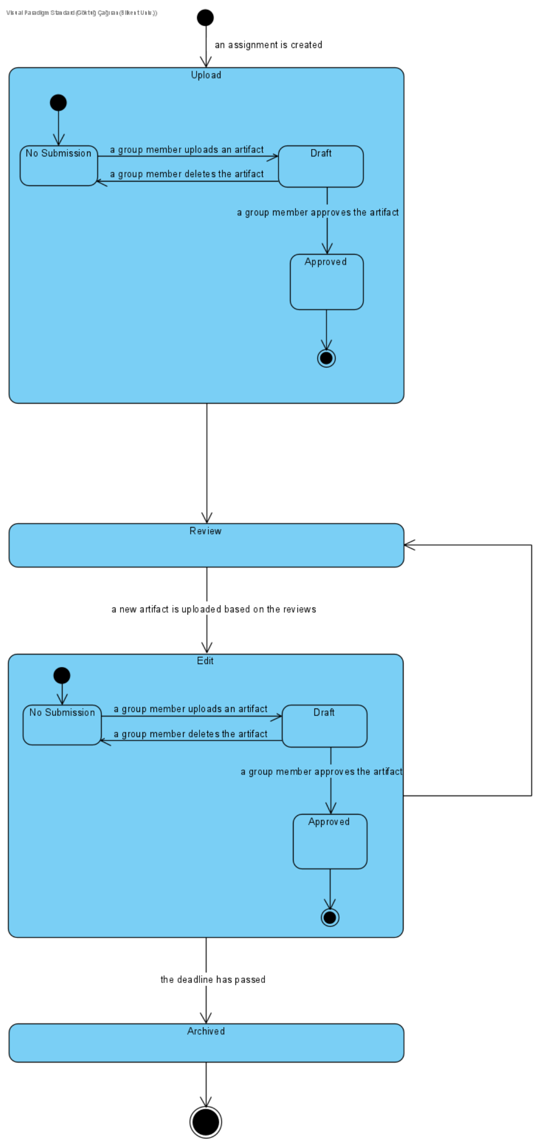

Visual Paradigm Standard(Göktuğ Çağıran(Bilkent Univ.))
Peer Reviews Enabled By The Instructor
Waiting For Review
A review is written
Draft
Review approved by the user
Final Review
Peer Review Deadline Passed
Archived

Figure 7: Peer Review Object State Diagram

To enter the first state instructor should enable the peer reviews. After it is enabled peer review state changes to waiting for review. When someone uploads a peer review, state changes to draft. After that when the user approves his/her review, state changes to final review. At the end when the deadline is met reviews are archived.

Artifact Object



an assignment is created
Upload
No Submission
a group member uploads an artifact
a group member deletes the artifact
Draft
a group member approves the artifact
Approved
Review
a new artifact is uploaded based on the reviews
Edit
No Submission
a group member uploads an artifact
a group member deletes the artifact
Draft
a group member approves the artifact
Approved
the deadline has passed
Archived

Figure 8: Artifact Object State Diagram

First of all the assignment is created by instructor from the assignments page. Then the uploading stage starts. When an artifact is submitted, submission stage changes from no submission to draft. If a student deletes the draft then it changes back to no submission. If a group member approves the artifact then the state changes to approved. Then the reviewing stage begins where TAs or instructors upload a review to uploaded artifact. After reviews are done editing stage begins this stage is exactly the same as first uploading stage. At the end when the deadline is met all artifacts are archived.

Student Object

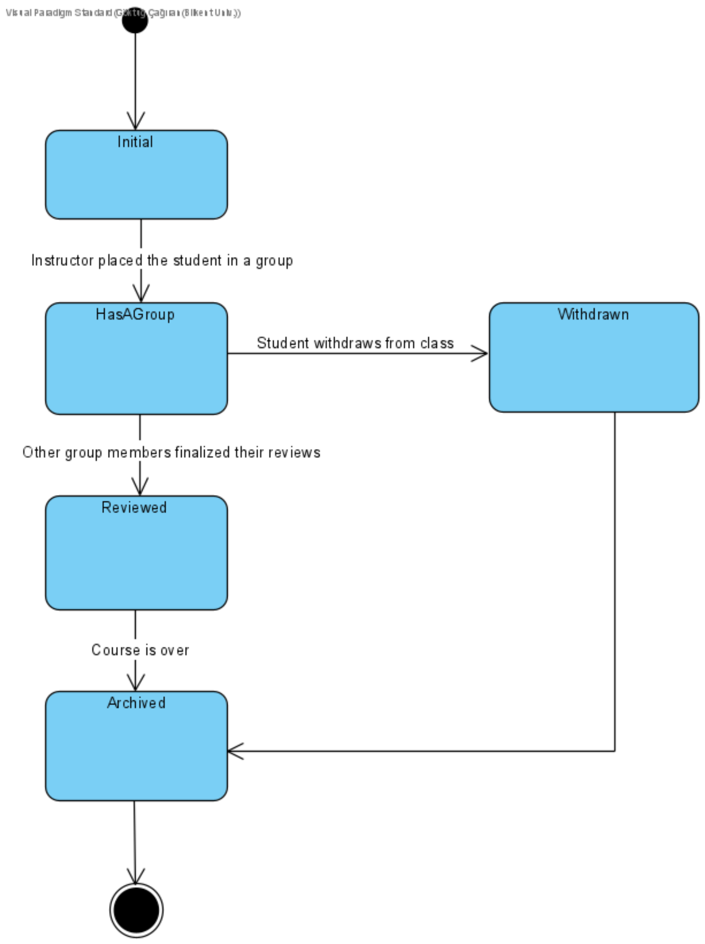

Initial
Instructor placed the student in a group
HasAGroup
Student withdraws from class
Withdrawn
Other group members finalized their reviews
Reviewed
Course is over
Archived

Figure 9: Student Object State Diagram

Initial state of a student changes to “hasAGroup” when the instructor places the student in a group. Through the course if the student withdraws from the class state directly passes to withdrawn then archived. If the student doesn’t withdraw then system waits for other group members’ peer reviews which they should upload and approve. Then when the course is over it is archived.

3.4.3 Activity Diagrams

Login and Signup

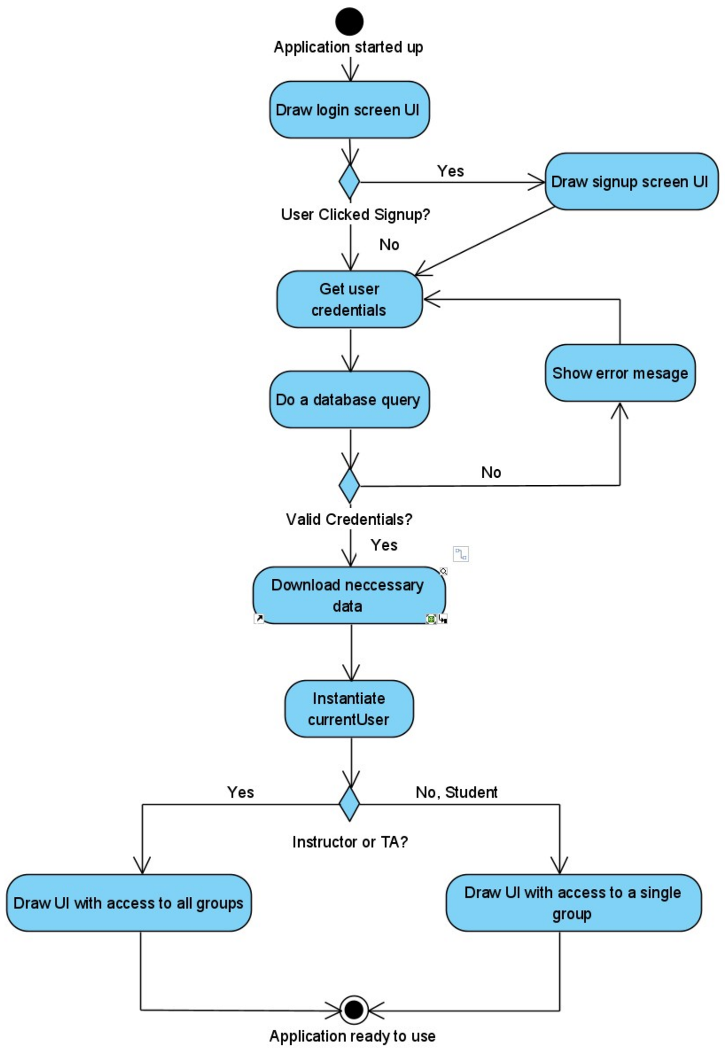


Figure 10: Login and Signup Activity Diagram

Upon application startup, the User is greeted with the Login Screen UI. The UI has E-Mail and Password fields, which the user can enter their credentials. The User can login, or click the “SignUp” button, which takes them to another screen with credential fields for signup. Both methods result in a database query, for login the system checks if the user exists, for signup the system checks if the user does not exist. The system draws the respective UI Dashboard based on account type in the case of a successful login/signup, or shows an error otherwise.

Group Management

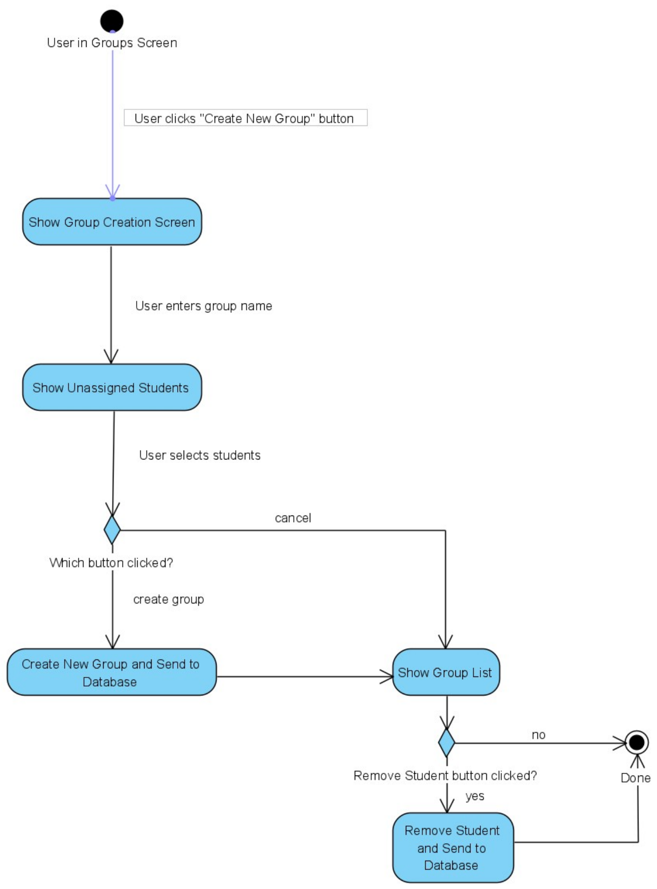
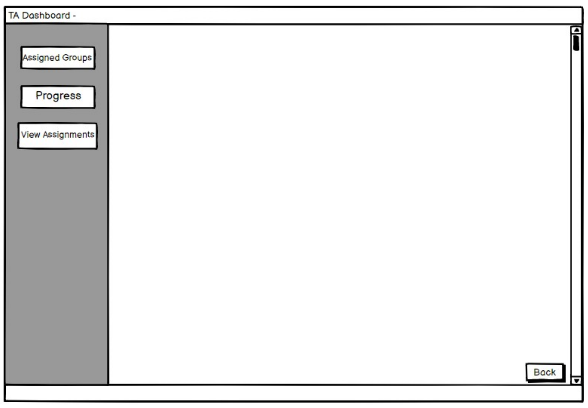
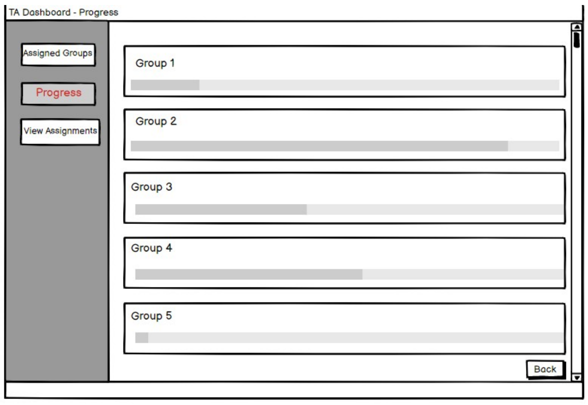

User in Groups Screen
User clicks "Create New Group" button
Show Group Creation Screen
User enters group name
Show Unassigned Students
User selects students
cancel
Which button clicked?
create group
Create New Group and Send to Database
Show Group List
no
Remove Student button clicked?
Done
yes
Remove Student and Send to Database

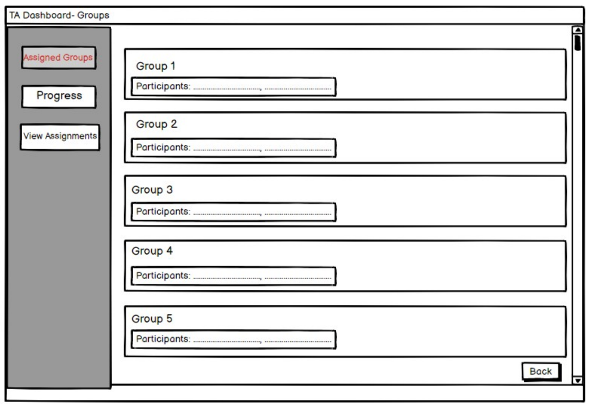
Figure 11: Group Management Activity Diagram

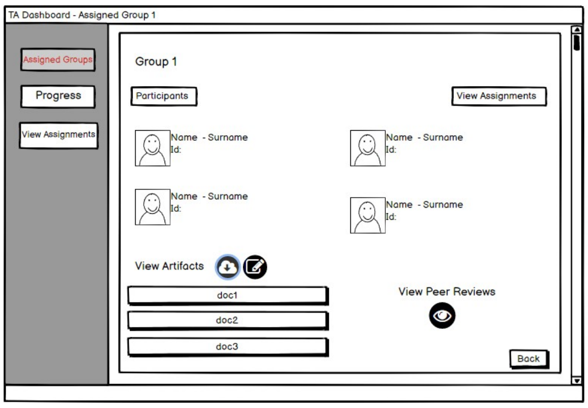
When in the “Groups” screen, an Instructor or TA sees all existing groups, with the ability to view their details if desired. There is also a “Create New Group” button, which takes the user to a screen with a field to enter a group name and a list of students without groups. Upon selection of users and a group name, the system creates a new group and stores it in the database. In the group details screen, each student has a “Remove Student” button in their panel, which when clicked removes them from the group, adds them to the list of students without groups, and updates the database.

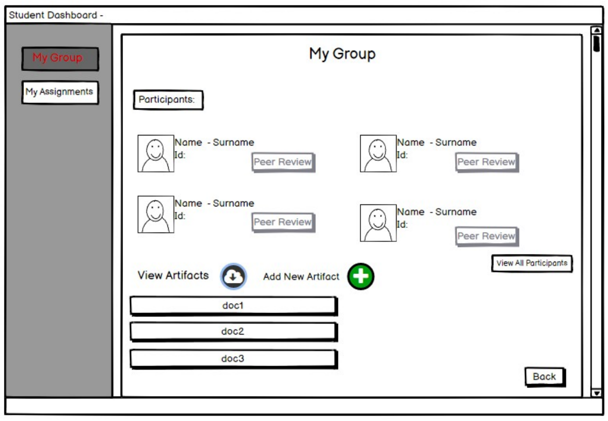
3.5 User Interface

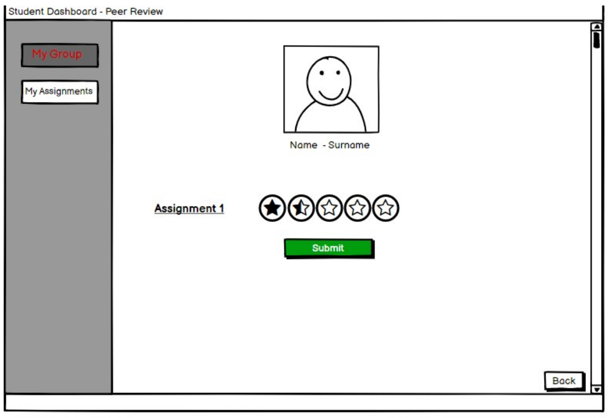


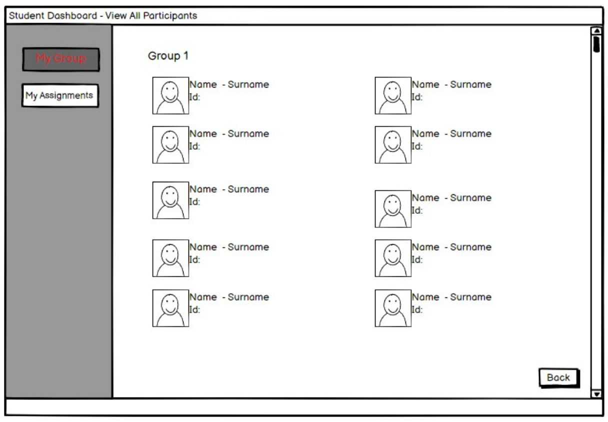


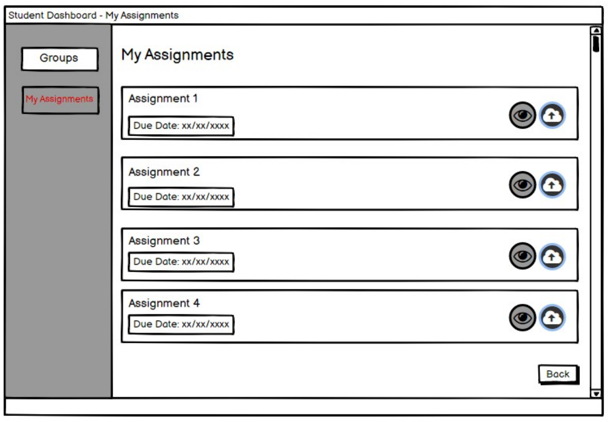


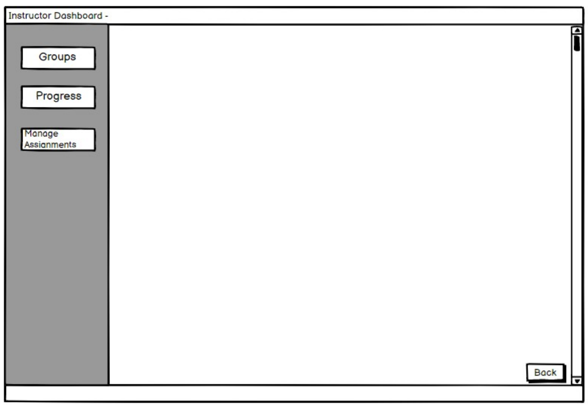


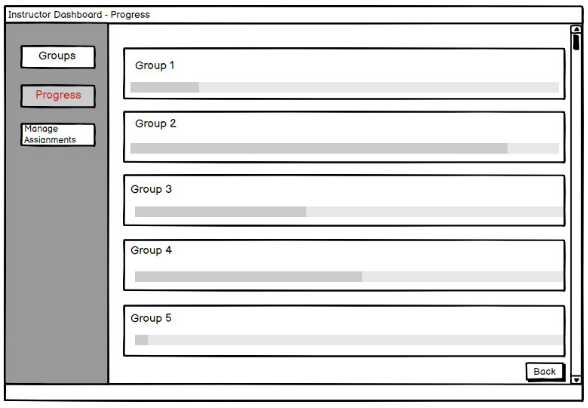


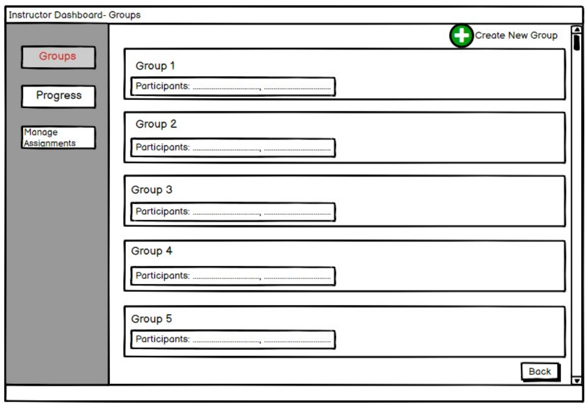


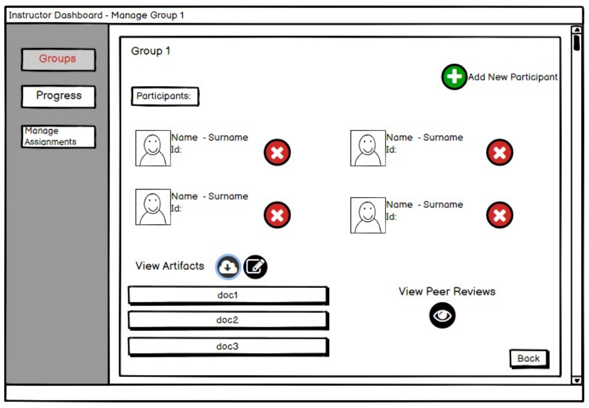


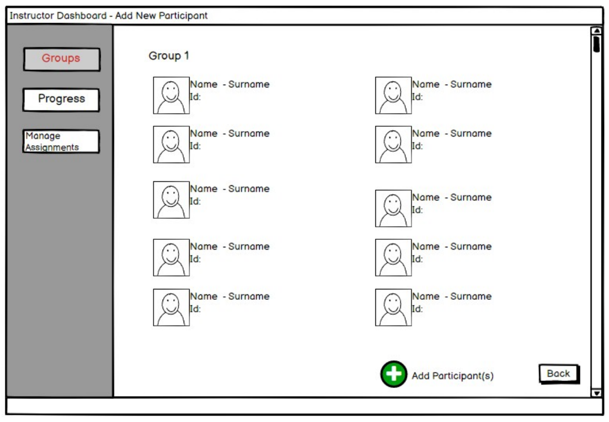


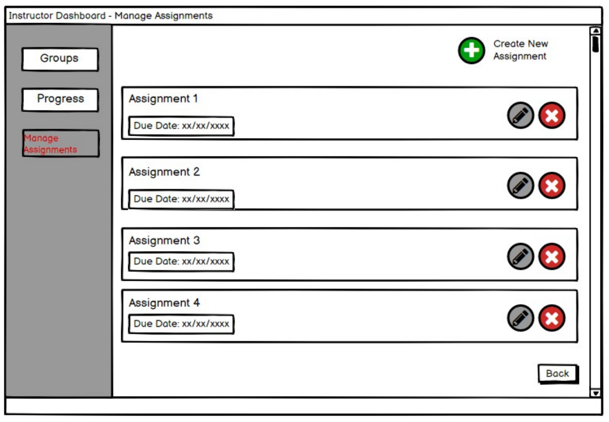

Instructor Dashboard -
Groups
Progress
Manage Assignments
Back

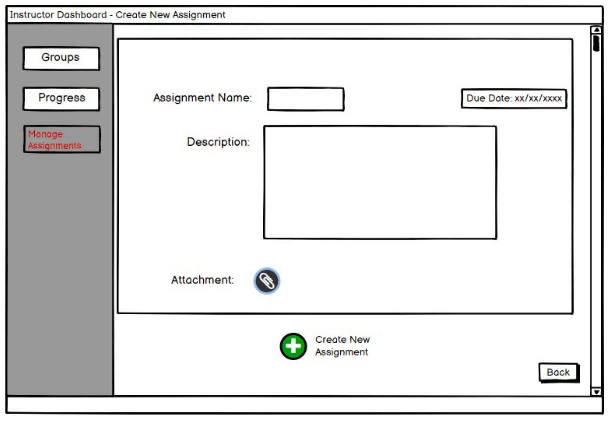

Instructor Dashboard - Progress
Groups
Progress
Manage Assignments
Group 1
Group 2
Group 3
Group 4
Group 5
Back

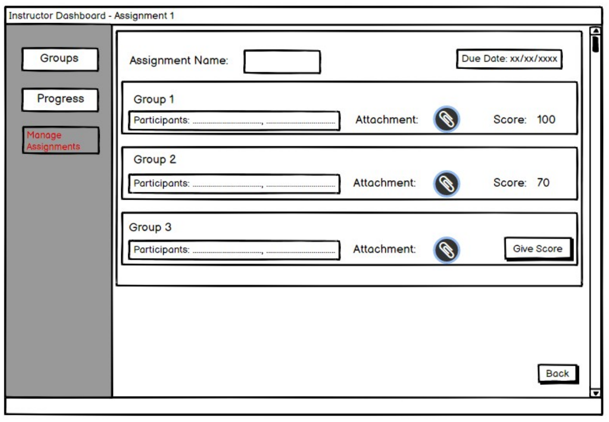

Instructor Dashboard- Groups
Groups
Progress
Manage Assignments
Create New Group
Group 1
Participants: ...., ...
Group 2
Participants: ...., ...
Group 3
Participants: ...., ...
Group 4
Participants: ...., ...
Group 5
Participants: ...., ...
Back


Instructor Dashboard - Manage Group 1
Groups
Progress
Manage Assignments
Group 1
Add New Participant
Participants:
Name - Surname
Id:
Name - Surname
Id:
Name - Surname
Id:
Name - Surname
Id:
View Artifacts
doc1
doc2
doc3
View Peer Reviews
Back


Instructor Dashboard - Add New Participant
Groups
Progress
Manage Assignments
Group 1
Name - Surname
Id:
Name - Surname
Id:
Name - Surname
Id:
Name - Surname
Id:
Name - Surname
Id:
Name - Surname
Id:
Name - Surname
Id:
Name - Surname
Id:
Name - Surname
Id:
Name - Surname
Id:
Add Participant(s)
Back


Instructor Dashboard - Manage Assignments
Groups
Progress
Manage Assignments
Create New Assignment
Assignment 1
Due Date: xx/xx/xxxx
Assignment 2
Due Date: xx/xx/xxxx
Assignment 3
Due Date: xx/xx/xxxx
Assignment 4
Due Date: xx/xx/xxxx
Back


Instructor Dashboard - Create New Assignment
Groups
Progress
Manage Assignments
Assignment Name:
Due Date: xx/xx/xxxx
Description:
Attachment:
Create New Assignment
Back


Instructor Dashboard - Assignment 1
Groups
Progress
Manage Assignments
Assignment Name:
Due Date: xx/xx/xxxx
Group 1
Participants: ...., ...
Attachment:
Score: 100
Group 2
Participants: ...., ...
Attachment:
Score: 70
Group 3
Participants: ...., ...
Attachment:
Give Score
Back