Bilkent University
Department of Computer Engineering

CS 319 Object-Oriented Software Engineering Term Project

CS319: Peer Review App
SNITCH

Analysis Report IT2

Group 1-I

<u>Group Members:</u>
Ümit Çivi 21703064
Onat Korkmaz 21704028
Abdulkadir Erol 21703049
Erdem Ege Eroğlu 21601636
Muhammed Maruf Şatır 21702908

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

Progress/Analysis Report IT2
25.04.2021
This report is submitted to the Department of Computer Engineering of Bilkent University.

Table of Contents

1 Introduction 4

2 Proposed System 4

2.1 Overview 4

2.1.1 Instructor / TAs 4

2.1.2 Students 5

2.2 Functional Requirements 5

2.2.1 Sign Up and Login 5

2.2.2 Manage project groups 5

2.2.2.1 Create a group 5

2.2.2.2 Assign student to a group 5

2.2.2.3 Delete group 4

2.2.2.4 Change a student's group 6

2.2.3 View an Information Card 6

2.2.3.1 Edit User Information....6

2.2.3.2 Withdraw or Drop Course 6

2.2.4 Review Panels 6

2.2.4.1 View Groups List 7

2.2.4.2 Select User Group 7

2.2.4.2.1 Peer Review 7

2.2.4.3 Select Other Groups 7

2.2.4.3.1 Review Others' Artifacts 7

2.3 Non-Functional Requirements....7

2.3.1 User interface and Human Factors 7

2.3.2 Documentation 7

2.3.3 Hardware Considerations 8

2.3.4 Performance Characteristics 8

2.3.5 Error Handling and Extreme Conditions 8

2.3.6 System Interfacing 8

2.3.7 System Modifications 8
2.3.8 Physical Environment 9
2.3.9 Security Issues 9
2.3.10 Usability 9
2.3.11 Compatibility 9
2.4 Pseudo Requirements 9
2.4.1 Language 9
2.4.2 Version Control System 9
2.5 System Models 10
2.5.1 Use Case Models 10
2.5.2 Dynamic Models 31
2.5.2.1 Activity Diagram 31
2.5.2.1.1 Host Activity Diagram..31
2.5.2.1.2 Student Activity Diagram 32
2.5.2.2 State Machine Diagram 33
2.5.2.2.1 Student State Diagram..33
2.5.2.2.2 Submission State Diagram 34
2.5.2.3 Sequence Diagrams 35
2.5.2.3.1 PeerReview Sequence Diagram..35
2.5.2.3.2 Artifact Review Sequence Diagram 36
2.5.2.3.3 Student Login Sequence Diagram..36
2.5.2.3.4 Student Sign Up Sequence Diagram 37
2.5.2.3.4 Create Groups Sequence Diagram 37
2.5.2.3.4 Add Student to Group Sequence Diagram 37
2.5.3 Object and Class Models 38
2.5.4 User Interface – Navigational Paths and Screen Mock-Ups 48
3 Improvements Summary..59

1. Introduction

Snitch is a web-based application aimed to review peers in a group project of any course which has a group project. The application provides students to scale/comment on peers of the same group members individually. Moreover, the students can comment on other groups as well. Instructor and TAs have control over the visibility of the evaluations, only they can see these evaluations in order to prevent foreseen problems among groups/group members.

The purpose of the application is to ease the instructor and TAs work during group projects. The hosts and TAs do not have to be involved in part in peer-reviewing. The reviewing process will be done without the control of hosts and TAs and it will be organized. The hosts and TAs can categorize, arrange, or rank the reviews after the process is over.

Instructors & TAs will be assigned and used as **"host"** in the report.

2. Proposed System

2.1 Overview

The application will start by logging in to the application. If the students have not signed up yet, they should sign up first. Other roles do not require signup. After signing in to the application, the options differ between instructor/TAs and students. Students can edit their profile information whenever they want. Students can request group change in case they are not satisfied with their current group. Moreover, students can upload their artifacts to the system before the given deadline. After the deadline, the students or hosts can review artifacts and write comments about them. Peer review is also made only by students when they need to give peer grades at the end of the semester. What instructor and TAs do is different and more complex than the students'.

2.1.1 Instructor / TAs

Instructors or TAs will create groups with unique numbers and they will assign students to a particular group. After the group formation is done with the students who have a group with their friends, the remaining students will be assigned randomly by instructor/TAs. If there is a need of changing a student from one group to another, it will also be done by the instructor/TAs. Instructor/TAs can display statistics (compare/rank/sort students' peer review grades).

2.1.2 Students

When students sign in to the application, they will submit their groups to the instructor/TA so that they can form a group with their friends. Students can withdraw/drop the course. Students can see their information card which includes the student's personal profile.

2.2 Functional Requirements

2.2.1 Sign Up and Login

As the system will be on track with the course, both the instructor and students will have to sign up at the beginning of the semester and login each time accessing the server. The system launches with the instructor logging in as "host". After the launch, every student will have to sign up with their Bilkent mail address and then keep in touch with the system throughout the system for artifacts and peer reviews. Initially, after signing up, all students will join the "without group member" category, waiting for the host to assign them to groups.

2.2.2 Manage project groups

This section will cover most of the host's functions as the instructor will only review artifacts of groups after forming groups.

2.2.2.1 Create a group

At the beginning of the semester, the host will open the particular course and section's server and the first thing he will do is to form groups. S/he will decide how many students will a single group include at the maximum and also how many groups will be created considering the number of the course's attendants.

2.2.2.2 Assign student to a group

After all students are signed up and available for the system, the host assigns each student one-by-one to the group that they desire to be in. The students make lists to notify the instructor about the groups they formed themselves and the host records their user accounts into the groups s/he formed earlier.

2.2.2.3 Delete group

If some students drop the course during the add/drop period and the group count becomes too much, the host can remove a group from the system to overcome conflicts. Also, after the semester ends, the host can use this function in order to adjust for the next semester. Therefore, the instructor can reuse it thereafter.

2.2.2.4 Change a student's group

This is a special case as if after some portion of students give a list to the instructor and the group is formed and one of the students changes his/her mind and wants to be involved in another group. In this case, the host removes that student from the current group and enrolls the user into the desired one.

2.2.3 View an Information Card

This function will be an only-review page for the host in order to see the review scores of a particular student. For the user(students), this will be like a profile section in which they will view their own scores or edit main info about their profile. They can also view their peers information cards to gather some knowledge about them, too.

2.2.3.1 Edit User Information

On the information card page, this feature will appear for the students if they would like to change their profile photo or some general information about their profile. The changes made will appear on the information card immediately.

2.2.3.2 Withdraw or Drop Course

This is a special case for the students about the accessibility of group-student interactions. When a student drops or withdraws the course, the change will be made by the system accordingly, that the account of the student will be removed from the group, and the system.

2.2.4 Review Panels

The functions under this section is the cornerstone of the whole system, both the user and the students will be able to see all students' review scores, artifacts of other groups, and feedback messages from their peers. In addition to all of these review functions, the students will be able to give their peers review scores and feedback comments to other groups' artifacts.

2.2.4.1 View Groups List

Students and the host can see all groups in a list format to show their own or the other groups so that they can review the scores and give feedback to others.

2.2.4.2 Select User Group

When a student selects his/her own group in the list, s/he can see the members' scores and sort them in order of their scores.

2.2.4.2.1 Peer Review

When they select the peer review button, they first decide which scale to be reviewed. There are 3 different review scales which are punctuality, work, and communication. Next, they can enter scores from 1 to 10 for the selected scale and give written feedback about their contribution or edit some of them if they change their mind. In order to help the students grade each other and the host for asking the right questions, some template questions will be given to the host while creating them and also the scales will be given to the students with a neat interface.

2.2.4.3 Select Other Groups

If the user selects one of the groups that s/he doesn't belong, the user can see the particular group's members in a list format, click on them to view their information cards, and also see about the reports and works they have done in order to have an idea about their work, give feedback or maybe take some hints to get inspiration for their projects.

2.2.4.3.1 Review Others' Artifacts

Host or students select the group number then the artifact to be reviewed. After writing the review about the artifact, the reviewer can submit his/her comment regarding the artifact.

2.3 Non-Functional Requirements

2.3.1 User interface and Human Factors

The Peer Review system is a web application; thus, this application would be done to accommodate screens with different resolutions. Each button would be named as its function in order for users to navigate on the system.

2.3.2 Documentation

There is going to be several reports throughout the development of the peer review system that explain analysis, structure, etc. There will be a help button at the top-right corner. This button shows brief instructions for students to grade their peers and this button may include a video about how to use that user interface.

2.3.3 Hardware Considerations

The Peer Review System is a quite simple program that any computer that has an internet connection, and any operating system can run our peer review system. Users need to have a keyboard to write down their comments, and a mouse to control the program. Thus, a keyboard and mouse are required to use our peer review system.

2.3.4 Performance Characteristics

In the Peer Review System, all students have five different grades and comments, and all projects can be commented by others groups, teaching assistants, and instructors. Although all comments and information would be stored, because of the limited number of student size and number of projects (not more than 175 users), there would not be a mass of comments and information.

Moreover, the peer review system will be displayed on a single system for each user. Those two features make the system executable without any noticeable delay, which means a response takes maximum 0.25 milliseconds. Networking performance requirements are not needed to be specified.

There is no limitation on submission size because github links will be used in our system.

2.3.5 Error Handling and Extreme Conditions

Since the peer review system developed in PHP, we make use of try-catch blocks in the case of possible errors that might occur.

2.3.6 System Interfacing

Users will navigate in the project via mouse by clicking buttons. Occasionally, users will be using their keyboards for just typing. They can use all English letters, numbers, and symbols. Comments and grades will be directly forward to the database, that's because there would be no timed execution.

2.3.7 System Modifications

The Peer Review System does not require any particular system modifications. At the end of the semester teaching assistants or instructors can delete all data and remove the course from the database. Then, the instructor can start a course again for the next semester.

2.3.8 Physical Environment

The Peer Review System can take place for each students', teaching assistants' and instructor's computer. Therefore, each user has their own physical environment to use our system they can reach to the system via internet connection.

2.3.9 Security Issues

Students sign up to the system with their email that has ug.bilkent.edu.tr extension. Atomically, a verification mail would be sent to students' mail addresses. If they can type the code correctly, they can enroll in the system. By means of e-mail verification, we protect our system from unwilling attempts.

2.3.10 Usability

Teaching assistant can create an empty group. But when all students are removed the group will be deleted automatically.

If a user types an invalid email or password, the system automatically asks the user to correct email and password by refreshing the page right after the warning message.

2.3.11 Compatibility

Out peer review system will accept only github link as submission, so there is no specific type for submission

2.4 Pseudo Requirements

2.4.1 Language

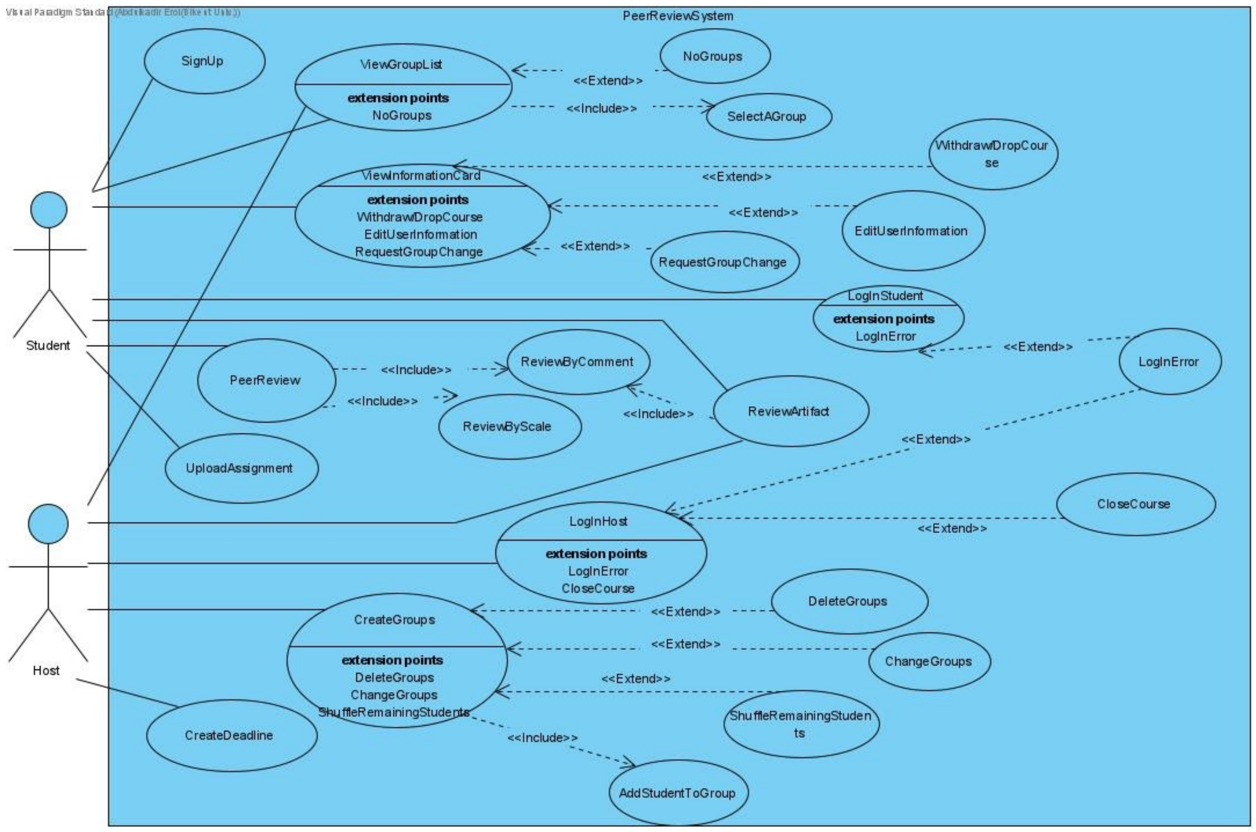
The backend of the peer review system would be developed in PHP which is an object-oriented programming language. The frontend will be implemented in HTML and CSS. PhpMyAdmin will be used as a database.

2.4.2 Version Control System

GitHub is going to be used as the version control system and the code repository of the peer review system.

2.5 System Models

2.5.1 Use Case Models



Use Case Name	SignUp
Participating Actor	Launched by Student
Flow of Events	1. Student enters the website. 2. Student enters his/her information to the signup form. 3. Student authenticates his/her account via email.
Entry Condition	This use case extends **NoGroups** use case. It is initiated by clicking to the “Sign-up” button.
Exit Condition	Closing the webpage. Clicking the “Submit” button.
Quality Requirements	1. It can be used only in the beginning panel. 2. E-mail address should be valid.

Use Case Name	LogInStudent
Participating Actor	Launched by Student
Flow of Events	1. The student writes his/her email and his/her password on the textboxes. 2. S/he presses the “Login” button.
Entry Condition	Student presses the “Login” button. This use case extends the **LogInError** use case. It is initiated by the system whenever the student enters a wrong email or wrong password.
Exit Condition	Closing the webpage. Clicking the “Login” button.
Quality Requirements	1. It can be used only in the beginning panel. 2. The given email must have been used for signup before. 3. The given email and password must match.

Use Case Name	LogInHost
Participating Actor	Launched by Host
Flow of Events	1. The host writes his/her email and his/her password on the textboxes. 2. S/he presses the “Login” button. 3. Host enters the peer review system.
Entry Condition	Host presses the “Login” button. This use case extends the **LogInError** and **CloseCourse** use cases. It is initiated by the system whenever the host enters a wrong email or wrong password. This use case extends to the CloseCourse use case. It is initiated by the system whenever the host closes the course at the end of the semester.
Exit Condition	Closing the webpage. Clicking the “Login” button.
Quality Requirements	1. It can be used only in the beginning panel. 2. The given email and password must match.

Use Case Name	CreateGroups
Participating Actor	Launched by Host
Flow of Events	1. Actor enters the system by logging in. 2. Depending on the number of students, the host creates an adequate number of groups. 3. Instructor then adds students to the groups depending on the group size.
Entry Condition	Logged in to the system with the appropriate role. This use case extends **DeleteGroups** use case. It is initiated by the system whenever the host deletes one of the groups. This use case extends **ChangeGrops** use case. It is initiated by the system whenever the host changes the group of a particular student. This use case extends **shuffleRandomgroups** use case. It is initiated by the system whenever the host shuffles the students without a group to random groups. This use case includes the **AddStudentToGroup** use case.
Exit Condition	Closing the webpage. Clicking the “Create Groups” button
Quality Requirements	1. The group size is determined by the host each semester. 2. This use case includes the **AddStudentsToGroup** use case.

Use Case Name	ViewInformationCard
Participating Actor	Launched by Student
Flow of Events	1. Students enter the system by logging in. 2. A student can see his/her own information card. 3. After inspection the student closes his/her information card.
Entry Condition	The actor must be logged into the system. This use case extends **EditUserInformation** use case. It is initiated by the system whenever the actor changes his/her information such as e-mail, name etc. This use case extends **Withdraw/DropCourse** use case. It is initiated by the system whenever one of the students drops the course or withdraws it. This use case extends **RequestGroupChange** use case. It is initiated by the system whenever one of the students requests his/her group to be changed by the host.
Exit Condition	It is exited whenever the exit request is committed.
Quality Requirements	

Use Case Name	ViewInformationCard
Participating Actor	Launched by Host
Flow of Events	1. Actor enters the system by logging in. 2. Host can see the information card of each student. 3. Host chooses one of the information cards. 4. After inspection the actor closes the information card.
Entry Condition	The actor must be logged into the system. This use case extends **EditUserInformation** use case. It is initiated by the system whenever the student changes his/her information such as e-mail, name, etc. This use case extends **Withdraw/DropCourse** use case. It is initiated by the system whenever one of the students drops the course or withdraws it.
Exit Condition	The actor has inspected the information card.
Quality Requirements	

Use Case Name	LogInError
Participating Actor	Launched by Student / Host
Flow of Events	1. Actor enters his/her login information to the login page. 2. An error message is displayed indicating that something went wrong.
Entry Condition	Actors try to login to the website
Exit Condition	Actors cannot login to the site.
Quality Requirements	

Use Case Name	CloseCourse
Participating Actor	Launched by Host
Flow of Events	1. Host enters the website. 2. Host presses the **CloseCourse** button. 3. All the groups will be deleted by the database.
Entry Condition	The semester should end.
Exit Condition	All the groups are deleted.
Quality Requirements	

Use Case Name	DeleteGroups
Participating Actor	Launched by Host
Flow of Events	1. Host enters the website. 2. Host presses the **DeleteGroups** button and specify which groups will be deleted. 3. Selected groups will be deleted and the students in the deleted groups will be assigned the “without a group” category.
Entry Condition	Clicking the “Delete Group” button.
Exit Condition	One or more of the groups are deleted.
Quality Requirements	

Use Case Name	ChangeGroups
Participating Actor	Launched by Host
Flow of Events	1. Host enters the website. 2. Host presses the ChangeGroups button and specifies the student’s new group number. 3. Selected students will be assigned to the new group and will be removed from the previous group.
Entry Condition	Students notify the host by clicking the “Change Group Request” button.
Exit Condition	Host clicks the “Change Group” button.
Quality Requirements	

Use Case Name	AddStudentsToGroups
Participating Actor	Launched by Host
Flow of Events	1. Host enters to the website 2. Host enters the CreateGroups use case 3. After creating a new group, host assign the students to groups
Entry Condition	There should be a group with an empty place for a student. There should be students in the “without a group” category.
Exit Condition	Host clicks the “Add Student to Group” button.
Quality Requirements	

Use Case Name	UploadAssignment
Participating Actor	Launched by Student
Flow of Events	1. Student enters the uploading page. 2. Student enters the name of the artifact and the GitHub link. 3. Student presses the submit button.
Entry Condition	Students press the “Upload Artifact” button in the information card menu.
Exit Condition	Student submits the artifact.
Quality Requirements	The artifact must be uploaded before the deadline associated with that artifact.

Use Case Name	Create Deadline
Participating Actor	Launched by Host
Flow of Events	1. Host creates a deadline for a particular artifact. 2. Host submits the deadline.
Entry Condition	Host press the “Create Deadline” button in the host main menu.
Exit Condition	Host submits the created deadline for the artifact.
Quality Requirements	1. Each artifact must have its own deadline.

Use Case Name	shuffleRemainingStudents
Participating Actor	Launched by Host
Flow of Events	1. Host enters the groups page. 2. Host presses the “shuffle students” button.
Entry Condition	Host presses the “Group List” button in the host main menu.
Exit Condition	Host shuffles the groups.
Quality Requirements	1. Email can be changed if the email is not used by other students. 2. The student must verify the old password by writing it again.

Use Case Name	Withdraw/DropCourse
Participating Actor	Launched by Student
Flow of Events	1. The student presses the “Withdraw/Drop Course” button to delete his/her account. 2. The student will be deleted from the system and that student’s group information will be updated.
Entry Condition	Student presses “Withdraw/Drop Course” button in the information card menu.
Exit Condition	Student has dropped or withdrawn from the course.
Quality Requirements	1. If a student withdraws or drops the course, the student account will be deleted from the system and the student can not enter the system with his/her account.

Use Case Name	ViewGroupList
Participating Actor	Launched by Student
Flow of Events	1. Student presses the “View Groups” button. 2. All of the groups and the group that student is in will be shown.
Entry Condition	After the student logs in their account they press the “View Groups” button.
Exit Condition	The student selects a group
Quality Requirements	1. Students can see the other groups when the groups are created by the host. 2. It includes **SelectAGroup** use case

Use Case Name	NoGroups
Participating Actor	Launched by System
Flow of Events	1. No group message is displayed.
Entry Condition	This case starts if and only if the instructor hasn’t created any groups.
Exit Condition	It exits if the person chooses the press “Exit” button.
Quality Requirements	1. There must be no groups.

Use Case Name	ReviewArtifact
Participating Actor	Launched by Student / Host
Flow of Events	1. Student can see the other groups' artifacts. 2. Student / Host reviews the artifact of a group 3. Student / Host submits the review
Entry Condition	Actor presses “Review Artifact” button
Exit Condition	Actor submits the review
Quality Requirements	1. The reviews will be done after artifact deadlines 2. It includes **ReviewByComment**.

Use Case Name	SelectAGroup
Participating Actor	Launched by student
Flow of Events	1. Student selects a group from group list
Entry Condition	This use case extends **NoGroups** use case. It is initiated by the system whenever there no group to choose.
Exit Condition	The student selects a group
Quality Requirements	

Use Case Name	PeerReview
Participating Actor	Launched by student
Flow of Events	1. Student can reach their review by pressing the “Review by Comment” and “Review by Scale” button.
Entry Condition	Clicking “Peer Review” button.
Exit Condition	It exits if the person chooses the press “Exit” button. It will return to selection of the group screen.
Quality Requirements	

Use Case Name	ReviewByComment
Participating Actor	Inherited from PeerReview use case
Flow of Events	1. Student can comment on the artifact of the other groups. 2. Student can comment only on their group members.
Entry Condition	Clicking “Review by Comment” button.
Exit Condition	Clicking “Submit” button. It will return to selection of the group screen.
Quality Requirements	1. Both of the artifacts of other groups and group members can be commented by student.

Use Case Name	ReviewByScale
Participating Actor	Inherited from PeerReview use case
Flow of Events	1. Students will select their group members. 2. Student will scale their group member grade from 1 to 10 3. Students will press the "Review" button.
Entry Condition	Clicking "Review by Scale" button.
Exit Condition	Clicking "Submit" button. It will return to selection of the group screen.
Quality Requirements	1. Students will review their friends only once.

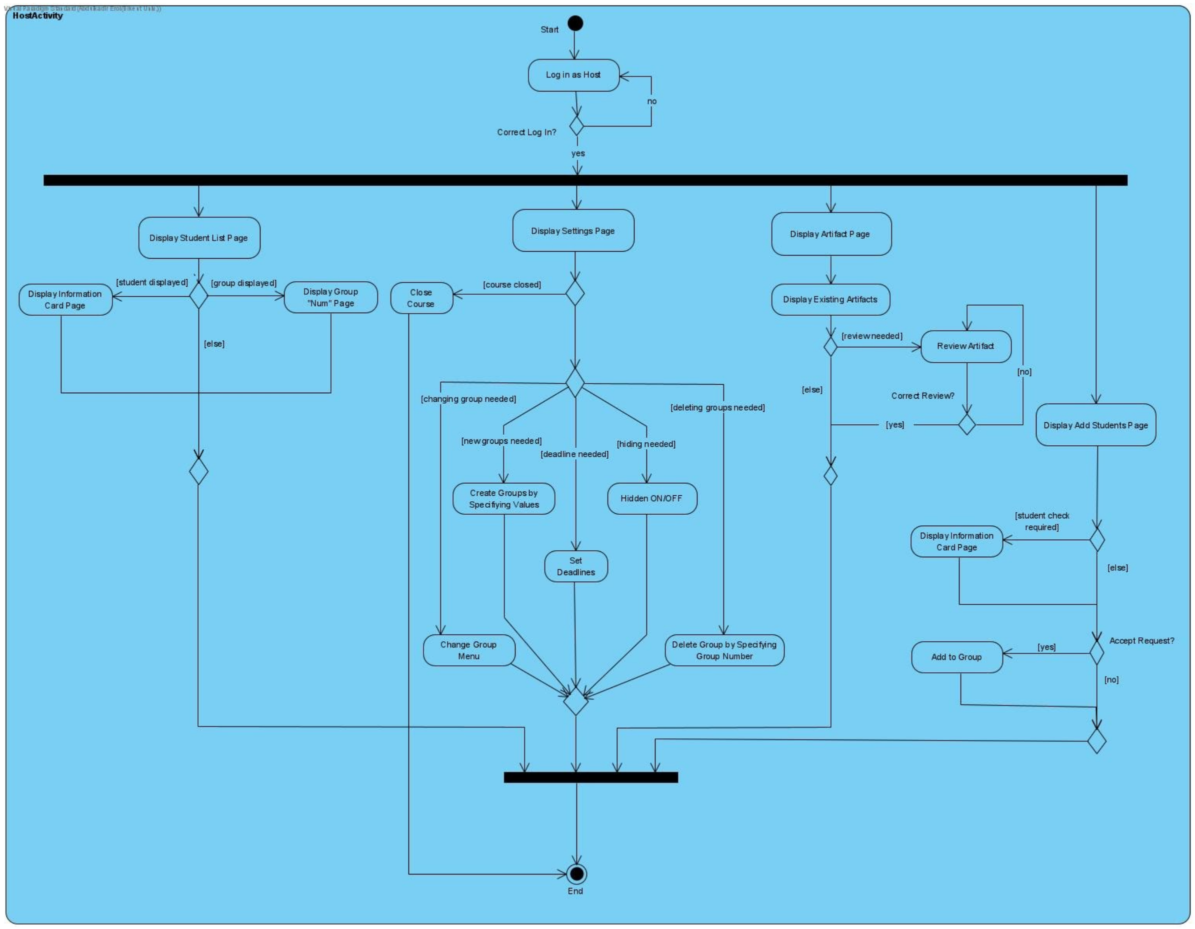
Use Case Name	EditUserInformation
Participating Actor	Launched by Student
Flow of Events	Students can fill the places with the email, old password and new password. Students can delete/change their photos. When the user presses the “Update” button, the system will change his/her id and password.
Entry Condition	Students press the “Edit Information” button in the information card menu.
Exit Condition	Student submits the updated information.
Quality Requirements	Email can be changed if the email is not used by other students. The student must verify the old password by writing it again.

Use Case Name	ShuffleRemainingStudents
Participating Actor	Launched by Host
Flow of Events	1. Host presses the shuffle students button to assign students without groups to random groups.
Entry Condition	Host presses the “Group List” button on the host main menu.
Exit Condition	Host presses the “Shuffle Students” button on the host main menu.
Quality Requirements	The maximum number of students in the group will not be exceeded.

2.5.2 Dynamic Models

2.5.2.1 Activity Diagrams

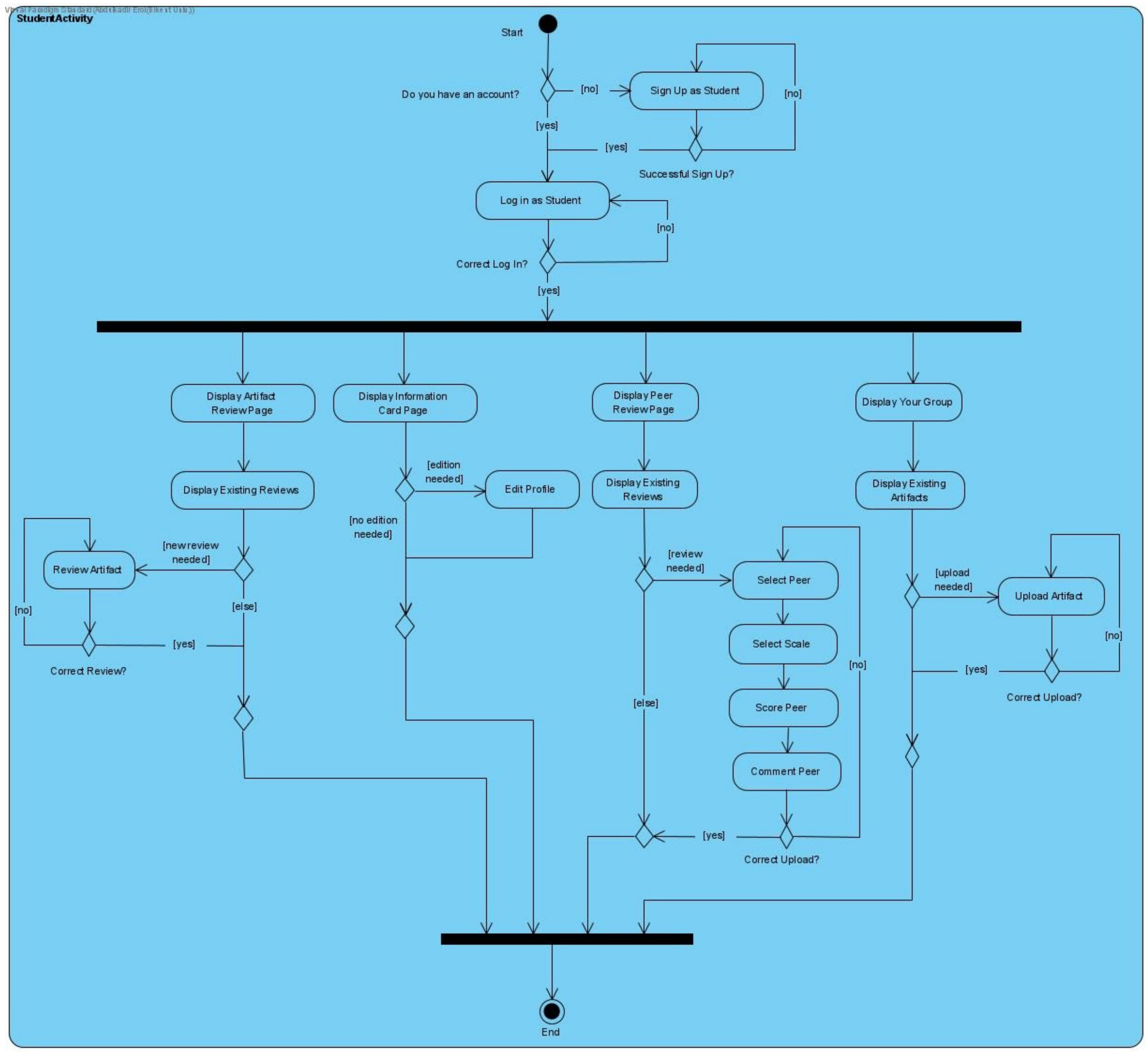
2.5.2.1.1 Host Activity Diagram



Host Activity Diagram shows the process of the Host. At the beginning, Host must have logged in to the system correctly. Then, Host can choose from different options such as Display Student List Page, Display Settings Page, Display Artifact Page, and Display Add Students Page. In the Display Student List Page, Host will go to the Display Information Card Page, Display Group "Num" Page or continue without doing anything according to the state requirements. In the Display Settings Page, if Course Closed state occurs, Close Course action will be executed and activity diagram will close. Otherwise, there are some options that Host will proceed such as Change Group Menu, Create Groups by Specifying Values, Set Deadlines, Hidden ON/OFF, Delete Groups by Specifying

Group Number according to states. Then, it exits. In Display Artifact Page, Host Displays Existing Artifacts. If a review is needed, Host goes to the Review Artifact. If there is a correct review, it exists. Otherwise, another Review Artifact is required. In Display Add Students, if a student check is required Host Display Information Card Page. Then, if Host accepts, a student will be added to the group. Then it exits.

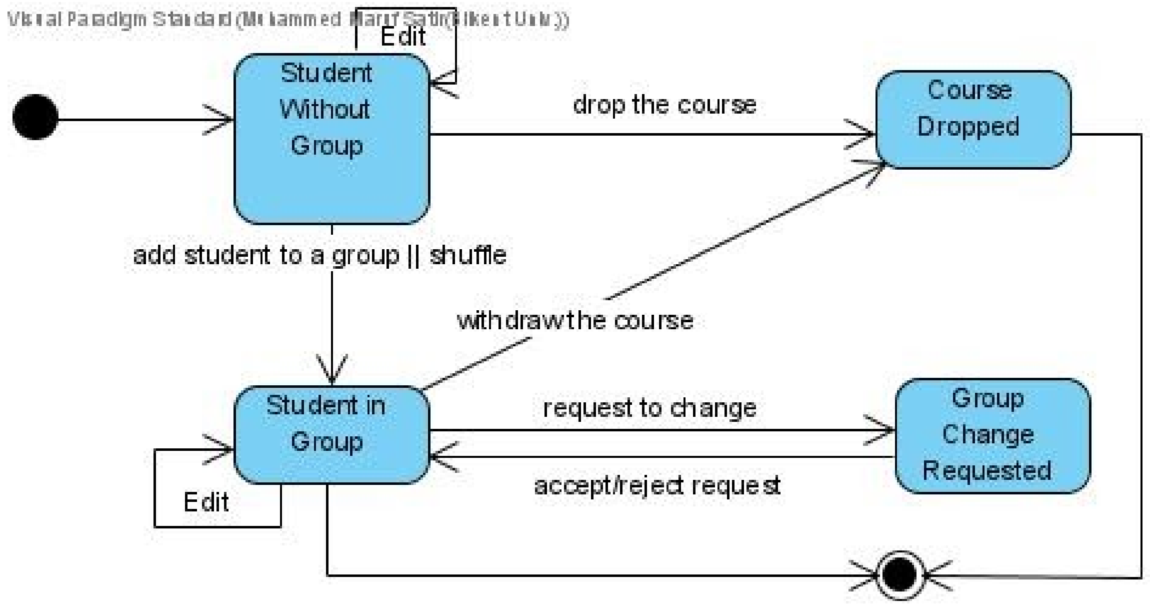
2.5.2.1.1 Student Activity Diagram



Student Activity Diagram shows the process of the students. At the beginning, students should have an account and must have logged in to the system correctly with this account. Then, students can choose from different options such as Display Artifact Review Page, Display Information Card Page, Display Peer Review Page and Display Your Group Page. In the Display Artifact Review Page, students Displays Existing Reviews. If a new review is needed, the student goes to the Review Artifact and reviews the artifact. If there is a correct review, it continues. In the Display Information Card Page, students edit their profile if edition needed. Otherwise, there is not a necessary action. In the Display Peer Review Page, students Displays Existing Reviews. If a new review is needed, students Select Peer, Select Scale, Score Peer, and Comment Peers. If there is a correct upload, it continues. In Display Your Group Page, students Display Existing Artifacts. Then, if upload needed, students Upload Artifact. If there is a correct upload, it continues.

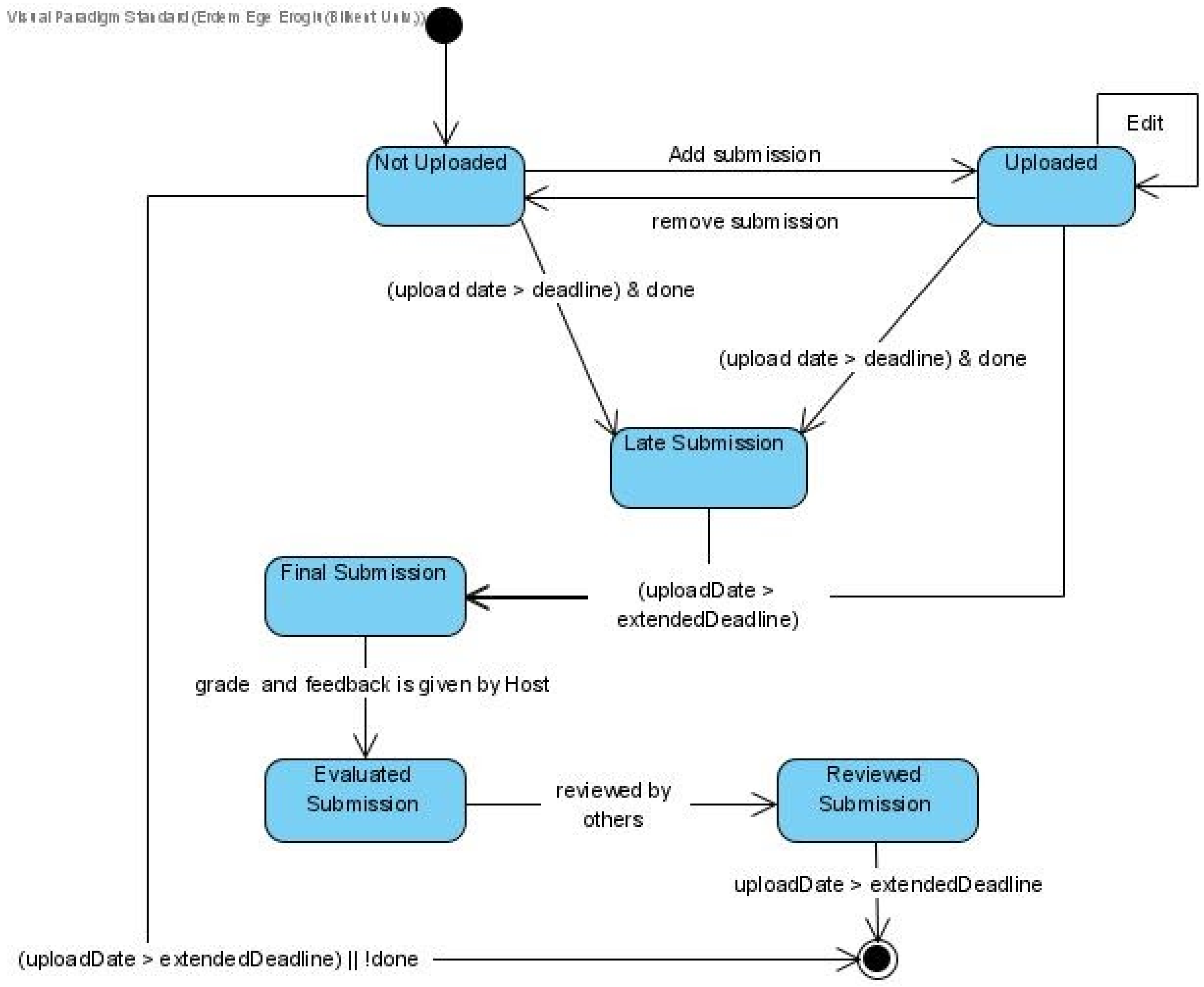
2.5.2.2 State Machine Diagram

2.5.2.2.1 Student State Diagram



Student at the state of Student Without Group may drop the course and it will cause the student to pass to the Course Dropped state and finally the final state. If the student is added to a certain group or host shuffled the students without groups it will lead the student to pass to the state of Student in Group. Editing information card for the states of Student Without Group and Student in Group states will not cause the changes of the states. The request to change the group will lead to change in state. It turns to the state of the Student in Group when the host responds to a request for change.

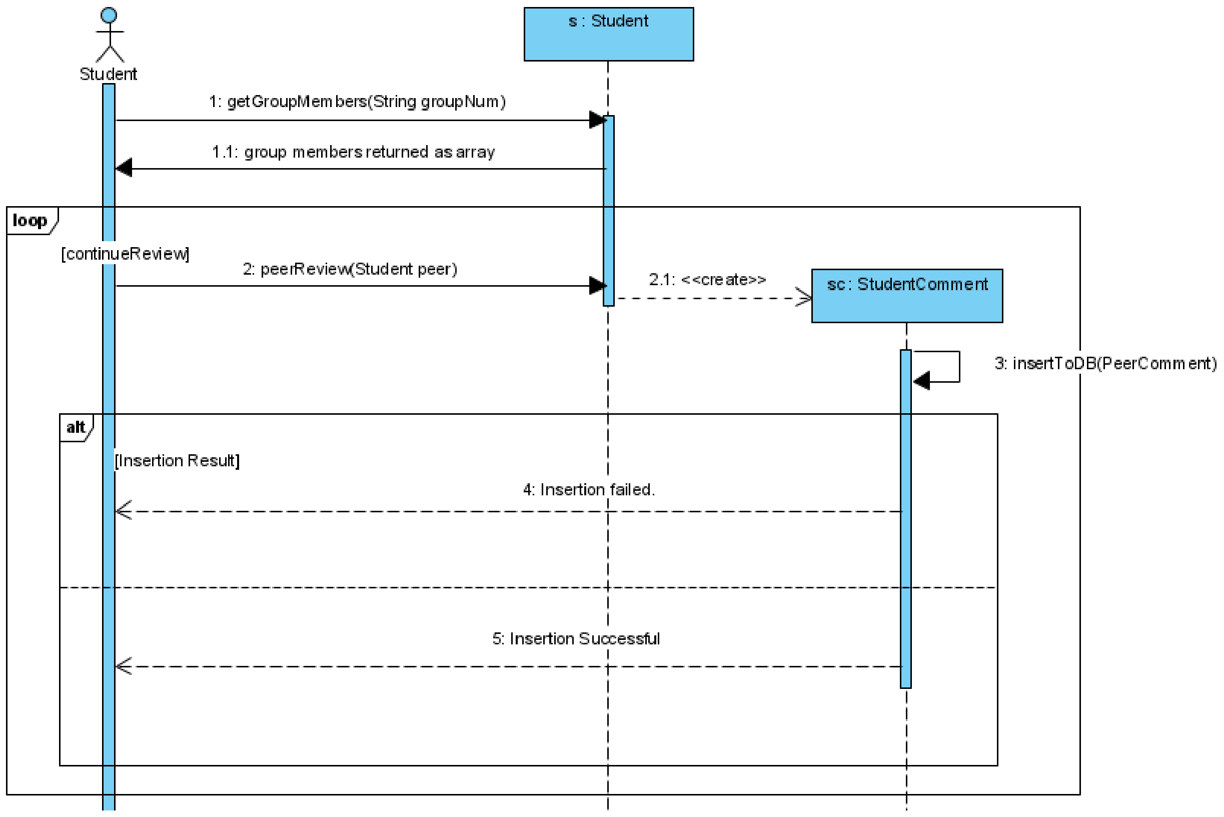
2.5.2.2.2 Submission State Diagram


Visual Paradigm Standard(Erdem Ege Eroglu(Bilkent Univ.))
Not Uploaded
Add submission
Uploaded
Edit
remove submission
(upload date > deadline) & done
(upload date > deadline) & done
Late Submission
Final Submission
(uploadDate > extendedDeadline)
grade and feedback is given by Host
Evaluated Submission
reviewed by others
Reviewed Submission
uploadDate > extendedDeadline
(uploadDate > extendedDeadline) || !done

State machine diagram of submission starts with “not uploaded” state. If there is no submission until the end of extended deadline date it goes to finish. When a student makes a submission the state followed by upload or late submission states, it depends on upload date. In upload state, submission can be removed or edited. We can reach final submission at the end of extended deadline if the submission is done. Then, when submission is graded and feedback is given by the host, the state goes to “evaluated submission”. After that, other students from different groups make reviews and the current state is now “reviewed submission”. Eventually, the state reaches the end at the end of the extended deadline.

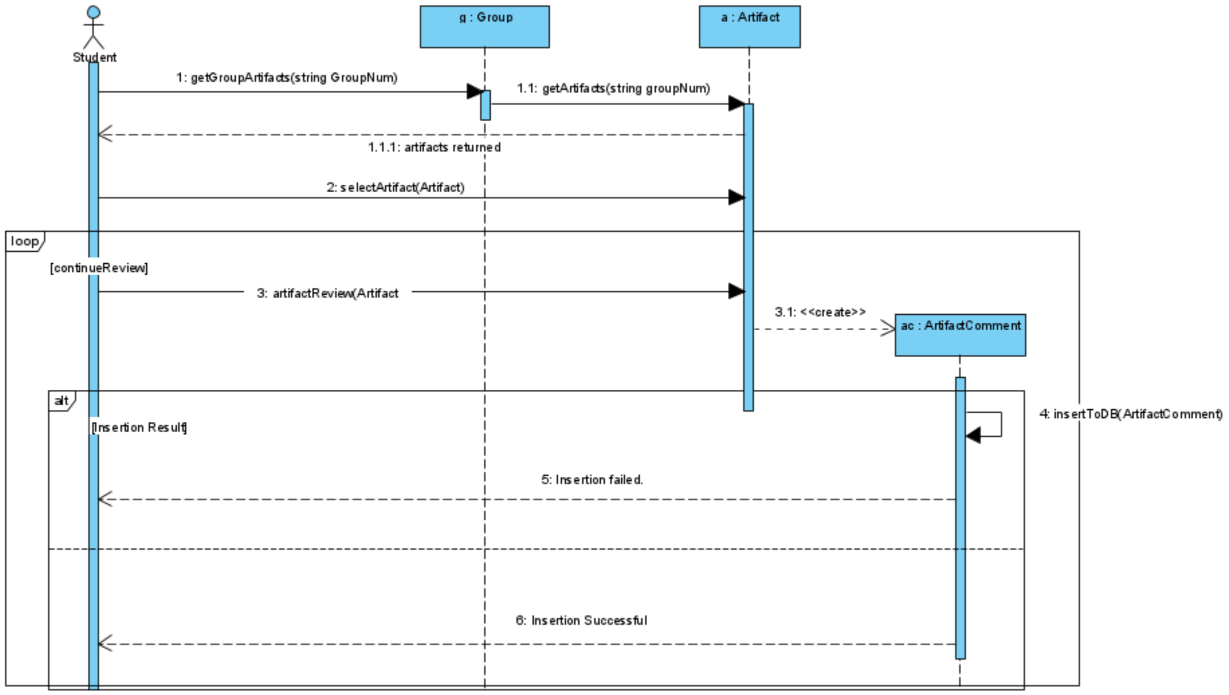
2.5.2.3 Sequence Diagrams

2.5.2.3.1 Peer Review Sequence Diagram



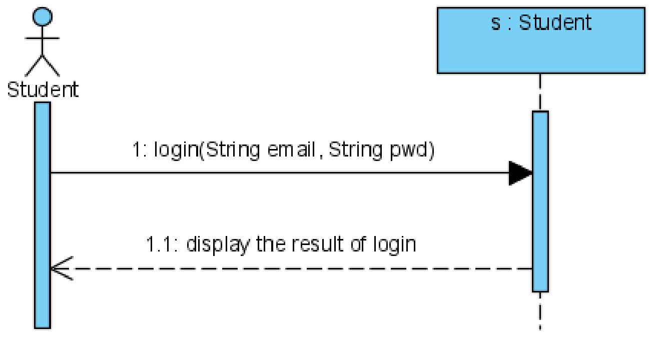
When the student enters the peer review page, the system will return the group members using the group number as a parameter. Then the student selects a peer and writes comments about him/her and submits the review. This will create a new PeerComment object and this will be inserted into the database so that it can be referred afterwards.

2.5.2.3.2 Artifact Review Sequence Diagram


Student
g : Group
a : Artifact
1: getGroupArtifacts(string GroupNum)
1.1: getArtifacts(string groupNum)
1.1.1: artifacts returned
2: selectArtifact(Artifact)
loop
[continueReview]
3: artifactReview(Artifact
3.1: <<create>>
ac : ArtifactComment
alt
[Insertion Result]
4: insertToDB(ArtifactComment)
5: Insertion failed.
6: Insertion Successful

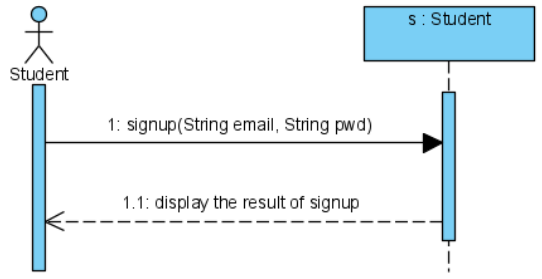
When the student enters the artifact review page, the system will return the group artifacts using the group number as a parameter. Then the student selects an artifact and writes comments about the artifact and submits the review. This will create a new ArtifactComment object and this will be inserted into the database so that it can be referred afterwards.

2.5.2.3.3 Login Sequence Diagram


Student
s : Student
1: login(String email, String pwd)
1.1: display the result of login

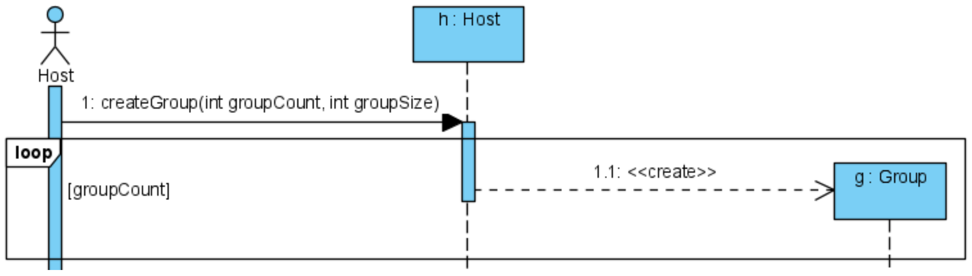
Students login to the application by using a method inherited from account class via providing a valid email and password combination.

2.5.2.3.4 Sign-up Sequence Diagram


Student
s : Student
1: signup(String email, String pwd)
1.1: display the result of signup

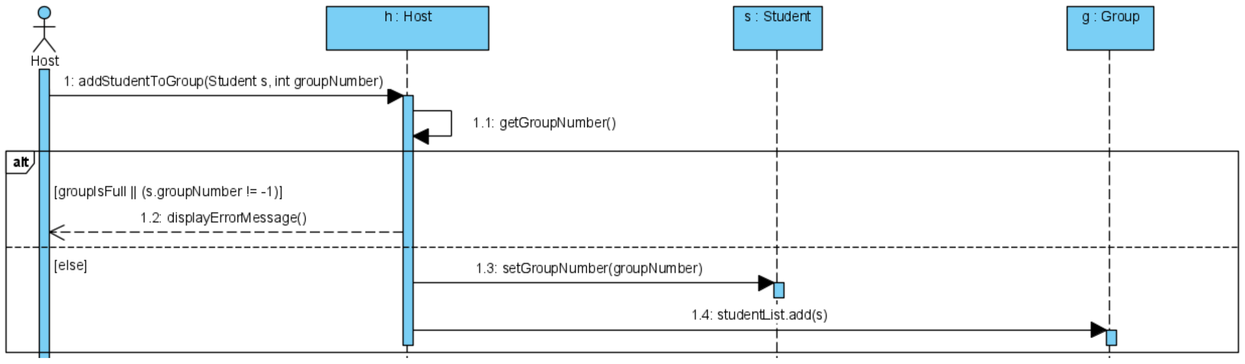
Students sign to the application by using a method inherited from account class via providing a valid email and password combination.

2.5.2.3.5 Create Groups Sequence Diagram



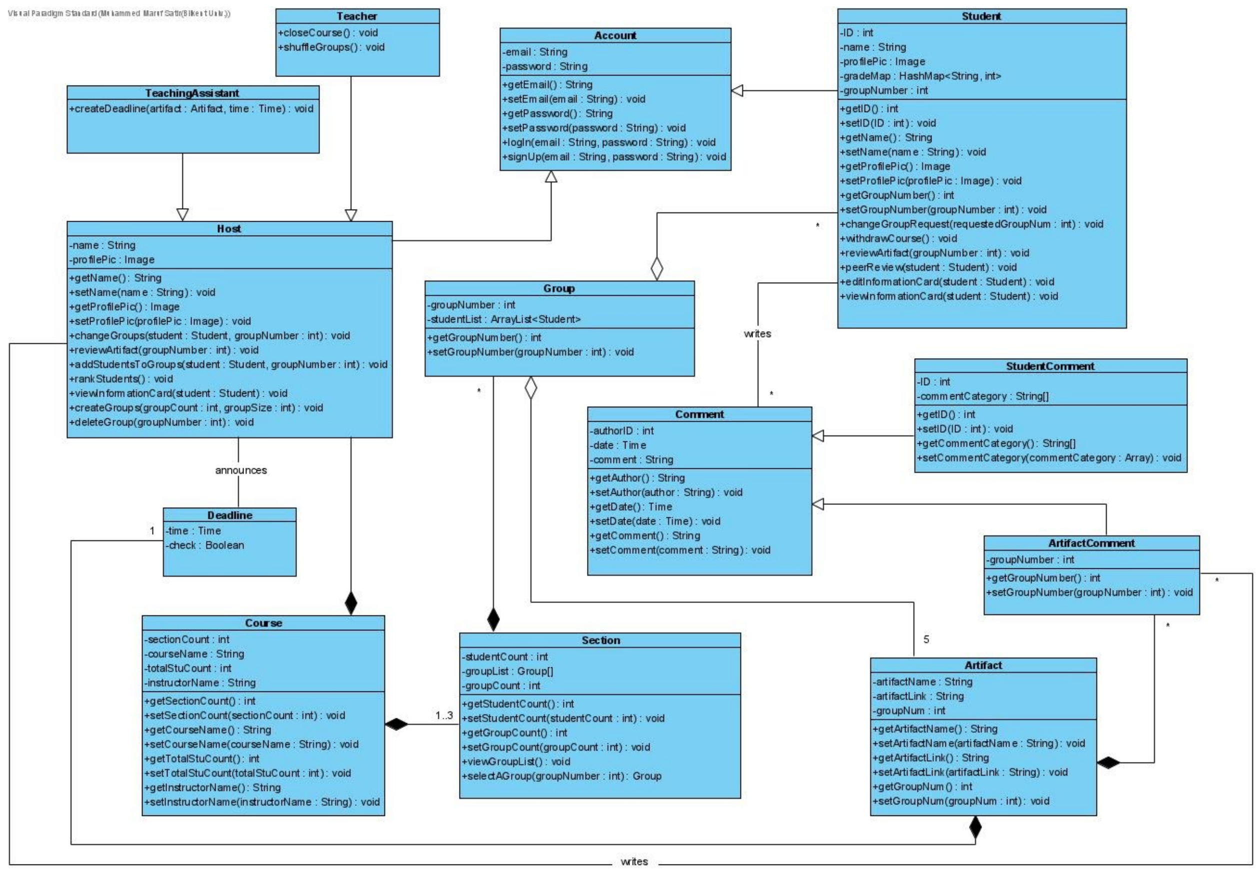
At the beginning of the semester, the host can create as many groups as s/he wants using the method from the Host class.

2.5.2.3.6 Add Student to Group Sequence Diagram


Host
h : Host
s : Student
g : Group
1: addStudentToGroup(Student s, int groupNumber)
1.1: getGroupNumber()
alt
[groupIsFull || (s.groupNumber != -1)]
1.2: displayErrorMessage()
[else]
1.3: setGroupNumber(groupNumber)
1.4: studentList.add(s)

Host can add any student which is not currently in a group to the desired group by the method in class Host.

2.5.3 Object and Class Models


Visual Paradigm Standard(Muhammed Maruf Satir(Bilkent Univ.))
Teacher
+closeCourse() : void
+shuffleGroups() : void
Account
-email : String
-password : String
+getEmail() : String
+setEmail(email : String) : void
+getPassword() : String
+setPassword(password : String) : void
+logIn(email : String, password : String) : void
+signUp(email : String, password : String) : void
Student
-ID : int
-name : String
-profilePic : Image
-gradeMap : HashMap<String, int>
-groupNumber : int
+getID() : int
+setID(ID : int) : void
+getName() : String
+setName(name : String) : void
+getProfilePic() : Image
+setProfilePic(profilePic : Image) : void
+getGroupNumber() : int
+setGroupNumber(groupNumber : int) : void
+changeGroupRequest(requestedGroupNum : int) : void
+withdrawCourse() : void
+reviewArtifact(groupNumber : int) : void
+peerReview(student : Student) : void
+editInformationCard(student : Student) : void
+viewInformationCard(student : Student) : void
TeachingAssistant
+createDeadline(artifact : Artifact, time : Time) : void
Host
-name : String
-profilePic : Image
+getName() : String
+setName(name : String) : void
+getProfilePic() : Image
+setProfilePic(profilePic : Image) : void
+changeGroups(student : Student, groupNumber : int) : void
+reviewArtifact(groupNumber : int) : void
+addStudentsToGroups(student : Student, groupNumber : int) : void
+rankStudents() : void
+viewInformationCard(student : Student) : void
+createGroups(groupCount : int, groupSize : int) : void
+deleteGroup(groupNumber : int) : void
Group
-groupNumber : int
-studentList : ArrayList<Student>
+getGroupNumber() : int
+setGroupNumber(groupNumber : int) : void
*
writes
*
StudentComment
-ID : int
-commentCategory : String[]
+getID() : int
+setID(ID : int) : void
+getCommentCategory() : String[]
+setCommentCategory(commentCategory : Array) : void
Comment
-authorID : int
-date : Time
-comment : String
+getAuthor() : String
+setAuthor(author : String) : void
+getDate() : Time
+setDate(date : Time) : void
+getComment() : String
+setComment(comment : String) : void
announces
Deadline
1
-time : Time
-check : Boolean
ArtifactComment
-groupNumber : int
+getGroupNumber() : int
+setGroupNumber(groupNumber : int) : void
*
*
*
Course
-sectionCount : int
-courseName : String
-totalStuCount : int
-instructorName : String
+getSectionCount() : int
+setSectionCount(sectionCount : int) : void
+getCourseName() : String
+setCourseName(courseName : String) : void
+getTotalStuCount() : int
+setTotalStuCount(totalStuCount : int) : void
+getInstructorName() : String
+setInstructorName(instructorName : String) : void
1..3
Section
-studentCount : int
-groupList : Group[]
-groupCount : int
+getStudentCount() : int
+setStudentCount(studentCount : int) : void
+getGroupCount() : int
+setGroupCount(groupCount : int) : void
+viewGroupList() : void
+selectAGroup(groupNumber : int) : Group
5
Artifact
-artifactName : String
-artifactLink : String
-groupNum : int
+getArtifactName() : String
+setArtifactName(artifactName : String) : void
+getArtifactLink() : String
+setArtifactLink(artifactLink : String) : void
+getGroupNum() : int
+setGroupNum(groupNum : int) : void
writes

Class Name	Host
Properties	· String name: holds the name of the instructor/TA · Image profilePic: holds the photo of the instructor/TA
Methods	· String getName(): returns the name of the instructor/TA · void setName(String name): sets the name of the instructor/TA to the parameter · Image getProfilePic(): returns the photo of the instructor/TA · void setProfilePic(Image profilePic): sets the photo of the instructor/TA to the parameter · void closeCourse(): terminates the course, the records of the groups and the students · void deleteGroup(int groupNumber): removes the group with the index of the parameter from the course · void createGroups(int groupCount, int groupSize): creates student groups as much as the parameter and sets the maximum capacity of each group to the second parameter · void shuffleGroups(): randomly assigns the students who aren't distributed in any groups yet · void changeGroups(Student student, int groupNumber): changes the group of the student in the parameter into the group in the parameter · void reviewArtifact(int groupNumber): displays the work done by the group of the number in the parameter · void addStudentToGroup(Student student, int groupNumber): adds the student in the parameter to the group whose index is in the second parameter unless the group size is full

Class Name	Account
Properties	· String email: holds the email address of the account · String password: hold the password information of the account
Methods	· String getEmail (): returns the email address of the account · void setEmail (String email): sets the email address of the account · String getPassword (): returns the password of the account · void setPassword (String password): sets the password of the account

Class Name	Comment
Properties	· String comment: holds the comment part of the Comment object · Time date: holds the date when the comment is submitted · int authorID: holds the author of the comment
Methods	· String getComment (): returns the comment part of the Comment object · void setComment (String comment): sets the comment part of the Comment object · Time getDate (): returns the submission date of the Comment object · void setdate (Time date): sets the submission date of the Comment object · String getAuthor (): returns the author of the Comment object · void setAuthor (String author): sets the author of the Comment object

Class Name	Group
Properties	· int groupNumber: holds the group number of the Group Object · ArrayList<Student> studentList: holds the members of the group as an ArrayList
Methods	· int getGroupNumber (): returns the groupNumber parameter of the Group object · void setGroupNumber (int groupNumber): sets the group number of Group object.

Class Name	Student
Property	· int ID: Holds the ID number of a student. · String name: Holds the name of the student. · Image profilePic: Holds the image of the profile picture of the student. · Hashmap<String, int> gradeMap: Holds the grade of a particular student on the student. · int groupNumber: Holds the group number that the student is in.
Constructor	Student(string name, int id): setName and setID to its parameters. setProfilePic set to null.

Methods	· int getID(): returns the ID of the student. · void setID(int ID): sets the ID of the player. · String getName(): returns the name of the student. · void setName(String name): sets the name of the student. · Image getProfilePic(): returns the image of the student. · void setProfilePic(Image image): sets the image of the student. · int getGroupNumber(): return the group number that student had been included. · void setGroupNumber(int groupNumber): sets the group number of the student. · void changeGroupRequest(int requestedGroupNum): student sends request to the host by indicating the requested group’s number. · void withdrawCourse(): student requests the withdrawal of the system and system deletes his/her account from the system. · void reviewArtifact(int groupNumber): student review artifact of the particular other group. · void peerReview(Student student): Student gives other group member grade and comments on that grade. · void getGroupMembers(): returns the group members of the student who calls this method.

Class Name	TeachingAssistant
Methods	void createDeadline(Artifact artifact, Time time): This method allows the teaching assistant to assign a deadline for the specific artifact.

Class Name	Teacher
Methods	void closeCourse(): Teacher is able to close course. This method will cause all database information to be deleted. void shuffleGroups(): Teachers can shuffle the groups and it will lead to students who are not in any group to joining existing groups but have no number of students at maximum. If all of the groups have a maximum number of students at maximum, a new group will be created and students will be assigning these groups. This process will continue until there is no student without a group. Students who dropped the course will not be included.

Class Name	StudentComment
Properties	· int ID: holds the ID of the student or host who is the target of the comment · String[] commentCategory: holds the points given to the student by the peers (points given for communication, contribution etc.)
Methods	· int getID(): returns the ID of the student · void setID (int ID): sets the email address of the account · String[] getCommentCategory (): gets the commentCategory of the studentComment · void setCommentCateogry (String[] commentCategory): sets the commentCategory of the studentComment

Class Name	ArtifactComment
Properties	• int GroupNumber: the number of the group which is the owner of the artifact • String artifactName: the name of the artifact
Methods	

Class Name	Deadline
Properties	• time Time: the time of the deadline as a string • check Boolean: a boolean variable to check whether the deadline passed

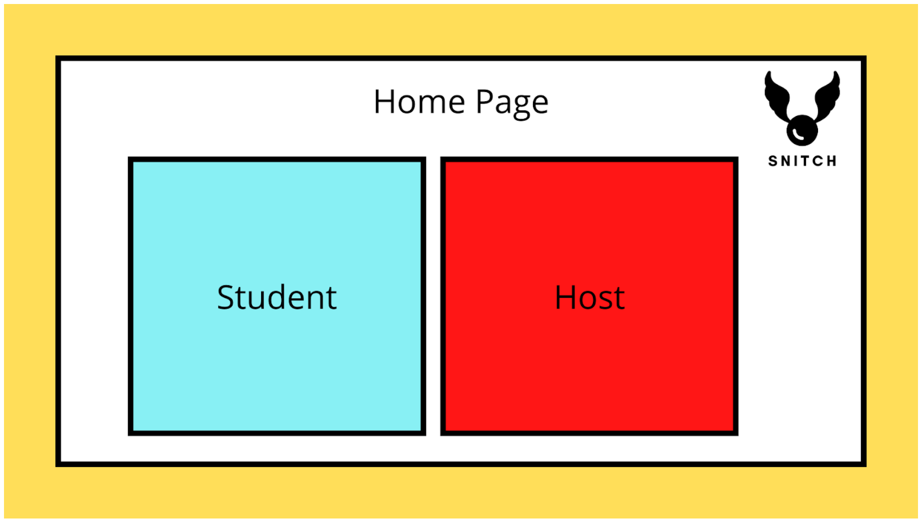
Class Name	Course
Properties	· int sectionCount: holds the number of sections. · String courseName: holds the name of the course. · int totalStuCount: holds the number of students registered to course. · String instructorName: holds the name of the instructor.
Methods	· int getSectionCount(): returns the number of sections. · void setSectionCount (int sectionCount): sets the number of sections. · String getCourseName(): returns the name of the course. · void setCourseName (String courseName): sets the name of the course. · int getTotalStuCount(): returns the student count in the course. · void setTotalStuCount(int totalStuCount): sets the students who registered the course. · String getInstructorName(): returns the instructor name of the course · void setInstructorName(String instructorName): sets the instructor name of the course

Class Name	Section
Properties	· int studentCount: holds total number of students in the section · Group[] groupList: the list of the students as an array · int groupCount: holds the number of groups in section.
Methods	· int getStudentCount(): gets the student count of the section · void setStudentCount(int studentCount): sets the student count of the section · int getGroupCount(): gets the count of the group of the section. · void setGroupCount(int groupCount): sets the group count of the section · void viewGroupList(): prints the group list of the section · Group selectAGroup(int groupNumber): selects a specific groups stated in the parameter.

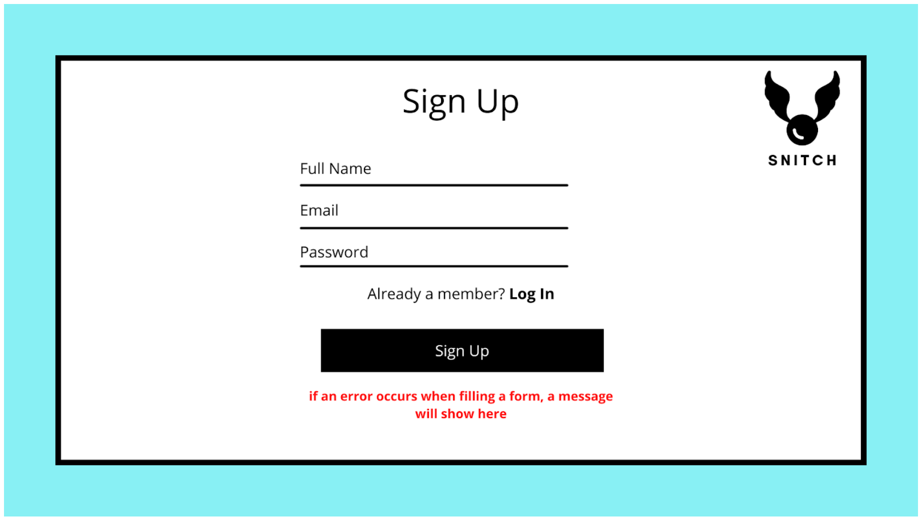
Class Name	Artifact
Properties	· String artifactName: holds the name of the artifact. · String artifactLink: holds the link of the artifact. · int groupNum: holds the group number of the artifact.

Methods	· String getArtifactName(): return the name of artifact · void setArtifactName(String artifactName): sets the name of artifact · String getArtifactLink(): return the link of artifact link · void setArtifactLink(String artifactLink): sets the link of artifact · int getGroupNum(): return the number of group · void setGroupNum(int groupNum): set the number of group

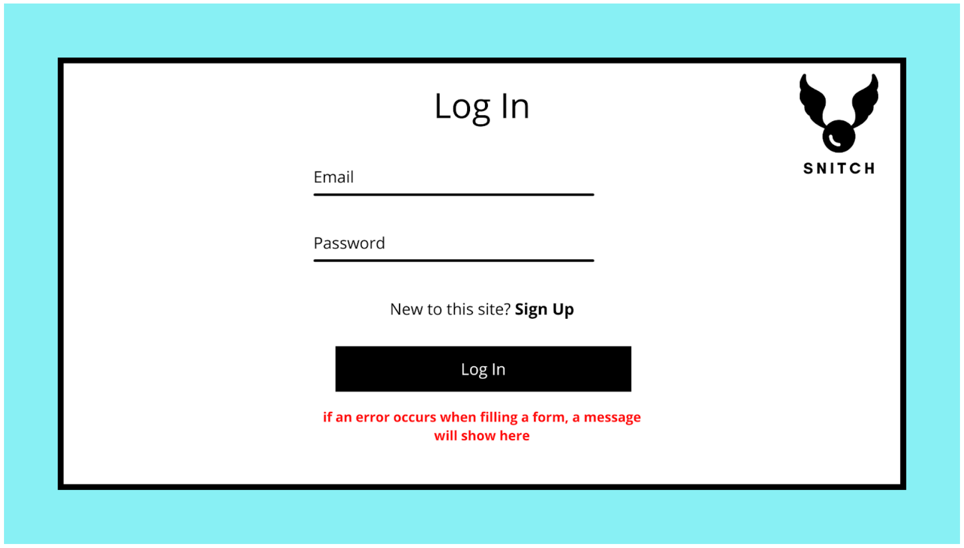
2.5.4 User Interface - Navigational Paths and Screen Mock-Ups



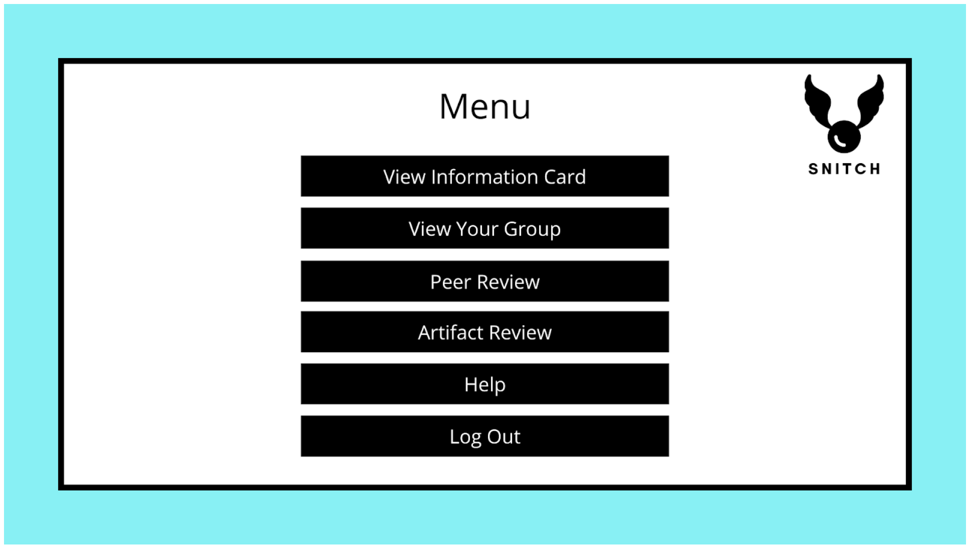
User selects the role in the system. Blue for students & red for instructor or teaching assistants. Yellow is for the pages which are the same for student and host.


Sign Up
SNITCH
Full Name
Email
Password
Already a member? **Log In**
Sign Up
if an error occurs when filling a form, a message will show here

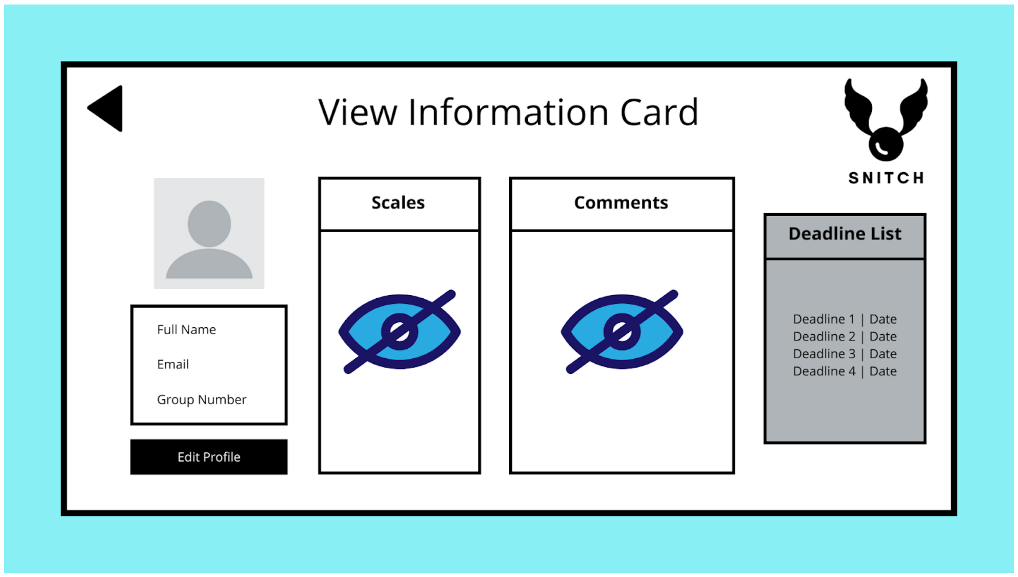
Students' Sign-Up page. If the student does not have an account, S/he can create an account. Students should type his full name, e-mail, and password. If s/he already has an account, s/he can turn to the login page. After all information parts are filled, students can complete the register part by clicking the "Sign Up" button. If there is an error, a warning message would be shown to the student.


Log In
SNITCH
Email
Password
New to this site? **Sign Up**
Log In
if an error occurs when filling a form, a message will show here

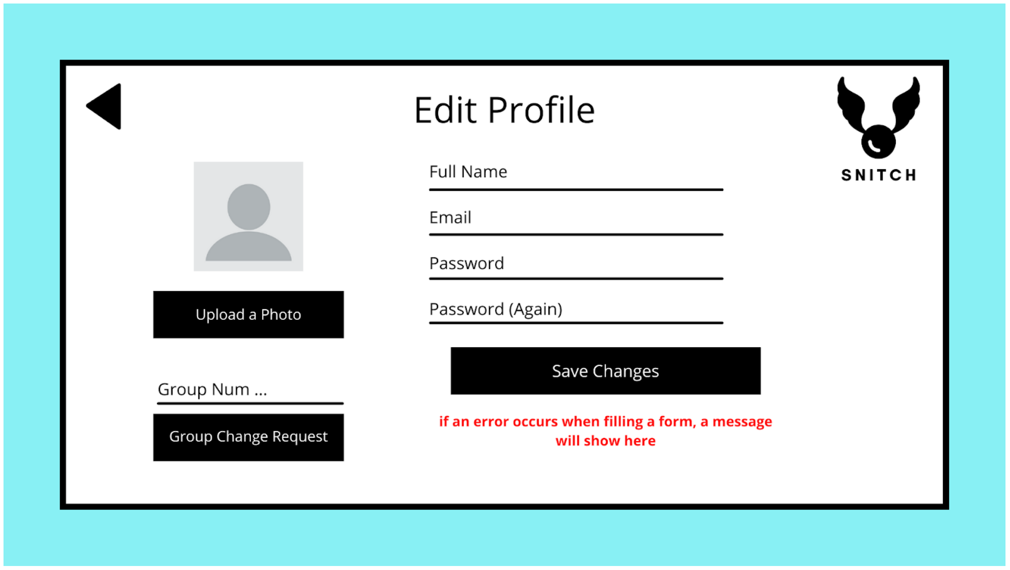
Student’s Login page. Students can login to the peer review system with their emails and passwords. If there is an error like an unmatched email and password, the warning message would be shown to the user and the user can try again. When the student enters the correct email and password, s/he will be able to access the peer review system. Also, if the student does not have an account, s/he can go to sign up page.



Students’ menu screen. In this menu students have six different options. First, students can see his or her information about the project by clicking “View Information Card”. View group button brings about other team members, students can reach him/her group works and upload a group work. Peer review part makes students able to grade their peers and make comments about them. Artifact review buttons come up with other groups' works and the student can see all of their artifacts and make comments on others' work. By means of a help button, students receive information about the system in order to use correctly. When the Log Out button is pressed, the student's account abandons the system.

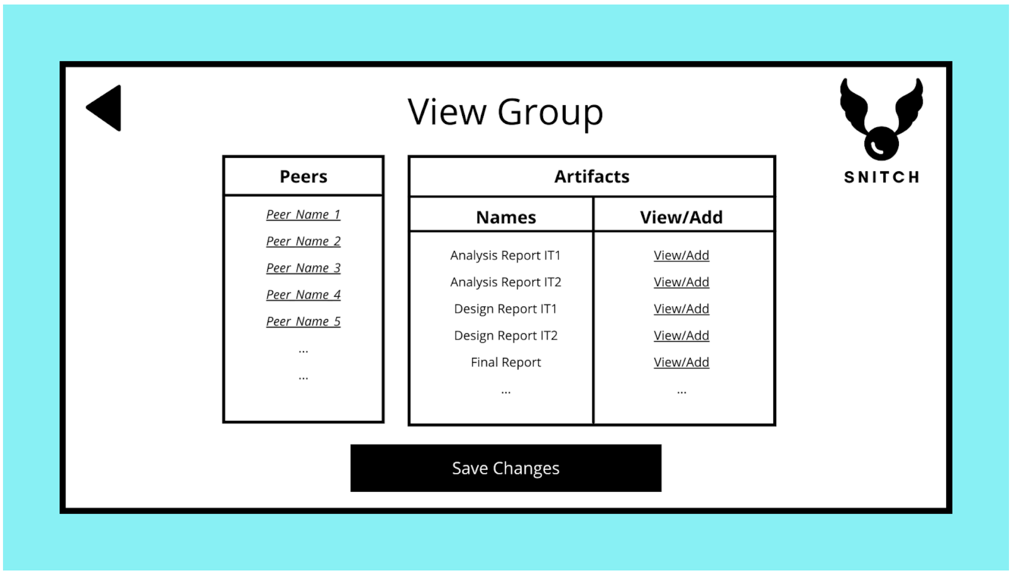


Student's view information card page. If the instructor let them, they can see their grades and comments about themselves. Deadline list also displays in this part to keep students updated. Finally, the student can see his or her photo, full name, e-mal and group number, and the student can make edits about him/herself.



if an error occurs when filling a form, a message will show here

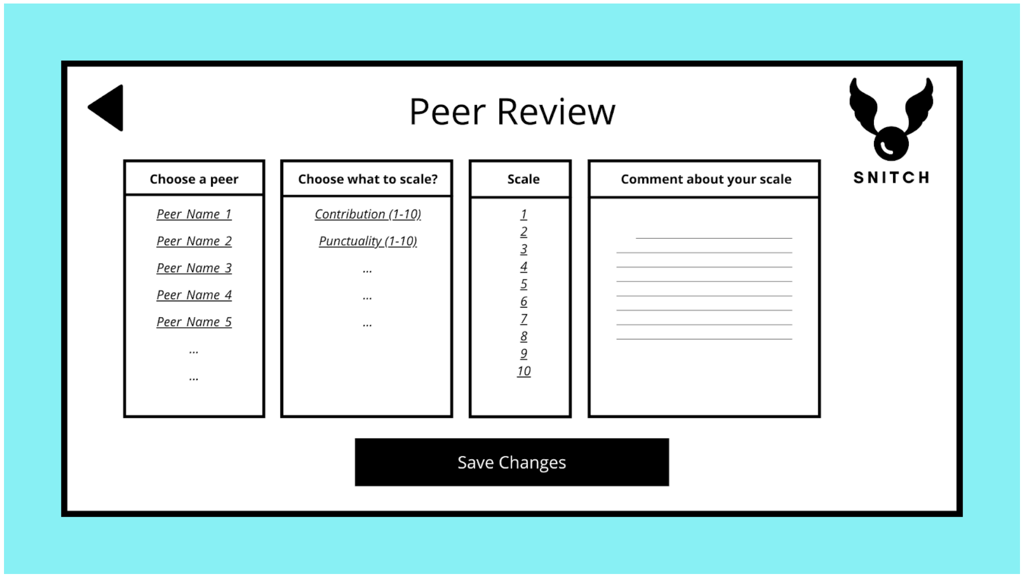
Edit profile page. Students can change profile picture, password and send a message to the instructor to change the group. When the save changes button is pressed, all changes are updated to the system. In case, if there is an error, the warning message would be shown to the user.



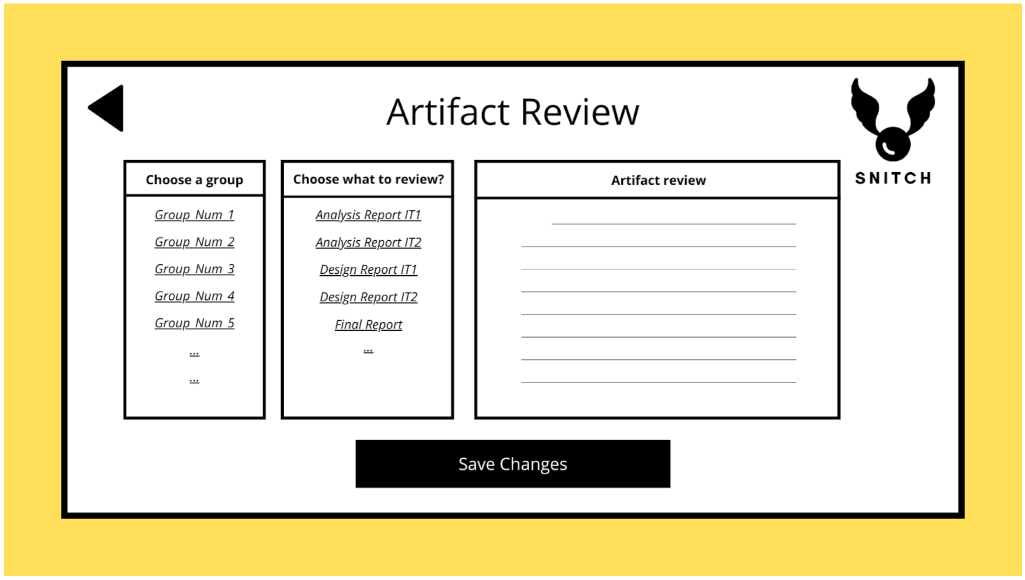
Peers
Peer Name 1
Peer Name 2
Peer Name 3
Peer Name 4
Peer Name 5
...
...

Artifacts	
Names	**View/Add**
Analysis Report IT1	View/Add
Analysis Report IT2	View/Add
Design Report IT1	View/Add
Design Report IT2	View/Add
Final Report	View/Add
...	...

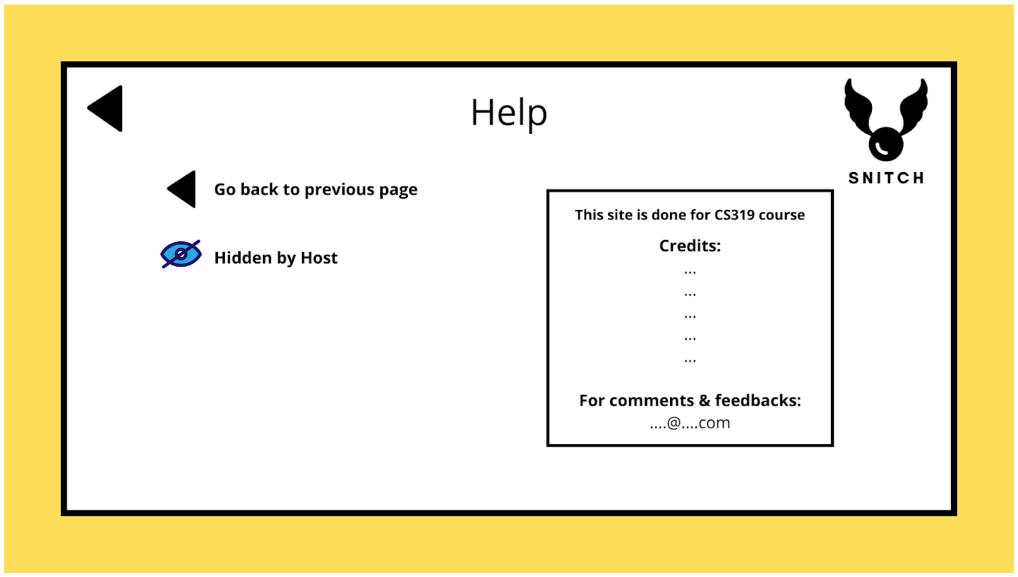
View group page. Students can see all of the team members and all artifact types. After selecting a type of artifact, students can upload the work or view it. When the save changes button is pressed, all changes are updated to the system.



Peer review page. Firstly, a peer should be chosen, then scale ought to be determined. After those two parts, the student gives the grade to the peer. Finally, the student is supposed to explain the point and make comments on peer's work. When the save changes button is pressed, all changes are updated to the system.



In the Artifact Review screen, students/hosts are able to make reviews by choosing a specific group and by indicating their assignments. The Save changes button will be clickable if and only if the student/hosts chooses a group and the assignment.



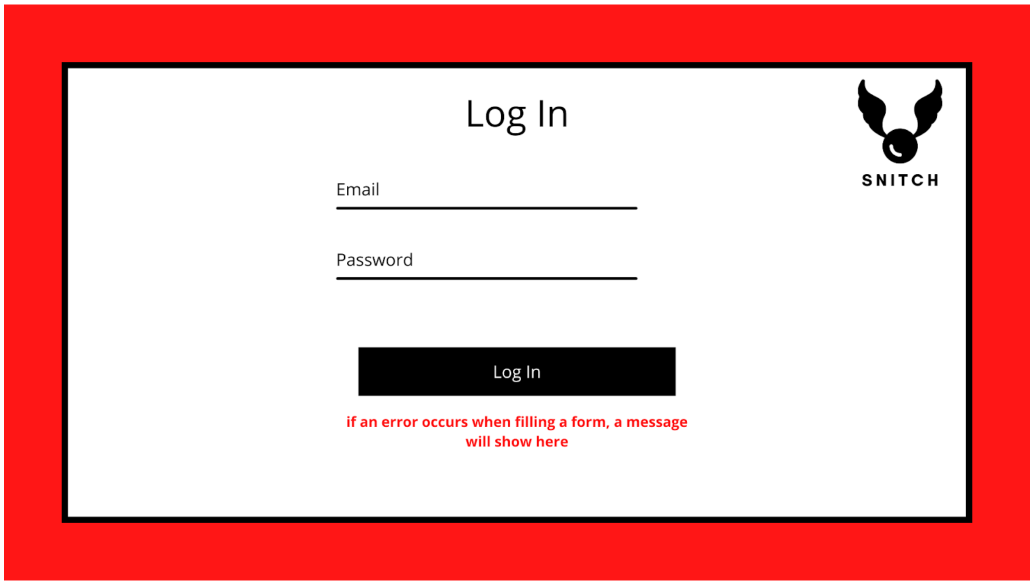
This site is done for CS319 course

Credits:

...
...
...
...
...

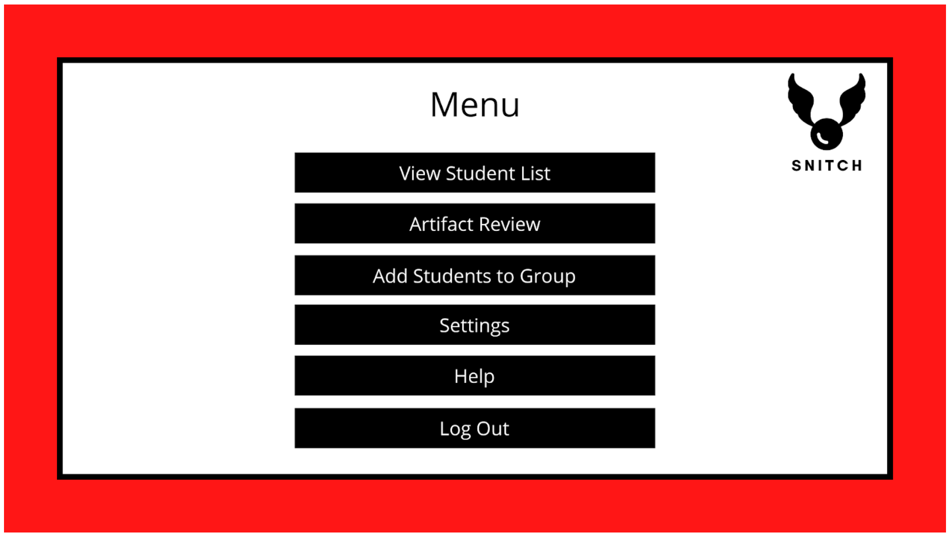
For comments & feedbacks:
....@....com

In the Help screen, students/hosts will be able to see the credits, comments and feedback section. Moreover, some basic information about the app will be available on this screen.

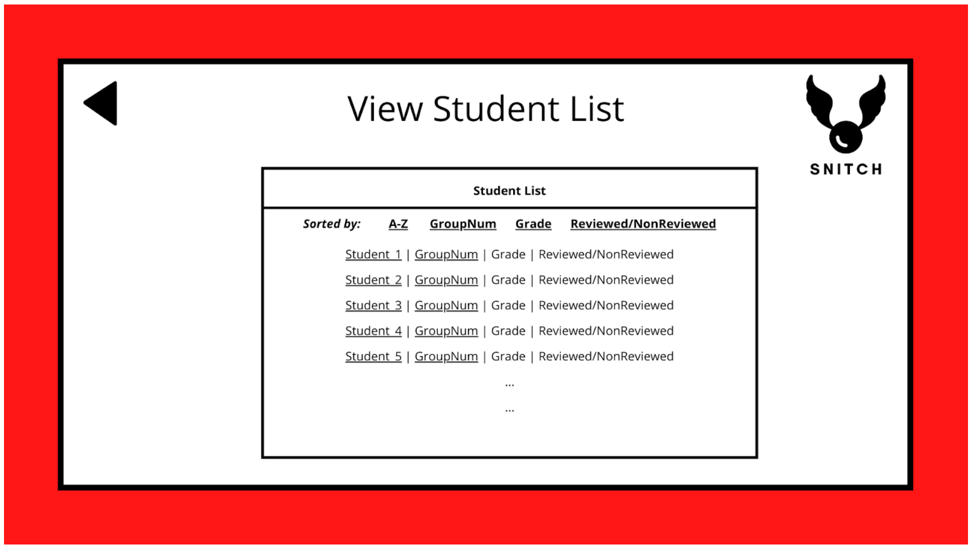


if an error occurs when filling a form, a message will show here

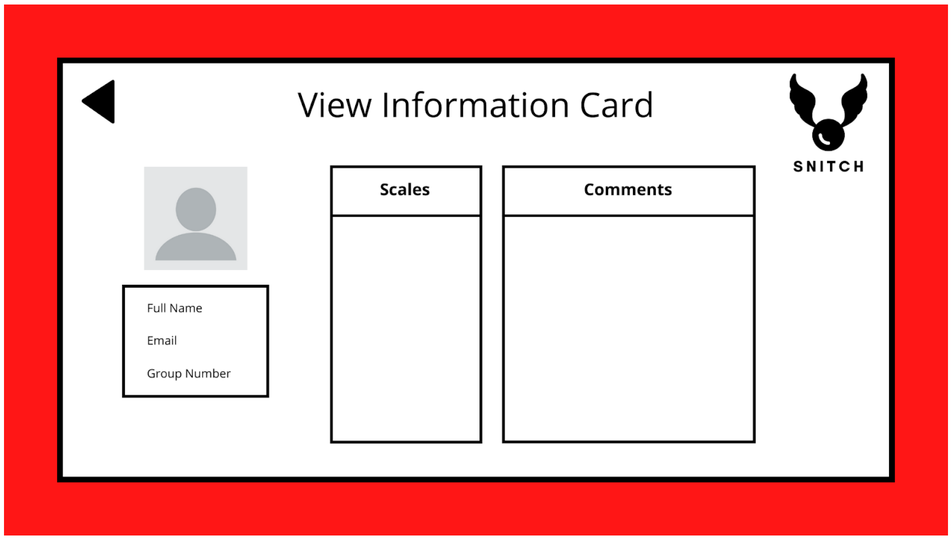
In the Host Log-in screen, the host must fill the email and password section to click the Login button. If the email and password are matched, the host will be able to enter his/her menu.



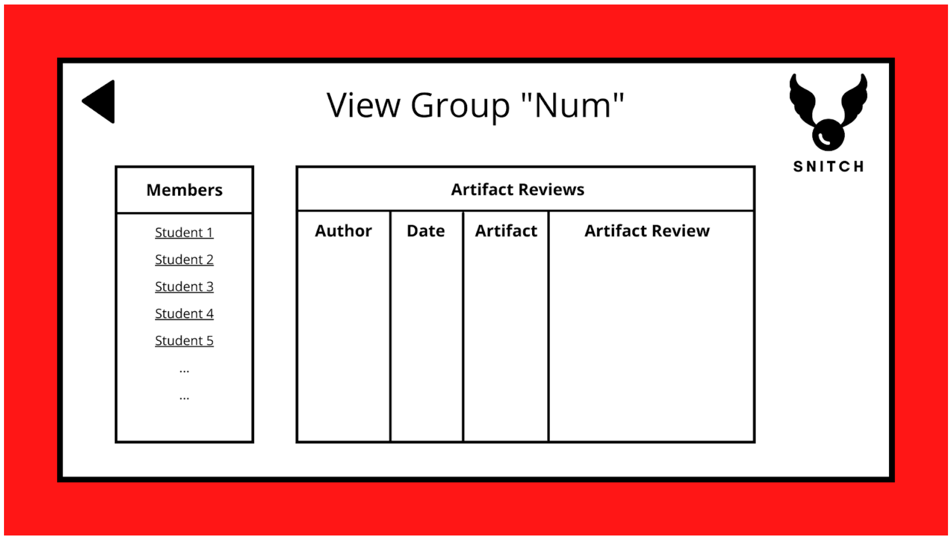
In the host Menu screen, the host will be able to decide between the options of View Student List, Artifact Review, Settings, Help, Add Students to Group, Log Out. Student will be able to log out from his/her account by clicking LogOut button.


View Student List
SNITCH
Student List
Sorted by: A-Z GroupNum Grade Reviewed/NonReviewed
Student 1 | GroupNum | Grade | Reviewed/NonReviewed
Student 2 | GroupNum | Grade | Reviewed/NonReviewed
Student 3 | GroupNum | Grade | Reviewed/NonReviewed
Student 4 | GroupNum | Grade | Reviewed/NonReviewed
Student 5 | GroupNum | Grade | Reviewed/NonReviewed
...
...

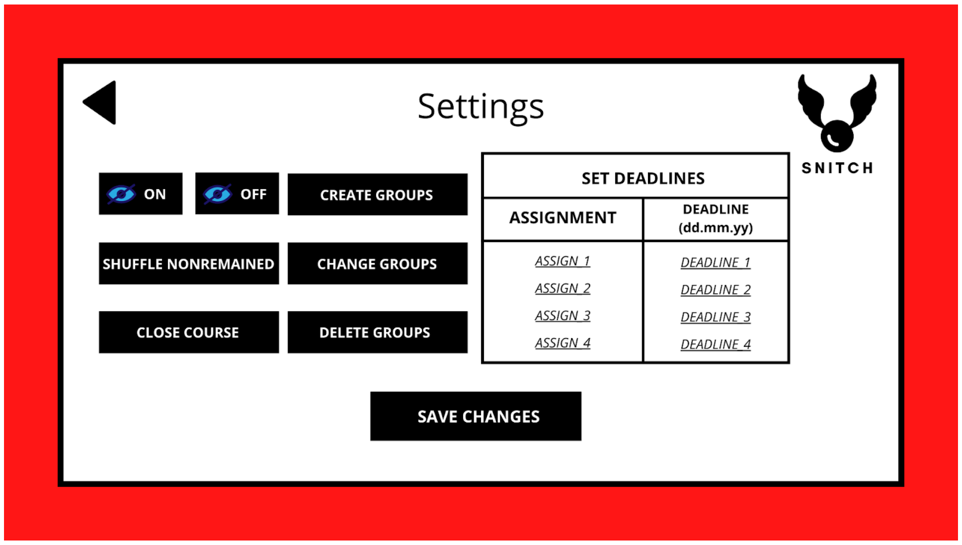
In the host View Student List screen, the host will be able to students by alphabetical order, group number, grade and by reviewed options. Host must click the buttons of A-Z for alphabetic order, GroupNum for group number, Grade for grades and Reviewed/NonReviewed for reviewed option. Moreover, the host can click a student to direct to the View Information Card of that particular student. Also, the host can click to GroupNum to direct to the View Group “Num'' screen of that particular group.



In the View Information Card screen of the host, the host will be able to see the particular student information card. This screen will include the name, email, group number, scales and matching comments of scales of that particular student. Host will be able to click the back button to return to main menu of the host.

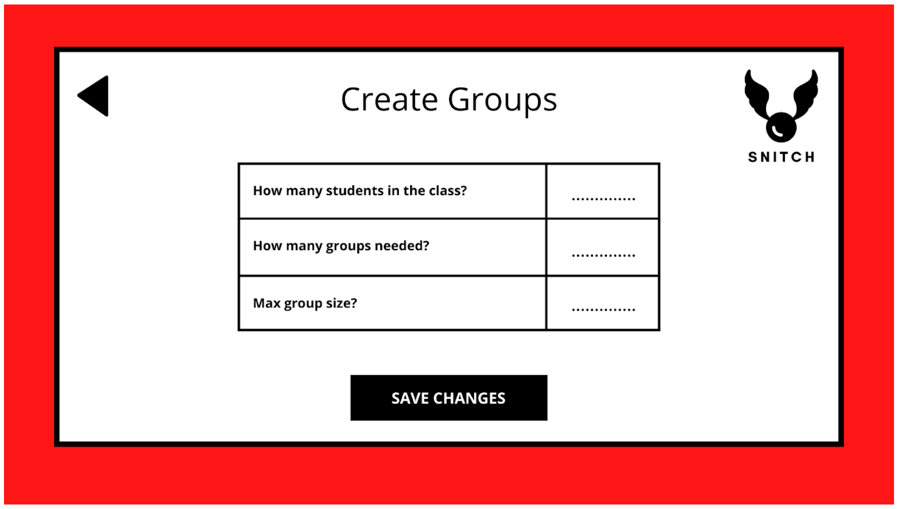


In the View Group Num screen, there will be the students in the particular group. Host will be able to click the students by their names to direct to their student cards. Moreover, they can see their artifacts reviews made by other students indicating their name, date and artifact name.



SET DEADLINES	
ASSIGNMENT	DEADLINE (dd.mm.yy)
ASSIGN 1	DEADLINE 1
ASSIGN 2	DEADLINE 2
ASSIGN 3	DEADLINE 3
ASSIGN 4	DEADLINE 4

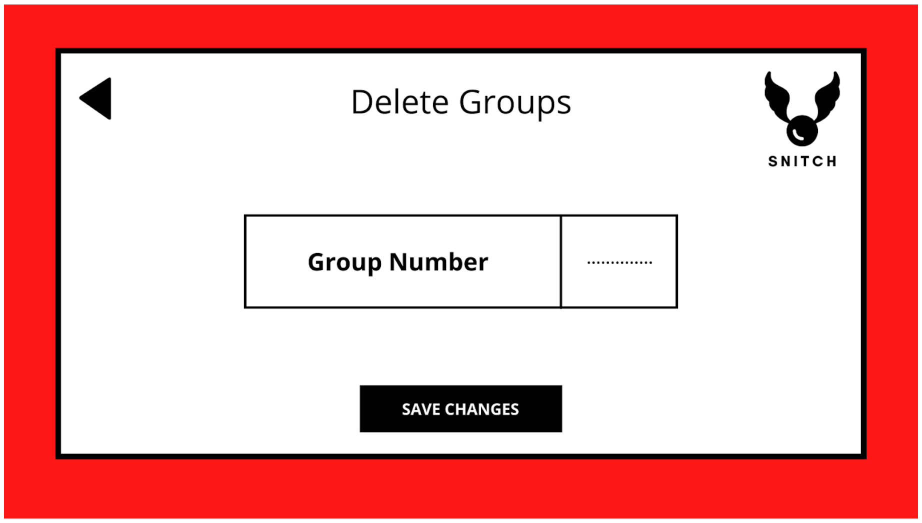
The "Settings" screen provides the host with such functions as "Change Visibility", "Create Groups", "Shuffle Nonremained", "Change Groups", "Close Course", "Delete Groups" and "Set Deadlines". The "Change Visibility" section is a switch function and it allows the host to make it visible for students to see who gave him/her which feedbacks and reviews. Therefore, it can be named as an anonymity switch. The "Create Groups" and "Change Groups" buttons will be explained later in their screens. The "Shuffle Nonremained" button assigns all of the students who have not assigned to any groups yet in random order. "Set Deadlines" section provides the host to create notifiers for the assignment deadlines in order for the students to check. The host can simply name new assignments, add their deadline dates or edit them. the "Delete Groups" button provides the host to remove the group of his/her choice and after the host enters the desired group number, that group is immediately removed from the course and if there were any students assigned in it, those students will go back to the Nonremained group, waiting to be assigned to their new groups. And finally, the "Close Course" button deleted all records of students, groups and artifacts to terminate the session. This function will be used by the host at the end of each semester.



Create Groups

How many students in the class?	....
How many groups needed?	....
Max group size?	....

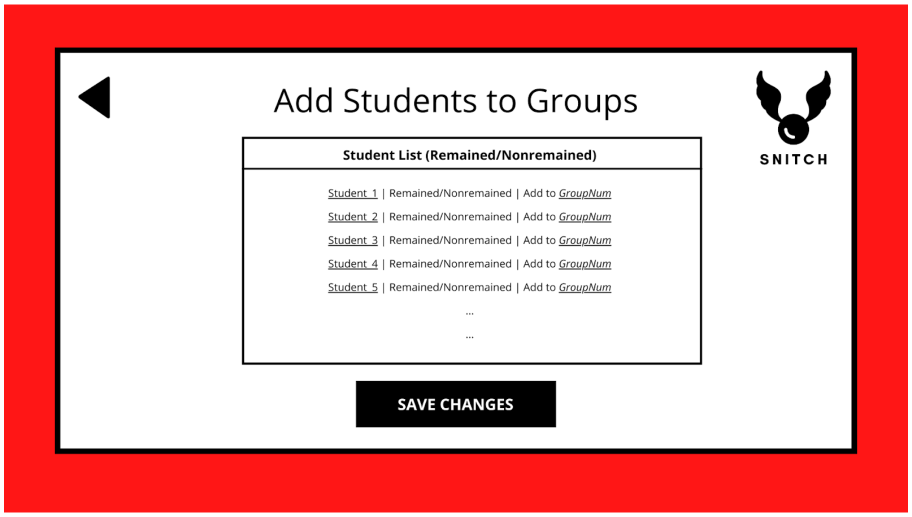
The “Create Groups” screen opens up when the button with the same name is pressed in the “Settings” screen. This page allows the host to enter the sliders of the groups by their choice like the numbers of students in the class, the number of groups, and the capacity of every single group. After these features are entered by the host, s/he can press the “Save Changes” button and the specialized groups will be created with their unique group numbers.



Delete Groups

Group Number	....

The “Delete Groups” screen opens up when the button with the same name is pressed in the “Settings” screen. This page allows the host to enter the group number which will be deleted. After the number is entered by the host, s/he can press the “Save Changes” button and the specialized group/s will be deleted.



Add Students to Groups

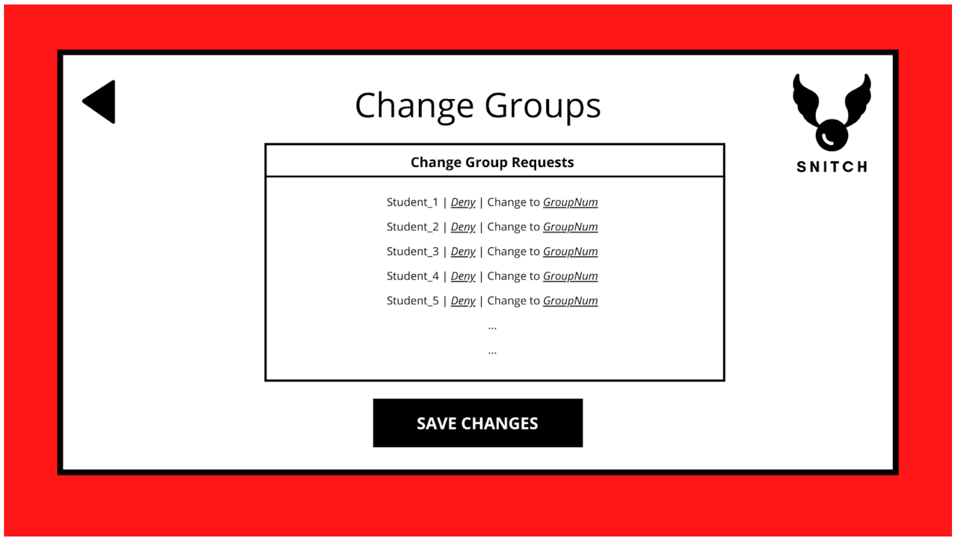
SNITCH

Student List (Remained/Nonremained)

Student_1 | Remained/Nonremained | Add to GroupNum
Student_2 | Remained/Nonremained | Add to GroupNum
Student_3 | Remained/Nonremained | Add to GroupNum
Student_4 | Remained/Nonremained | Add to GroupNum
Student_5 | Remained/Nonremained | Add to GroupNum
...
...

SAVE CHANGES

The "Add Student to Group" screen. The host selects the group number and adds the student to that group.



Change Groups

SNITCH

Change Group Requests

Student_1 | *Deny* | Change to *GroupNum*
Student_2 | *Deny* | Change to *GroupNum*
Student_3 | *Deny* | Change to *GroupNum*
Student_4 | *Deny* | Change to *GroupNum*
Student_5 | *Deny* | Change to *GroupNum*
...
...

SAVE CHANGES

The "Change Groups'' screen displays all the group change requests made by the students assigned in some groups. The requests are in a list format where the host can see which student demands to get transferred to which group. The host can press the "Deny" button to dismiss the request, or accept it by pressing "Change to <GroupNum>" where GroupNum is the group number in which the student demands to get transferred. If the desired group is full, an error message occurs in the host's page.

3. Improvement Summary

- Minor typo’s are fixed.
- Captions are added to the figures.
- Figures are mentioned within the text.
- Overview of the Proposed System is extended.
- Some stats, analytics about reviews are provided.

Use Case Diagram: RequestGroupChange use case added to the ViewInformationCard as an extension. UploadAssignment use case added to the Student actor. ReviewArtifact use case linked to the Student actor. Host linked to the ViewGroupList use case. ShuffleRemainingStudents use case has been added to the CreateGroups as an extension. CreateDeadline use case added to the Host.

Activity Diagram: In the first iteration, we have one activity diagram that shows every possible action in the system. For iteration two, there are two activity diagrams, one for the students and other for the hosts. These diagrams are not showing every action of the system. Instead, it shows how the process occurs.

State Diagrams: In the first iteration, our state diagram was not sufficient given. In the second iteration, state diagrams are created from the beginning. There are two state diagrams, one for students and one for submission. Student State Diagram shows student states from logging in to the system for the first time to withdrawing state and requesting change for groups. The Submission State Diagram is used for submitting, evaluating, and reviewing artifacts.

Sequence Diagrams: In the first iteration, our sequence diagrams were based on actors, which was wrong. In the second iteration, they are related to scenarios. Instead of one sequence diagram, we added several sequence diagrams for the main functionality. Also, in the lifeline part, we included objects instead of names like database, login page etc. Moreover, we explained the sequence diagram with a couple of sentences.

Class Diagram: In the first iteration, our class diagram has not an adequate number of classes to handle the complexity for required work to be done and there are changes required for existing classes as well. We expanded our class diagram by adding Deadline Class, Course Class, Section Class, Artifact Class with correct connection to the other classes. Student & Artifact Comment Classes are added to inherit Comment Class. Moreover, TeacherAssistant and Teacher Classes are added to inherit Host Class in order to differentiate them. Also, we added class description to the newly added classes. Lastly, multiplicity of the classes are controlled and some of them are fixed.

UI Design: The logo of the application is added to the mock-ups. “Delete Groups” screen is added to the system. Some small changes are made.