Institute of Information Technology

University of Dhaka

S.R.S. Individual Assignment

Software Project Management Application (SPMA)

Course Code: SE406

Submitted by

Abu Jafar Saifullah

Roll: BSSE 1109

Institute of Information Technology

University of Dhaka

Submitted to

Dr. Kazi Muheymin-Us-Sakib

Designation: Professor

Institute of Information Technology

University of Dhaka

Table of Contents

Usage Scenario

1. Batch Management:

There will be a provision of adding students as the member of the application. The Project Coordinator, an assigned teacher, also the admin of this application, should add one or more students to the application, called CR. The CR can add his/her classmates to the system. To add a student member following information will be needed – Student Roll Number and Student Name. The admin will registrar him/herself by giving his/her Name and ID. The admin can add other teachers as the supervisors with the same information.

2. Group Management:

The application should have group entity, and each group should be created by the Project Coordinator. Each group may contain one or multiple number of students. Each group should also have a group name, project title and a supervisor.

3. Activity and Marks distribution Management:

Each of the group requires to present their progress weekly to the coordinator. Coordinator will maintain the progress as the form of attendance, that is, if the progress is satisfactory, the student will get the attendance otherwise will be considered as absent. The sum of attendances will be converted to 10 marks. There will be one project-proposal-presentation carrying 10 marks, one Mid-presentation carrying 10 marks and one Final-presentation carrying 10 marks. The code-review (10) and project-showcasing (10) will consist 20 and Final-report 20. Rest 20 will be given by the supervisor. Coordinator will post the marks time to time. A student will be able to see her marks.

4. Post Management:

All the activity related notices will be posted by the CR. All the notices should automatically be posted on the Facebook group. (All project related data that is code and reports should be kept in the GitHub). GitHub link for each of the group will be available in the application. Any timely activity notice should also be on the upcoming activity corner.

Use Case Diagram

DEFINITION OF USE CASE:

A Use Case captures a contract that describes the system behavior under various conditions as the system responds to a request from one of its stakeholders. In essence, a Use Case tells a stylized story about how an end user interacts with the system under a specific set of circumstances. A Use Case diagram simply describes a story using corresponding actors who perform important roles in the story and makes the story understandable for the users. The first step in writing a Use Case is to define that set of “actors” that will be involved in the story. Actors are the different people that use the system or product within the context of the function and behavior that is to be described. Actors represent the roles that people play as the system operators. Every user has one or more goals when using system.

Primary Actor:

Primary actors interact directly to achieve required system function and derive the intended benefit from the system. They work directly and frequently with the software.

Secondary Actor:

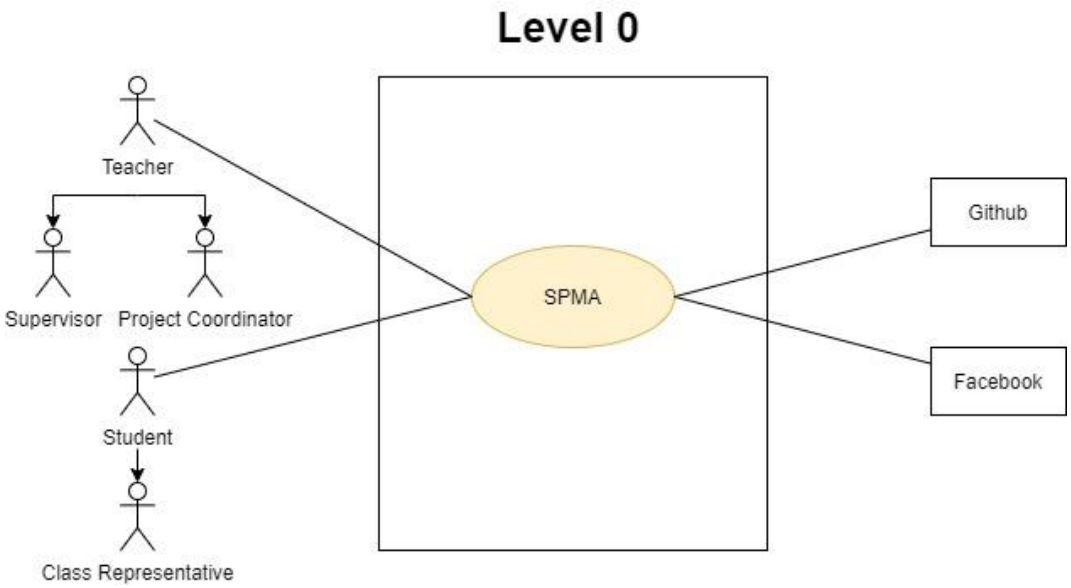
Secondary actors support the system so that primary actors can do their work. They either produce or consume information.

Use Case diagrams give the non-technical view of overall system.

Level 0:

Primary Actors: Project Coordinator, Class Representative, Student, Supervisor

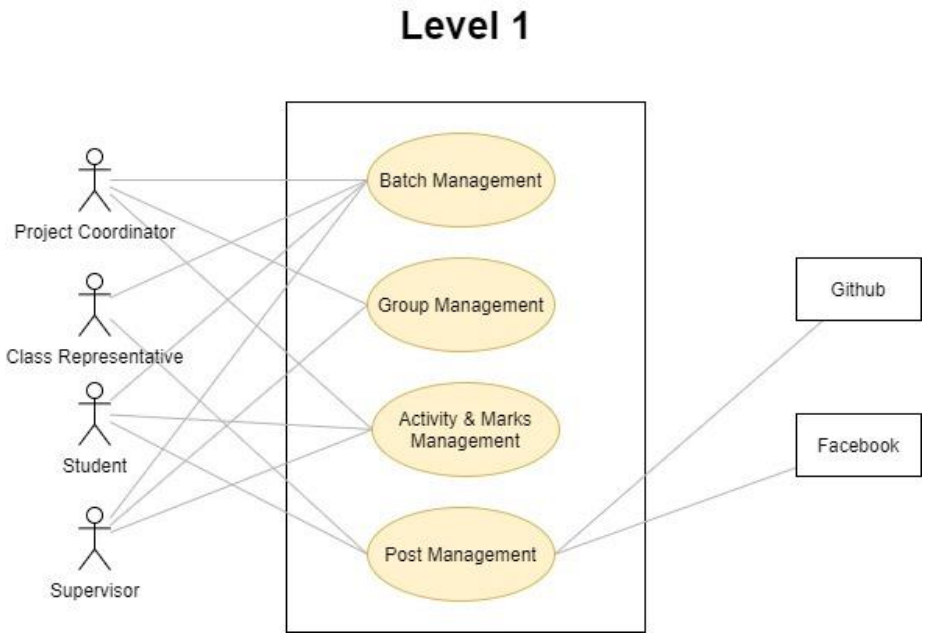
Secondary Actors: GitHub, Facebook



Level 1:

Primary Actors: Project Coordinator, Class Representative, Student, Supervisor

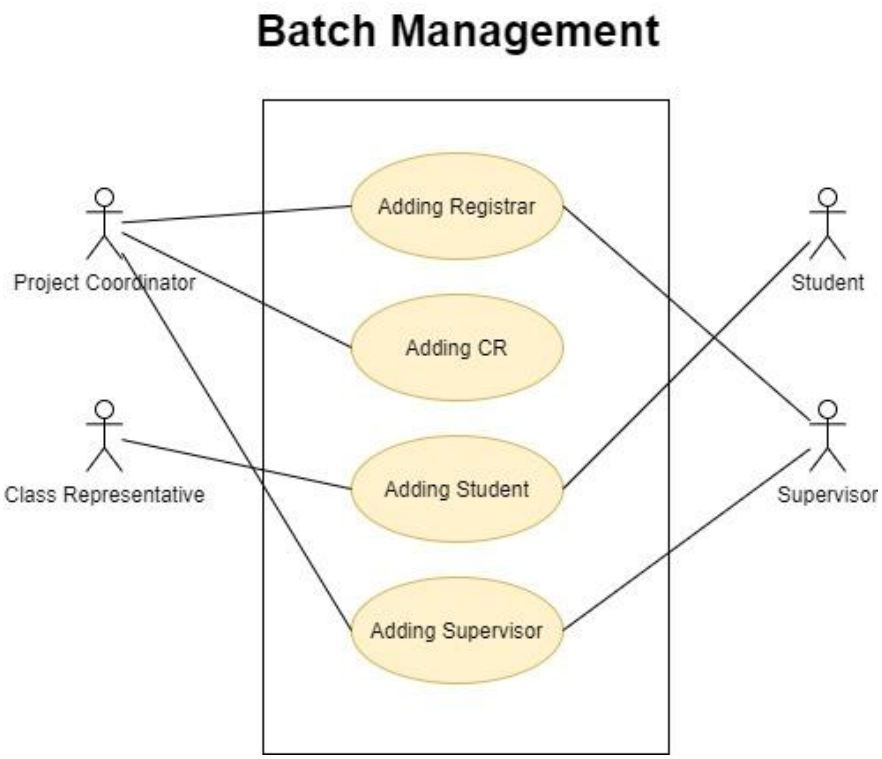
Secondary Actors: GitHub, Facebook



Level 1.1:

Primary Actors: Project Coordinator, Class Representative

Secondary Actors: Student, Supervisor



Description of use case diagram level 1.1:

Registrar: The Project Coordinator(admin) will register him/herself by giving name and ID.

Add CR: The Project Coordinator would add one or more students to the application as Class Representative.

Add Supervisor: The admin can add other teachers as the supervisors with the information regarding their name and id.

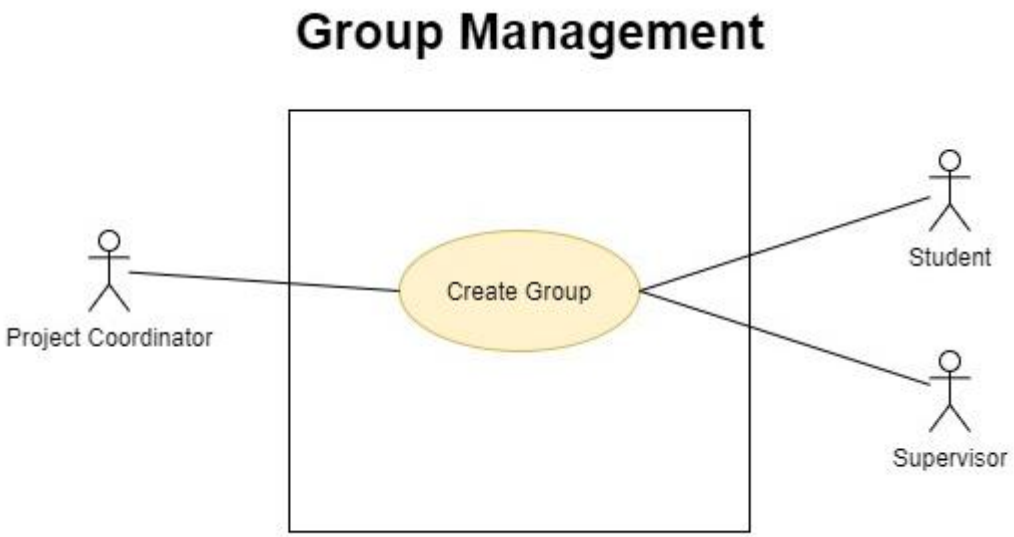
Add Student: The CR can add his/her classmates to the system. To add a student member the student will provide the necessary info (Student Roll Number and Student Name).

Level 1.2:

Primary Actors: Project Coordinator

Secondary Actors: Student, Supervisor

Description of use case diagram level 1.2:

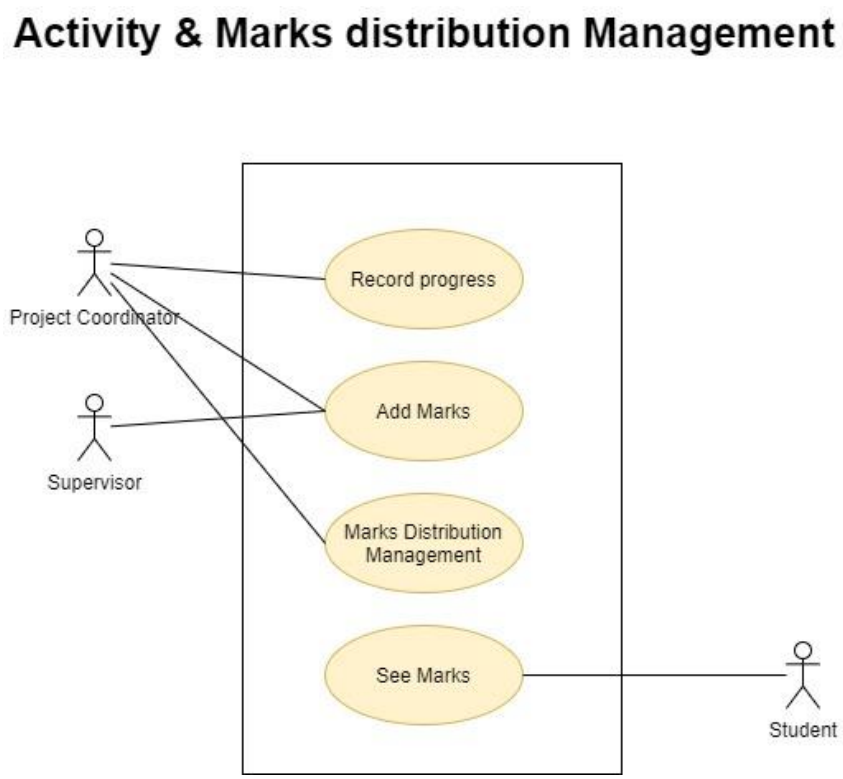


Creation of Group: The Project Coordinator (Admin) will create groups consisting of multiple students and assign a supervisor for each group.

Level 1.3:

Primary Actors: Project Coordinator, Class Representative, Supervisor

Secondary Actors: Student



Description of use case diagram level 1.3:

Weekly Progress: Each of the groups requires to present their progress weekly to the coordinator. The coordinator will maintain the progress in the form of attendance.

Add Marks: The Project Coordinator will evaluate the students and add marks based on 'Attendance', 'Project Proposal Presentation', 'Mid Presentation', 'Code Review' and 'Project Showcasing'. The supervisor needs to add marks 'Supervisor'

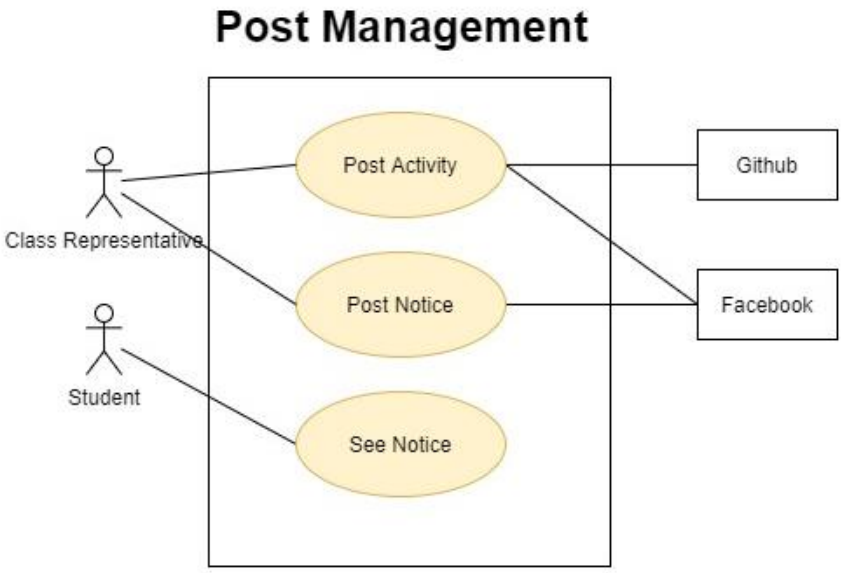
Marks Distribution Management: In this module, the Project Coordinator will have the permission to tabulate batch marks and distribute marks as their weight. The Project Coordinator will also post marks.

See Marks: The student will be able to check their marks obtained.

Level 1.4:

Primary Actors: Class Representative, Student, Supervisor

Secondary Actors: GitHub, Facebook



Post Activity: All project-related data that is code and reports should be kept in GitHub. The GitHub link for each of the groups will be available in the application.

Post Notice: All the activity-related notices will be posted by the CR. All the notices should automatically be posted on the Facebook group.

See Notice: The student will be able to check on the notices and also post GitHub links and reports.

ACTIVITY DIAGRAM

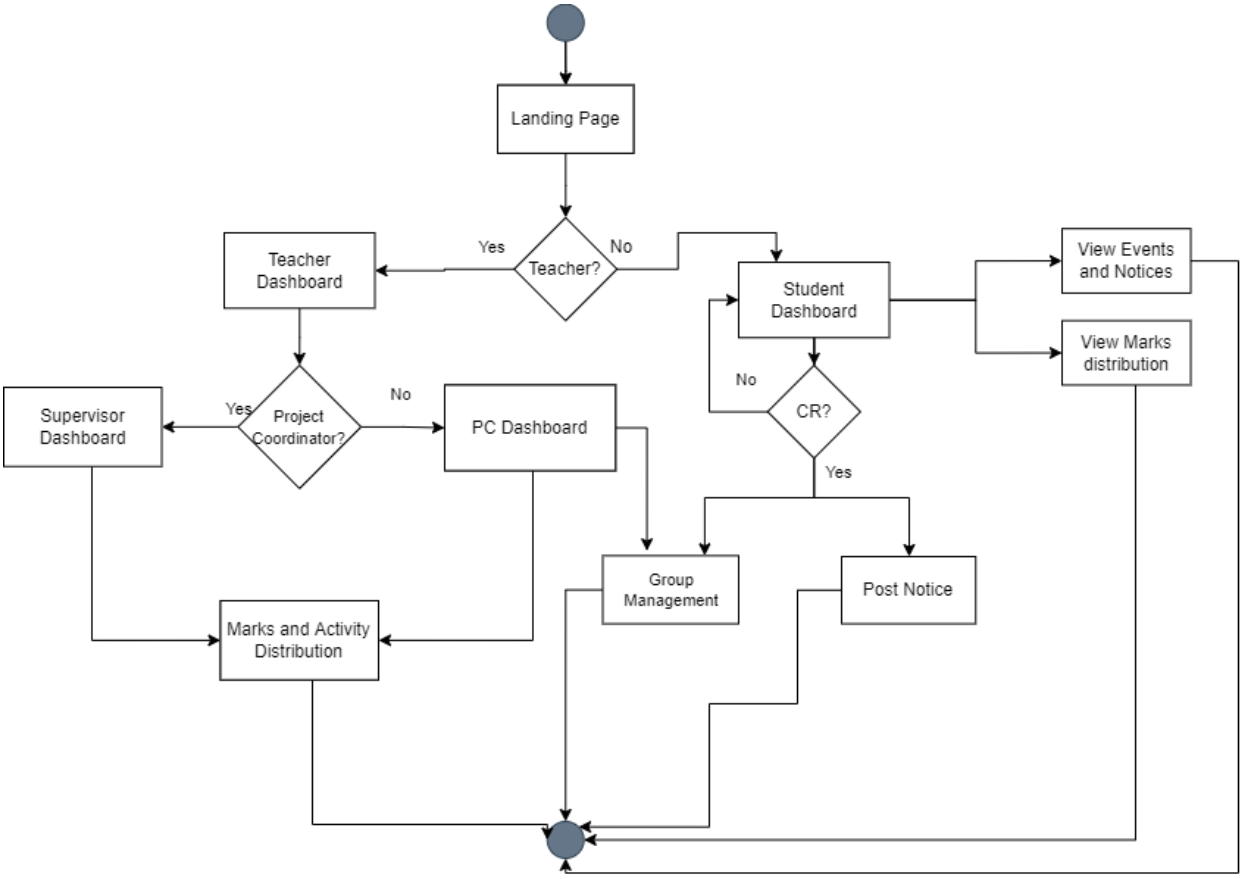
Definition of Activity Diagram:

The activity diagram is an important behavioral diagram in the UML diagram to describe dynamic aspects of the system. An activity diagram is essentially an advanced version of a flowchart that models the flow from one activity to another activity.

Level 1:

Name: Software Project Management Application

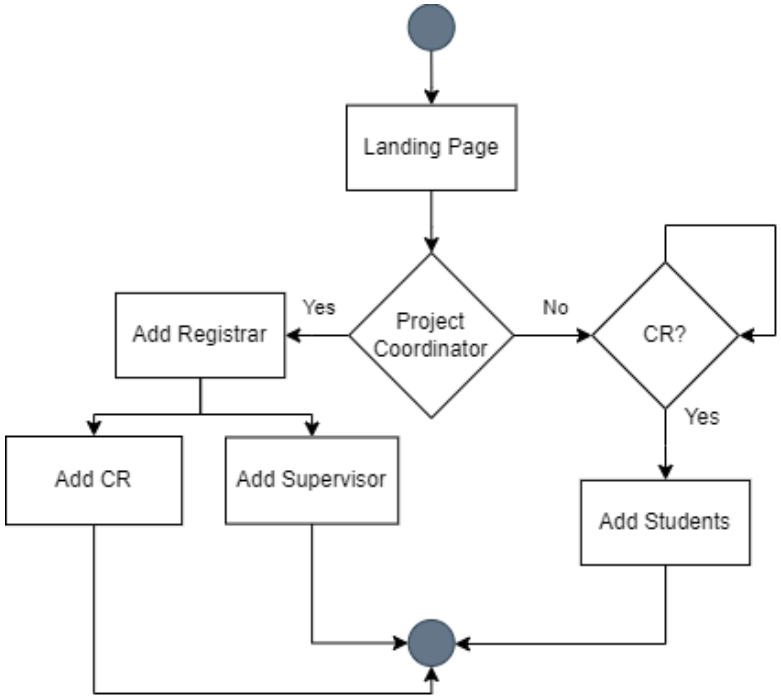
Reference: Use case Diagram 'Level – 1'



Level 1.1:

Name: Batch Management

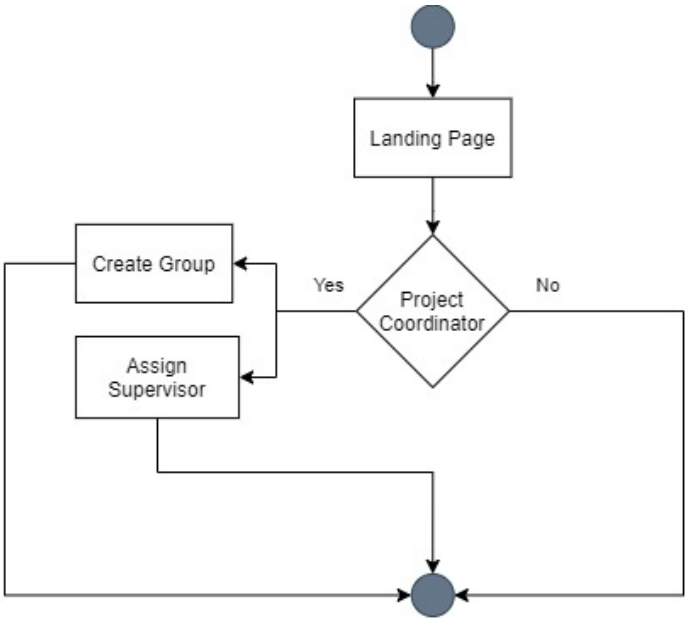
Reference: Use case Diagram 'Level – 1.1'



Level 1.2:

Name: Group Management

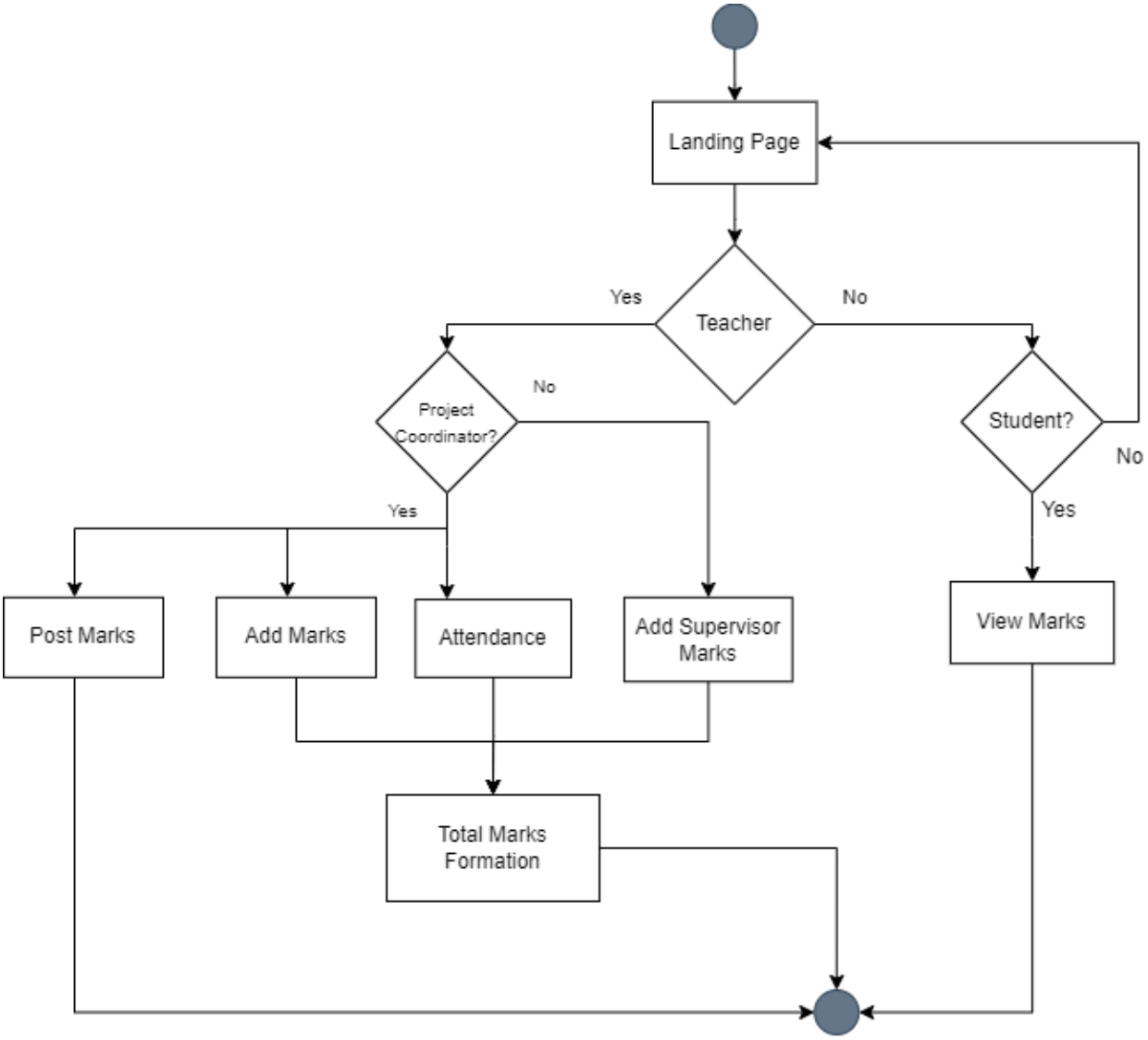
Reference: Use case Diagram 'Level – 1.2'



Level 1.3:

Name: Activity & Marks Distribution Management

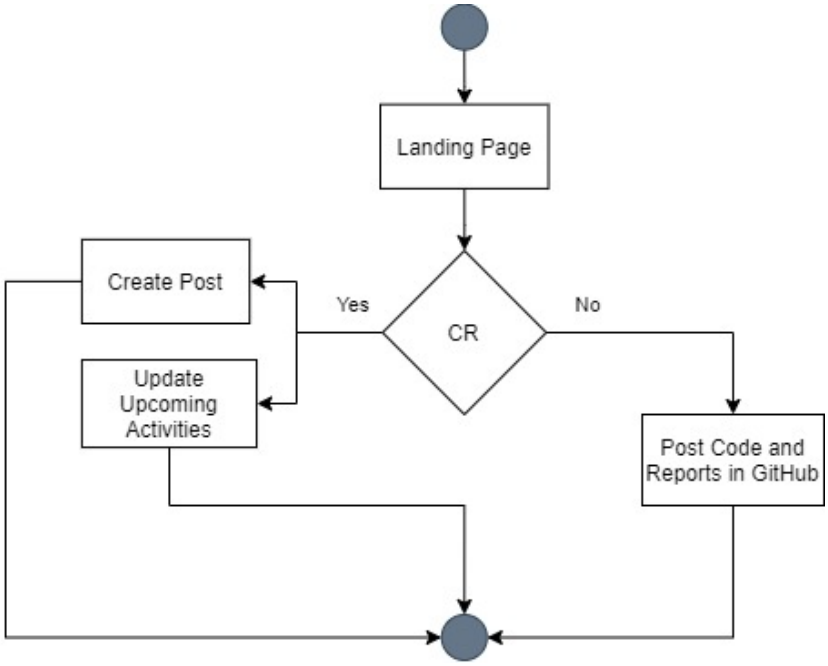
Reference: Use case Diagram 'Level – 1.3'



Level 1.4:

Name: Post Management

Reference: Use case Diagram 'Level – 1.4'



Swimlane Diagram

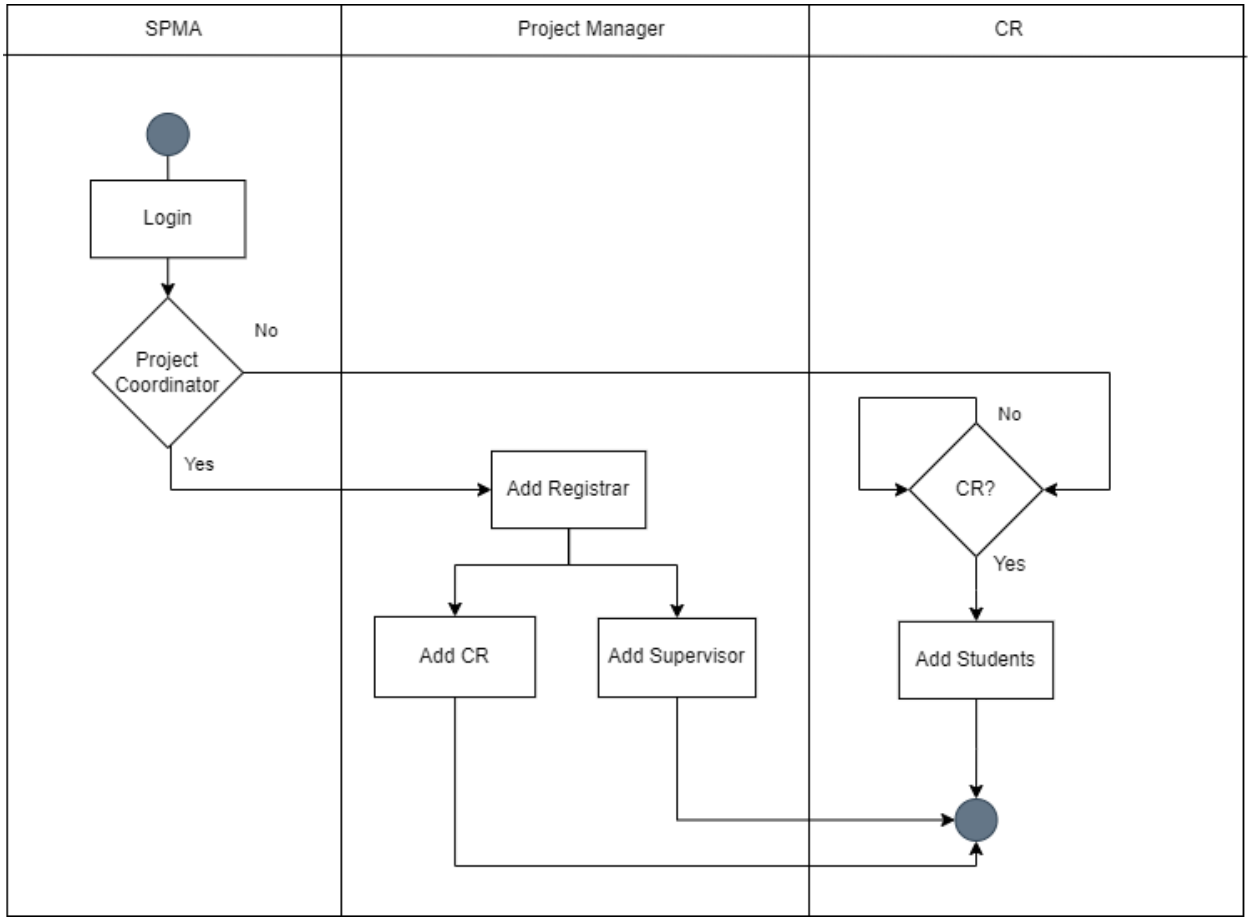
Definition:

A Swimlane diagram is a type of flowchart that delineates who does what in a process. Using the metaphor of lanes in a pool, a swim-lane diagram provides clarity and accountability by placing process steps within the horizontal or vertical 'swimlanes' of a particular employee, workgroup, or department. It shows connections, communication, and handoffs between these lanes, and it can serve to highlight waste, redundancy, and inefficiency in a process.

SID (Swimlane ID): 1.1

Name: Batch Management

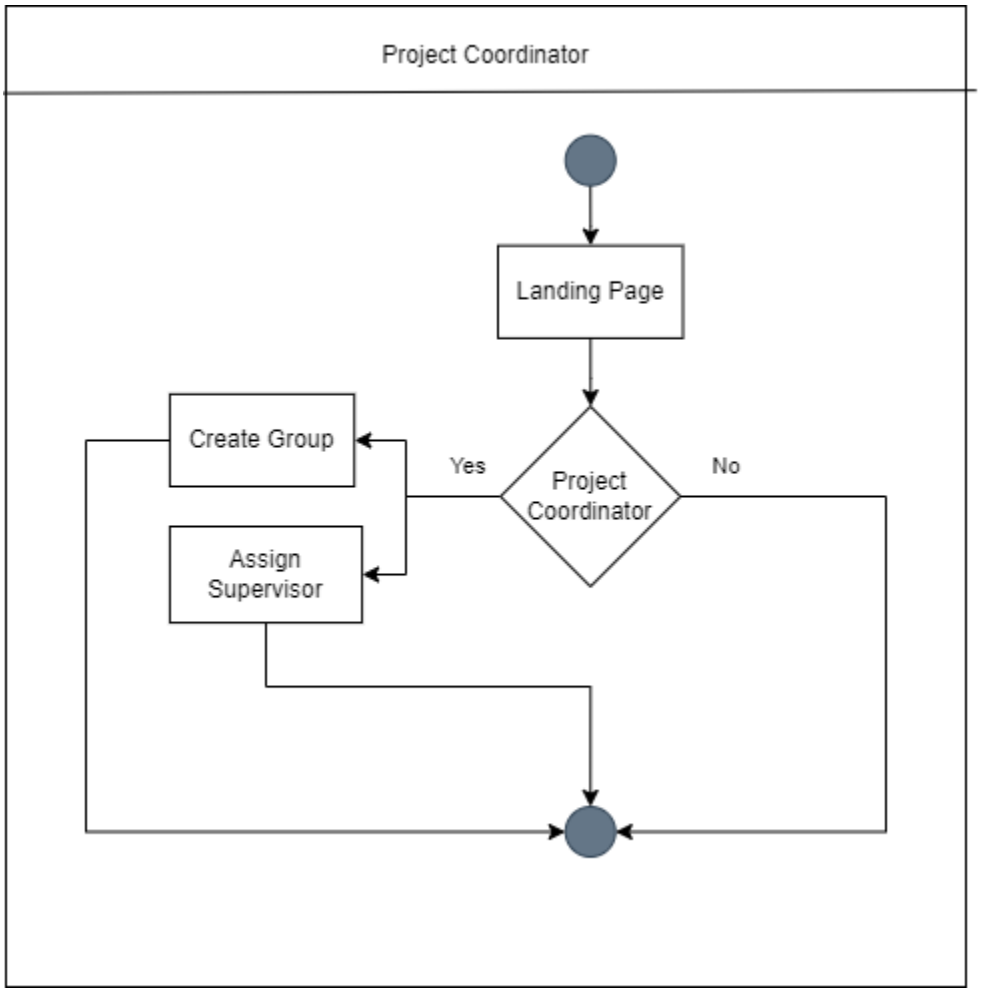
Reference: Use case & Activity diagram level-1.1



SID (Swimlane ID): 1.2

Name: Group Management

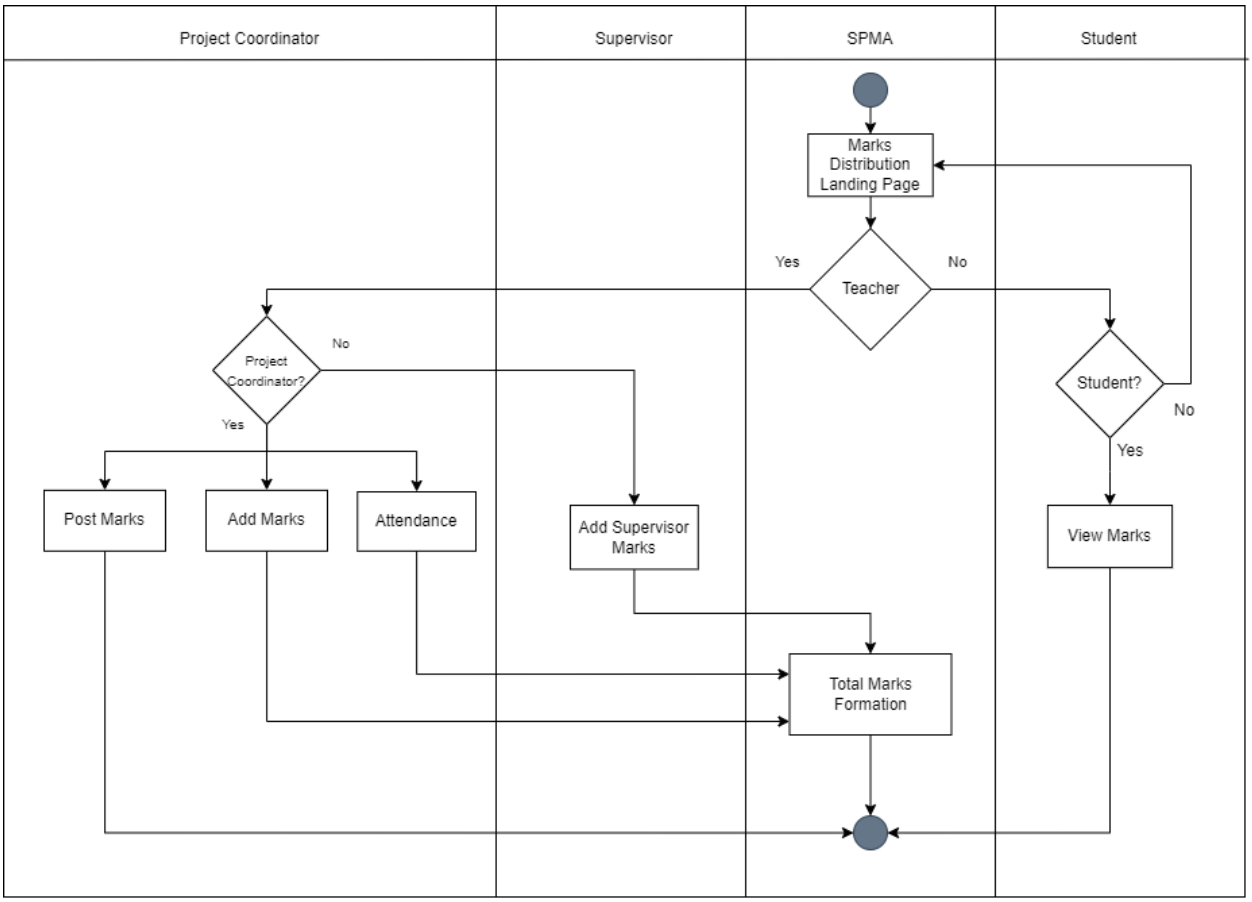
Reference: Use case & Activity diagram level-1.2



SID (Swimlane ID): 1.3

Name: Activity & Marks Management

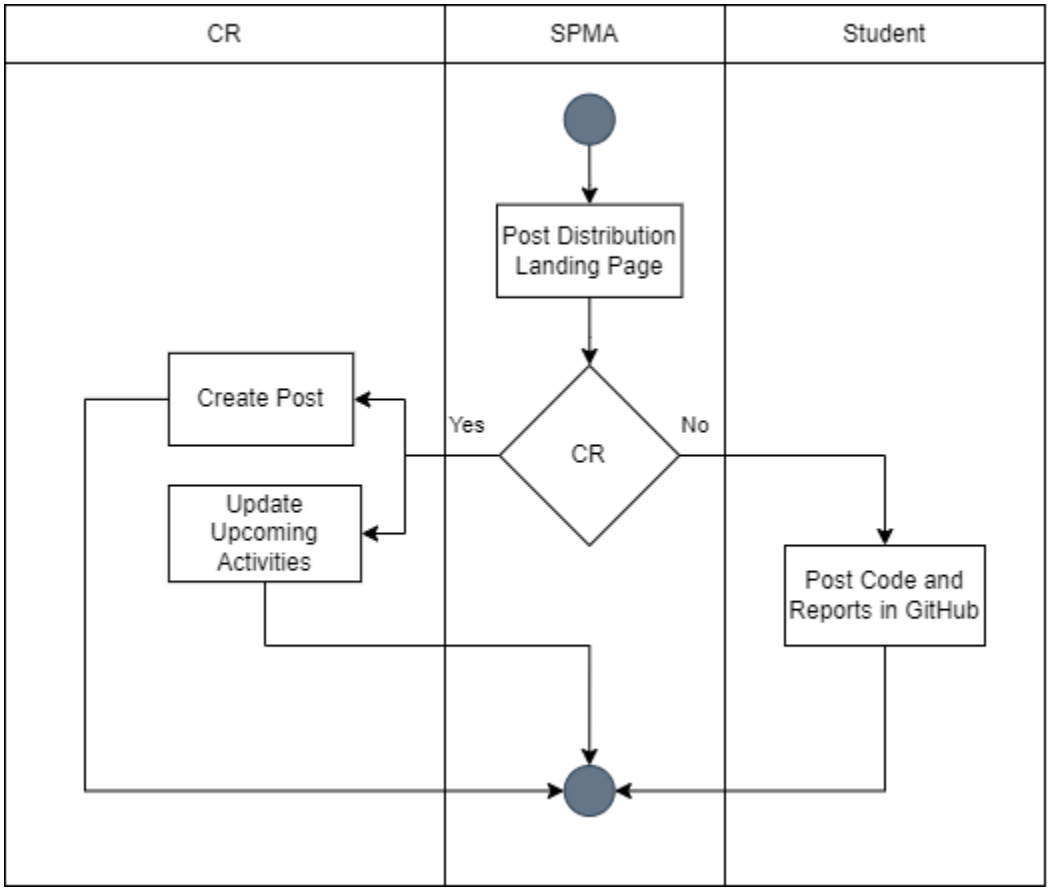
Reference: Use case & Activity diagram level-1.3



SID (Swimlane ID): 1.4

Name: Post Management

Reference: Use case & Activity diagram level-1.4



Database Modelling

Data modelling concept:

If software requirements include the necessity to create, extend or interact with a database or complex data structures need to be constructed and manipulated, then the software team chooses to create data models as part of overall requirements modelling. The entity relationship diagram (ERD) defines all data objects that are processed within the system, the relationships between the data objects and the information about how the data objects are entered, stored, transformed and produced within the system.

Data objects:

A data object is a representation of composite information that must be understood by the software. Here, composite information means information that has a number of different properties or attributes. A data object can be an external entity, a thing, an occurrence, a role, an organizational unit, a place or a structure.

Data Object Identification:

SL	Nouns	Problem/ Solution Space	Attributes
1.	Batch	S	16
2.	Provision	P	
3.	Member	P	
4.	Application	P	
5.	Project-Coordinator	S	14, 15
6.	Teacher	P	
7.	Admin	P	
8.	CR	S	
9.	Classmates	P	
10	System	P	

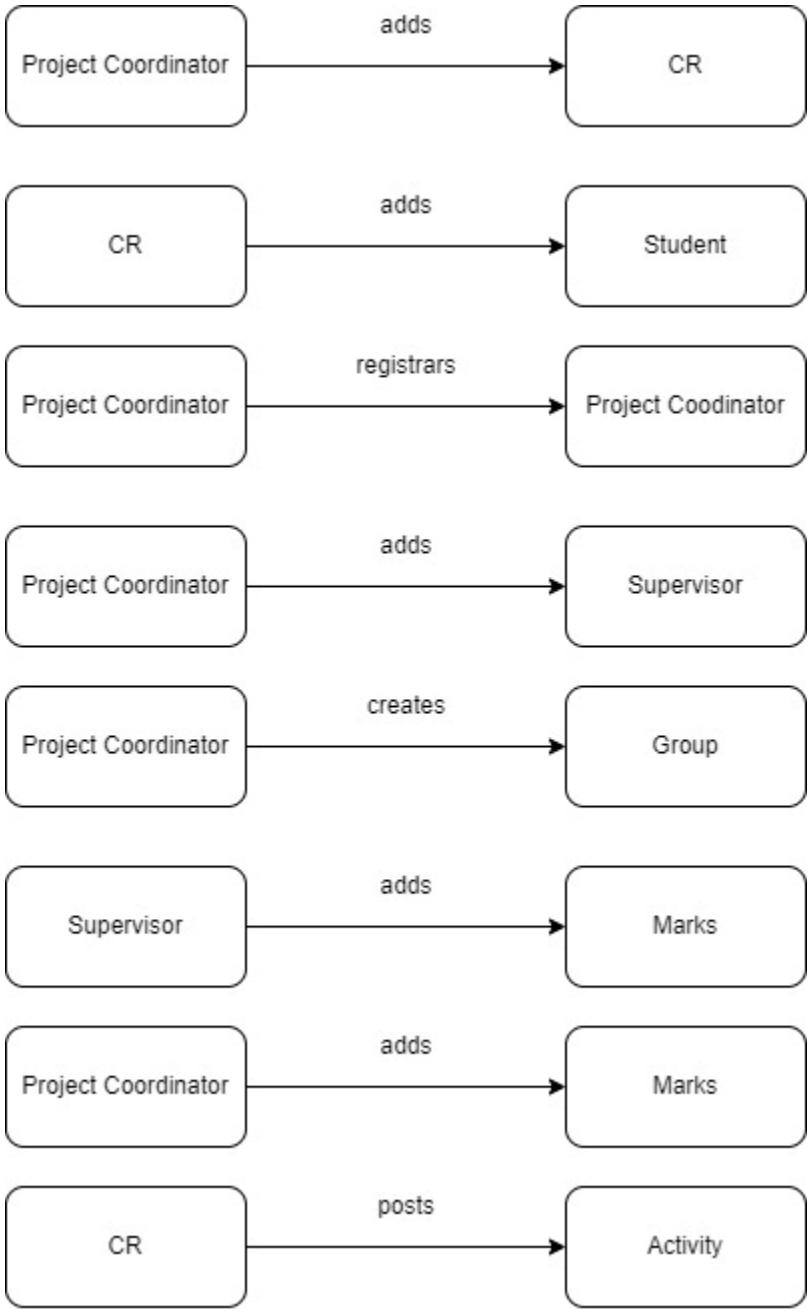
11	Information	P	
12	Student	S	13, 14
13	Roll-number	S	
14	Name	S	
15	ID	S	
16	Supervisor	S	14, 15
17	Group	S	12, 14, 16, 20,
18	Entity	P	
19	Project	P	
20	Title	S	
21	Progress	P	
22	Form	P	
23	Attendance	S	
24	Sum	P	
25	Project-proposal-presentation	S	
26	Mid-presentation	S	
27	Final-presentation	S	
28	Code-review	S	
29	Project-showcasing	S	
30	Final-report	S	
31	Rest(Remaining)	S	
32	Time	P	
33	Coordinator	P	
34	Marks	S	23, 25, 26, 27, 28,29,30,31

35	Notices	S	
36	Activity	S	35, 39, 40
37	Facebook	S	
38	Data	P	
39	Code	S	
40	Reports	S	
41	GitHub-link	S	
42	Corner	P	

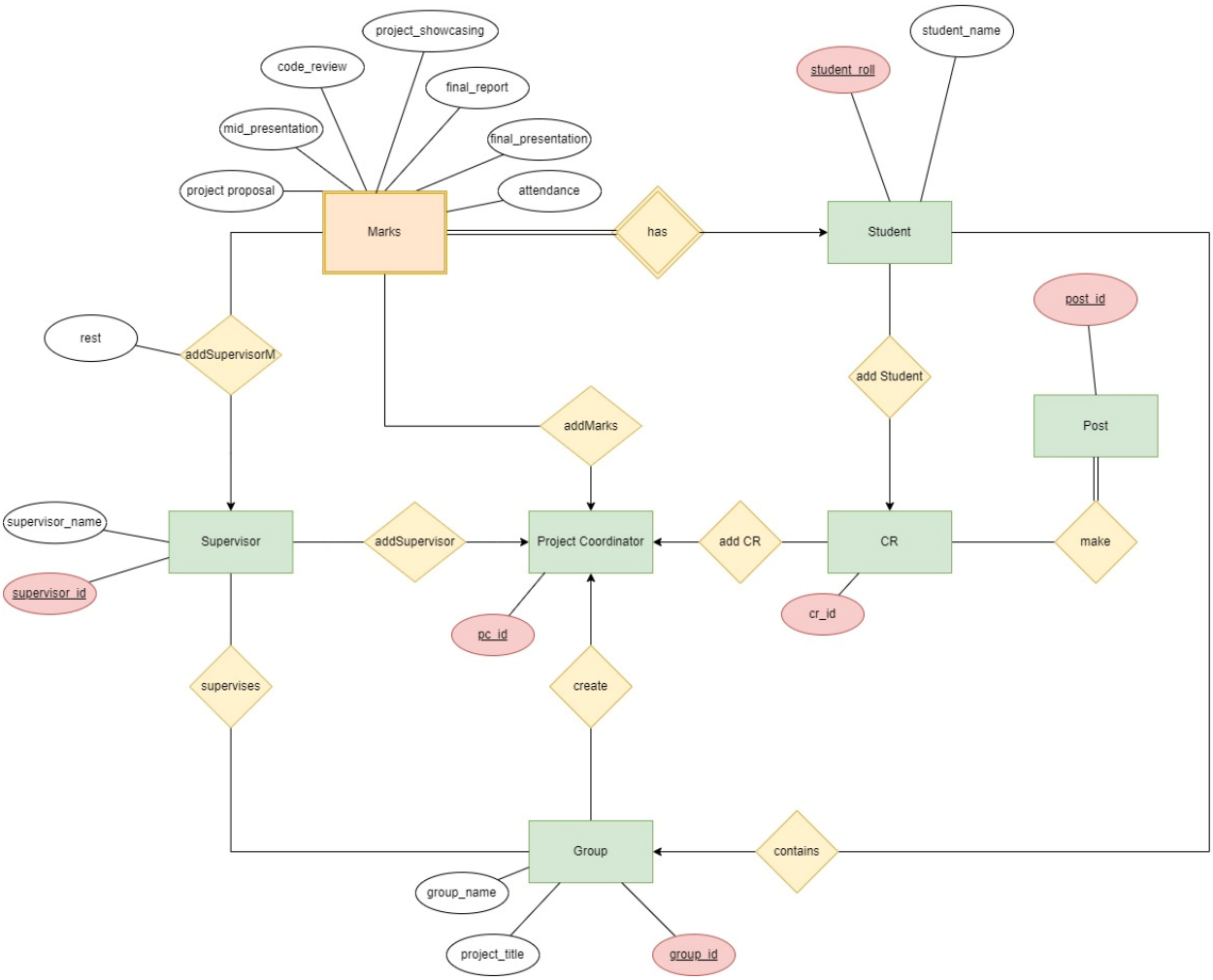
Final Data Object

- Student
- Project-coordinator
- Supervisor
- Group
- CR
- Activity
- Marks

Relationship among Data Objects



Entity Relationship Diagram



Schema Diagram

Data Object	Attribute	Type	Size
Student	-<u>student_roll</u> -student_name *-cr_id* *-group_id*	varchar varchar varchar varchar	40 40 40 40

Project Coordinator	-pc_id	varchar	40
CR	-cr_id *-pc_id*	varchar varchar	40 40
Supervisor	-supervisor_id -supervisor_name *-pc_id*	varchar varchar varchar	40 40 40
Group	-group_id -group_name -group_title -supervisor_id -pc_id	varchar varchar varchar varchar varchar	40 40 40 40 40
Marks	-student_roll *-pc_id* *-supervisor_id* -attendance -project proposal -mid_presentation -final_presentation -code_review -project_showcasing -final_report	varchar varchar varchar Number Number Number Number Number Number Number	40 40 40 4 4 4 4 4 4 4
Post	-post_id -cr_id	varchar	40
Supervises	-group_id -supervisor_id	varchar varchar	40 40

CLASS-BASED MODELING

CLASS BASED MODELING CONCEPT:

Class-based modelling represents the objects that the system will manipulate, the operations that will be applied to the objects, relationships between the objects and the collaborations that occur between the classes that are defined.

Noun list from Software Project Management Application

SL	Nouns
43.	Batch
44.	Provision
45.	Member
46.	Application
47.	Project-Coordinator
48.	Teacher
49.	Admin
50.	CR
51.	Classmates
52.	System
53.	Information
54.	Student
55.	Roll-number
56.	Name
57.	ID
58.	Supervisor
59.	Group
60.	Entity
61.	Project

62.	Title
63.	Progress
64.	Form
65.	Attendance
66.	Sum
67.	Project-proposal-presentation
68.	Mid-presentation
69.	Final-presentation
70.	Code-review
71.	Project-showcasing
72.	Final-report
73.	Rest
74.	Time
75.	Coordinator
76.	Marks
77.	Notices
78.	Activity
79.	Facebook
80.	Data
81.	Code
82.	Reports
83.	GitHub-link
84.	Corner

List of Verbs in SPMA

SL	Verb
1.	Adding Student
2.	Add classmates
3.	Registrar
4.	Add teachers
5.	Create group
6.	Present progress weekly
7.	Maintain progress
8.	Convert attendance to marks
9.	Add marks
10.	Post marks
11.	See marks
12.	Post an activity
13.	Update upcoming activity corner

General Classification:

Candidate classes were then characterized in seven general classes. The seven general characteristics are as follows:

1. External entities
2. Things
3. Events
4. Roles
5. Organizational units
6. Places
7. Structures

Potential nouns to become a class after general classification

SL	Nouns	General Classification
1.	Batch	5,7
2.	Project-Coordinator	4,5,7
3.	Teacher	4,5,7
4.	CR	4,5,7
5.	Student	4,5,7
6.	Roll-number	2
7.	Name	2
8.	ID	2
9.	Supervisor	4
10	Group	5, 7
11	Title	2
12	Attendance	2
13	Project-proposal-presentation	2
14	Mid-presentation	2
15	Final-presentation	2
16	Code-review	2
17	Project-showcasing	2
18	Final-report	2
19	Rest	2
20	Marks	2, 3 ,7
21	Notices	2
22	Activity	2, 3, 7
23	Facebook	1, 2

24	Code	1, 2
25	Reports	2
26	GitHub-link	1, 2

Selection Criteria:

The candidate classes are then selected as classes by six Selection Criteria. A candidate class generally becomes a class when it fulfills around three characteristics.

1. Retain information
2. Needed services
3. Multiple attributes
4. Common attributes
5. Common operations
6. Essential requirements

Potential general classified nouns to become a class after selection criteria:

SL	Nouns	Selection Criteria
1.	Batch	1, 2,
2.	Project-Coordinator	1-6(selected)
3.	CR	1-6(selected)
4.	Student	1-6(selected)
5.	Roll-number	1, 2
6.	Name	1, 2
7.	ID	1, 2
8.	Supervisor	1-6(selected)
9.	Group	1-6(selected)
10	Title	1, 2

11	Attendance	1, 2
12	Project-proposal-presentation	1, 2
13	Mid-presentation	1, 2
14	Final-presentation	1, 2
15	Code-review	1, 2
16	Project-showcasing	1, 2
17	Final-report	1, 2
18	Rest	1, 2
19	Marks	1-6(selected)
20	Notices	1
21	Activity	1-6(selected)
22	Facebook	6
23	Code	1, 2, 6
24	Reports	1, 2 ,6
25	GitHub-link	6

Selected Classes

1. Project Coordinator
2. CR
3. Student
4. Supervisor
5. Group
6. Marks
7. Posts
8. Facebook Group
9. GitHub Link

Attribute and Method Identification

Class Name	Attribute	Method
Student	-student_roll -student_name -group_name -marks	+present_progress() +see_marks() +see_notice()
Project Coordinator	-pc_id -name	+register() +add_CR() +create_group() +add_group_details() +add_supervisor() +maintain_progress_as_attendace() +post_marks(type)
CR	-roll_number -cr_id	+add_student() +post_activity_notice() +add_github_link()
Supervisor	-supervisor_id -supervisor_name -group_name	+give_marks()
Group	-group_name -project_title -supervisor_name	+store_progress() +show_marks()
Marks	-student_roll -project_proposal -mid_presentation -final_presentation -code_review -project_showcasing -supervisor_marks -final_report	+convert_attendance() +total_marks_tabulatation()
Posts	-post_id -post _detail	+show_all_posts() +post_to_Facebook()

		+update_activity_corner()
Facebook Group	-post_id -post_details	+fetch_notices()
Github Link	-github_id -project_link	-generate_link() +send_group_link()

Class Cards

Class: Student	
Responsibilities	Collaborators
Present Progress	Group, Project Coordinator
See Marks	Group, Marks
See Posts	Post

Class: Project Coordinator	
Responsibilities	Collaborators
Register	
Add CR	CR
Create Group	Group
Add Group Details	Group
Add Supervisor	Group, Supervisor
Maintain Progress as Attendance	Marks, Student
Post Marks	Group, Marks

Class: CR	
Responsibilities	Collaborators
Add Classmates	Student
Post Activity Posts	Posts

Class: Supervisor	
Responsibilities	Collaborators
Give Marks	Marks

Class: Group	
Responsibilities	Collaborators
Store Progress	Student, Mark, Project Coordinator

Class: Marks	
Responsibilities	Collaborators
Convert Attendance Marks	Project Coordinator
Project Proposal	Student, Project Coordinator
Mid Presentation	Student, Project Coordinator
Final Presentation	Student, Project Coordinator
Code Review	Student, Project Coordinator
Project Showcasing	Student, Project Coordinator
Final Report	Student, Project Coordinator
Add Marks from Supervisor	Supervisor, Student
Generate Final Marks	
Send Final Marks	Group, Student

Class: Posts	
Responsibilities	Collaborators
Show all Posts	CR
Post to Facebook Group	Facebook Group
Update Upcoming Activity Corner	
Add GitHub Links of the Groups	GitHub Link

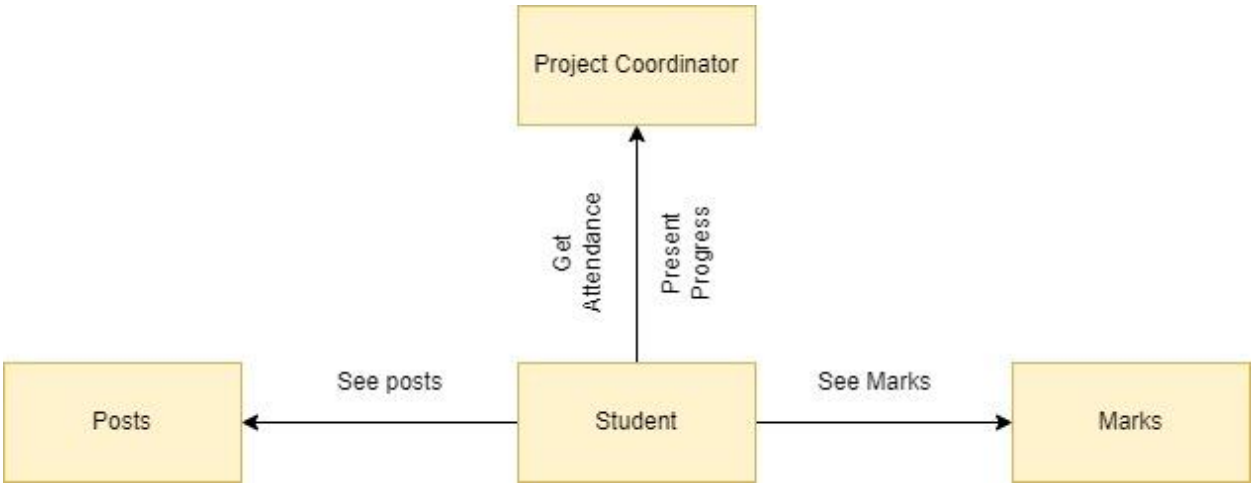
Class: Facebook Group	
Responsibilities	Collaborators
Post the Notices	Posts

Class: GitHub	
Responsibilities	Collaborators
Generate GitHub Link of the Group Project	Group
Send the Link	Posts

CRC Diagram

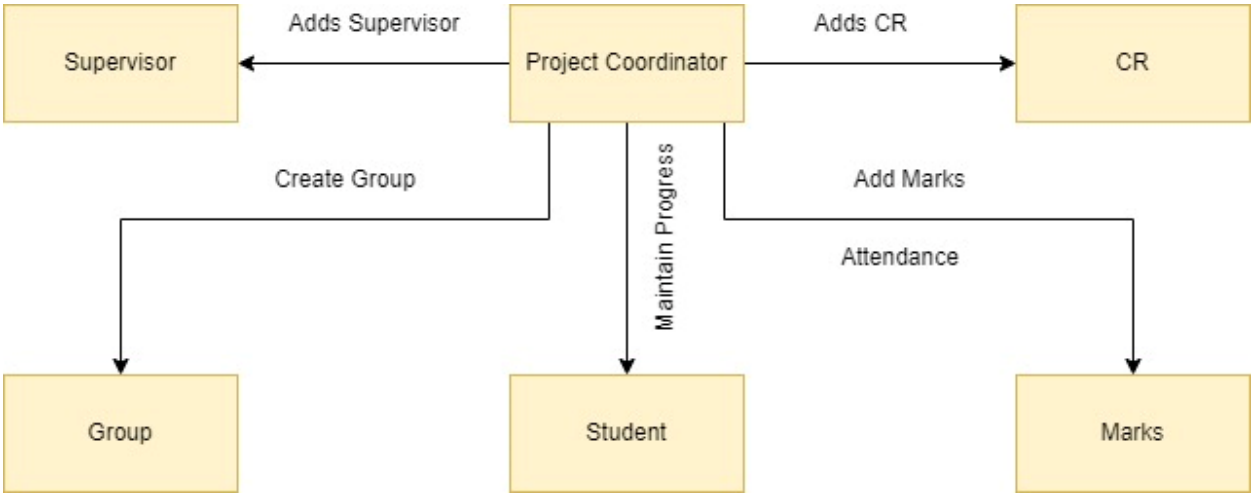
ID: 1

Name: Student



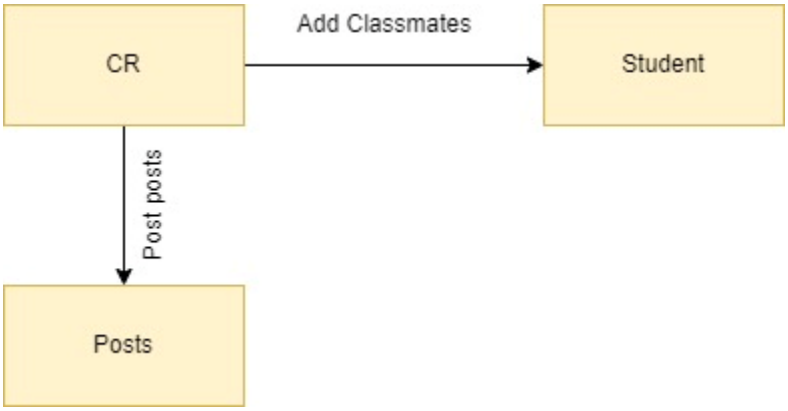
ID: 2

Name: Project Coordinator



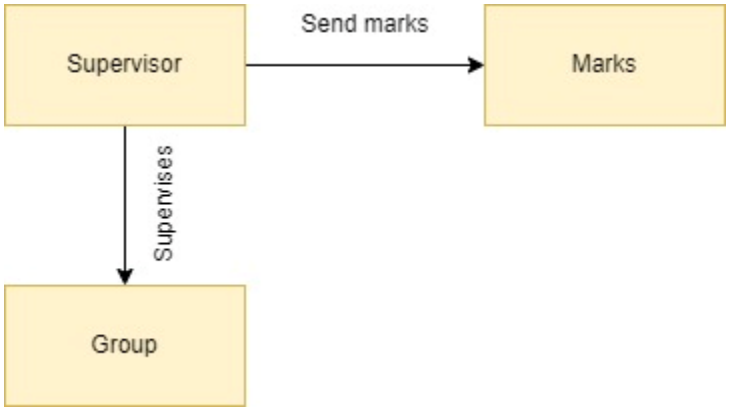
ID: 3

Name: CR



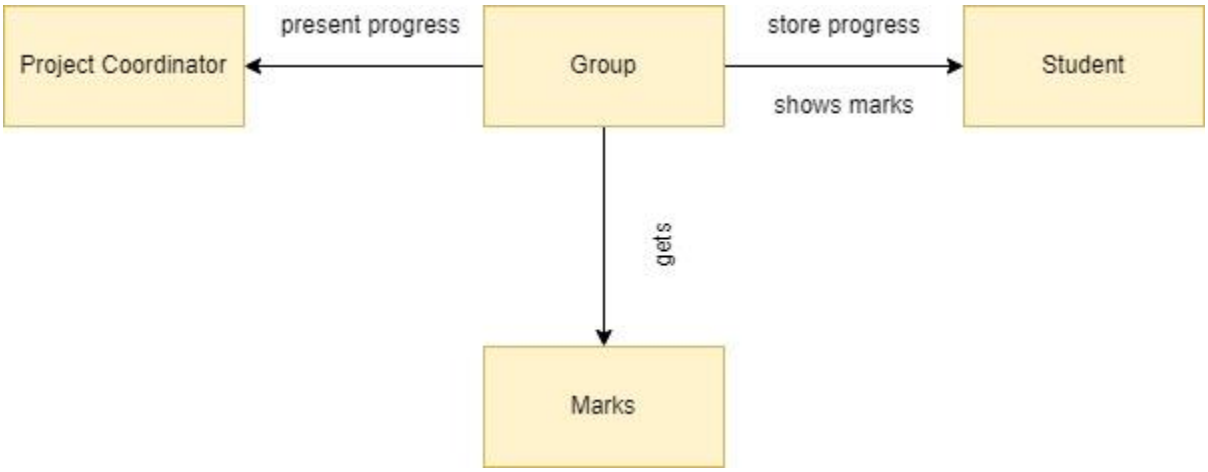
ID: 4

Name: Supervisor



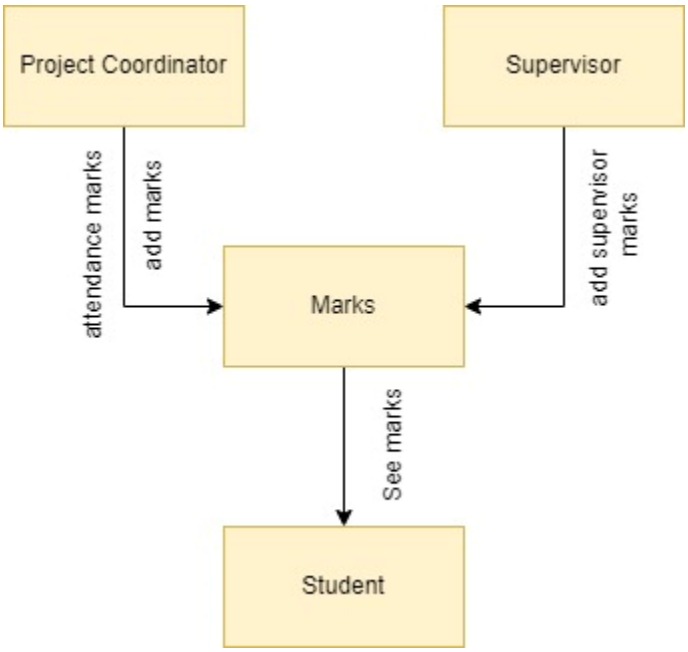
ID: 5

Name: Group



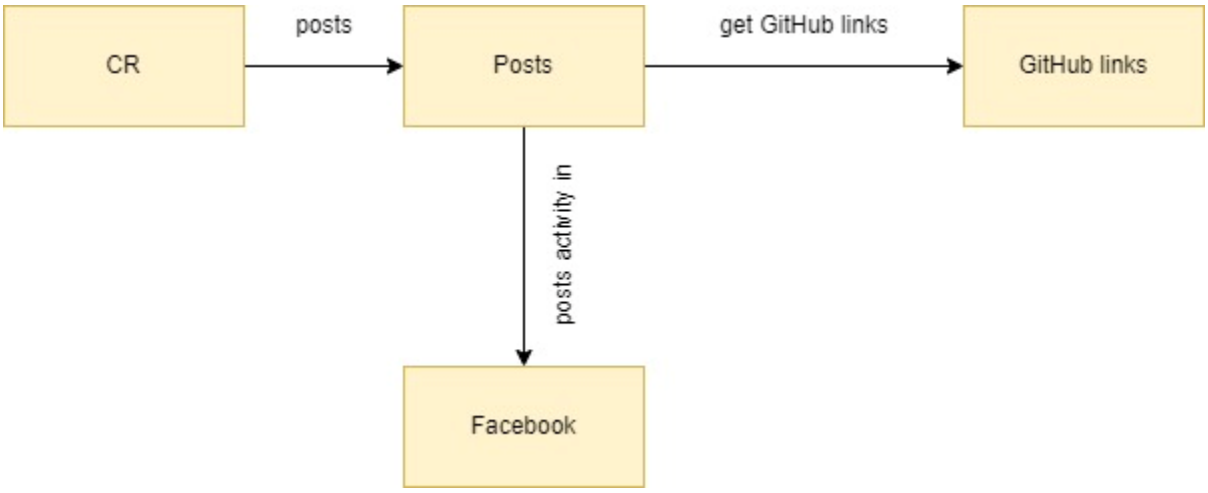
ID: 6

Name: Marks



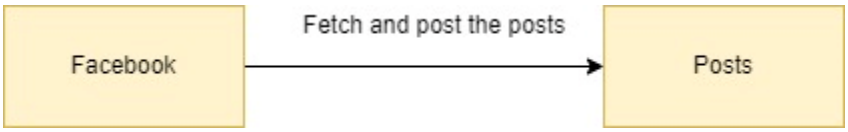
ID: 7

Name: Posts



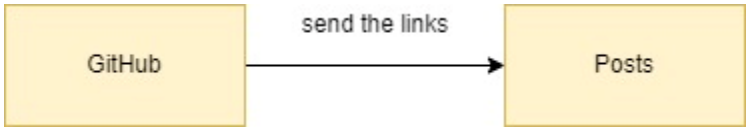
ID: 8

Name: Facebook



ID: 9

Name: GitHub



Behavioral Modeling

The behavioral model indicates how software will respond to external events or stimuli. In the context of behavioral modeling, two different characterizations of states must be considered: (1) the state of each class as the system performs its function and (2) the state of the system as observed from the outside as the system performs its function.

State Transition Diagram

One component of a behavioral model is a UML state diagram that represents active states for each class and the events (triggers) that cause changes between these active states.

Event Table

Serial No	Event	Event Name	Initiator	Collaborator	Associated Methods
1.	Register as Admin	Registrar	Project Coordinator		-register()
2.	Add CR	Add_CR	Project Coordinator	CR	+add_CR()
3.	Add Supervisor	Add_Supervisor	Project Coordinator	Supervisor, Group	+add_supervisor()
4.	Add Students	Add_Student	CR	Student	+add_student()
5.	Create Group	Create_Group	Project Coordinator	Group	+create_group()
6.	Add Group Details	Add_Group_Detail	Project Coordinator	Group	+add_group_details()
7.	Present Progress	Present_Progress	Student	Group, Project Coordinator	+present_progress()

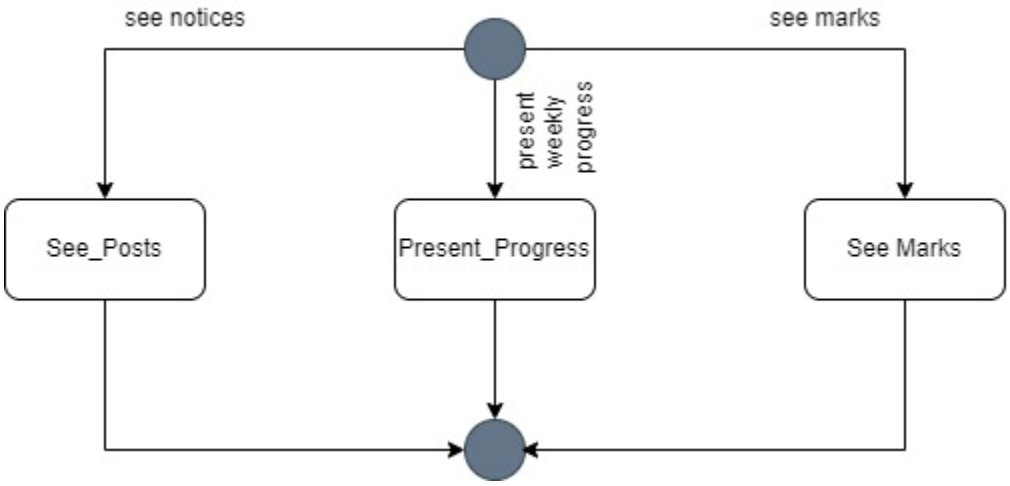
8.	Store Progress	Store_Progress	Group	Student, Project Coordinator	+store_progress()
9.	Maintain Progress as Attendance	Maintain_Progress_as_Attendance	Project Coordinator	Student, Group, Marks	+maintain_progress_as_attendance()
10.	Give Marks	Give_Marks	Supervisor	Group, Marks	+give_marks()
11.	Add Marks	Add_Marks	Project Coordinator	Marks	+add_marks()
12.	Convert Attendance Marks	Con_Att_Marks	Marks	Project Coordinator, Student	+convert_attendance()
13.	Project Proposal	Project_Proposal	Marks	Project Coordinator, Student	+project_proposal()
14.	Mid Presentation	Mid_Presentation	Marks	Project Coordinator, Student	+mid_presentation()
15.	Final Presentation	Final_Presentation	Marks	Project Coordinator, Student	+final_presentation()
16.	Code Review	Code_Review	Marks	Project Coordinator, Student	+code_review()
17.	Project Showcasing	Project_Showcasing	Marks	Project Coordinator, Student	+project_showcasing()
18.	Final Report	Final_Report	Marks	Project Coordinator, Student	+final_report()
19.	Add Marks from Supervisor	Add_Supr_Marks	Marks	Supervisor	+add_supervisor_marks()
20.	Generate Final Marks	Generate_Fmarks	Marks		-gen_final_marks()

21.	Send Final Marks	Send_Fmarks	Marks	Group, Student, Project Coordinator	+send_final_marks()
22.	Post Marks	Post_Marks	Project Coordinator	Group, Marks	+post_marks()
23.	See Marks	See_Marks	Student	Group, Marks	+see_marks()
24.	Show Marks	Show_Marks	Group	Marks, Student	+show_marks()
25.	Post Activity Notices	Post_Act_Notice	CR	Notice	+post_activity_notice()
26.	See Notice	See_Posts	Student	Notice	+see_notice()
27.	Show All Posts	Show_Posts	Posts	CR	+show_all_posts()
28.	Post to Facebook Group	Post_to_FB	Posts	Facebook	+post_to_facebook()
29.	Update Upcoming Activity Corner	Update_Activity_Corner	Posts	CR	+update_activity_corner()
30.	Add GitHub Links to the Groups	Add_Git_Links	Posts	GitHub	+add_github_link()
31.	Post the Posts	Post_Posts	Facebook	Posts	+fetch_notices()
32.	Generate GitHub Link	Gen_GitHub_Link	GitHub	Group	-generate_link()
33.	Send GitHub Link	Send_GitHub_Link	GitHub	Notice	+send_group_link()

State Transition Diagram

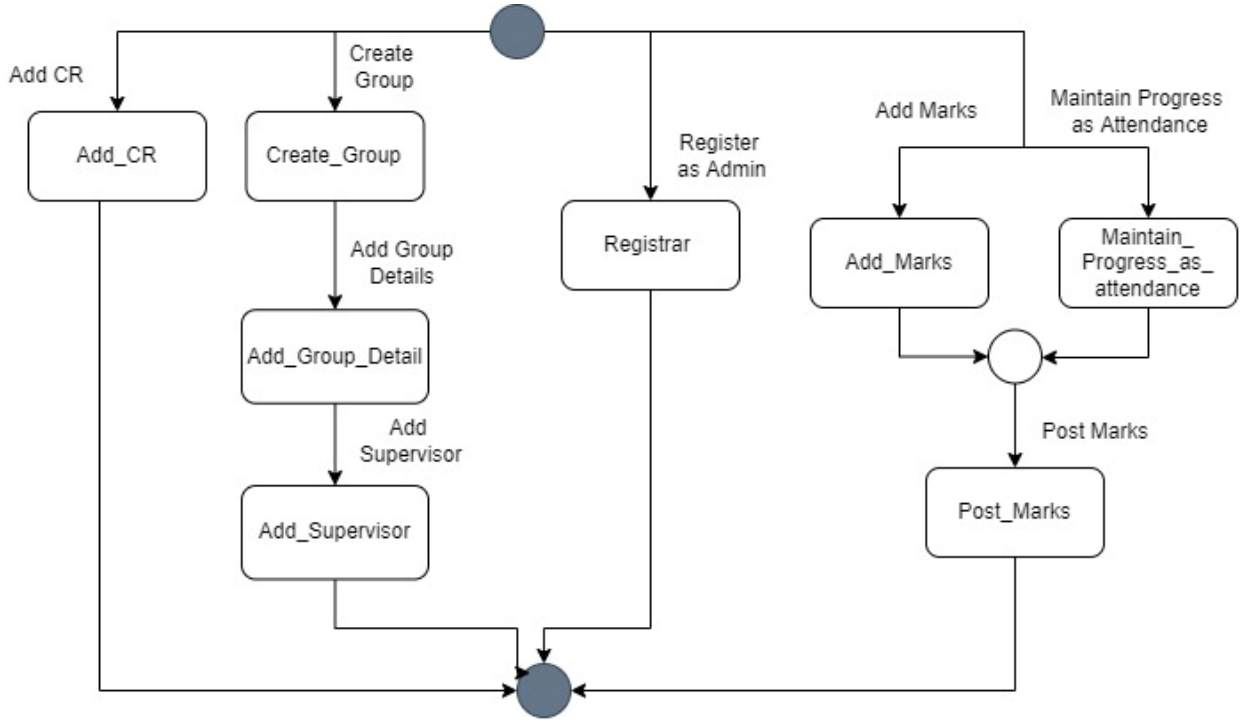
ID: 1

Name: Student



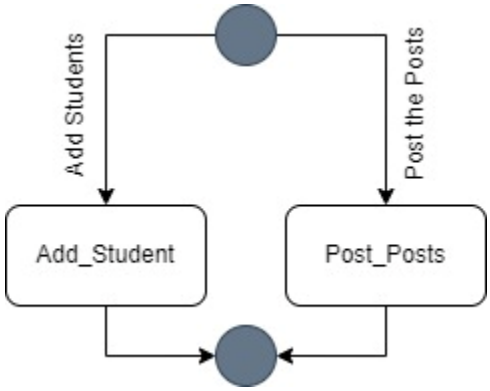
ID: 2

Name: Project Coordinator



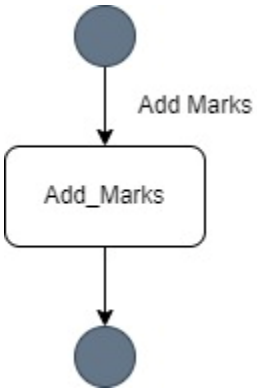
ID: 3

Name: CR



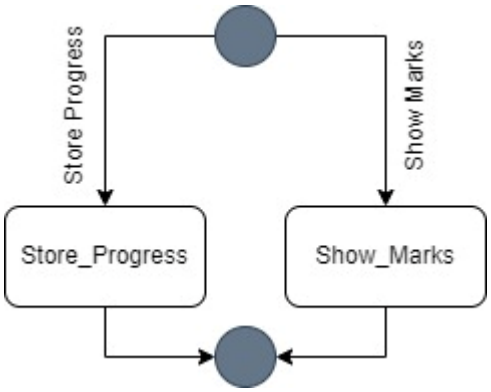
ID: 4

Name: Supervisor



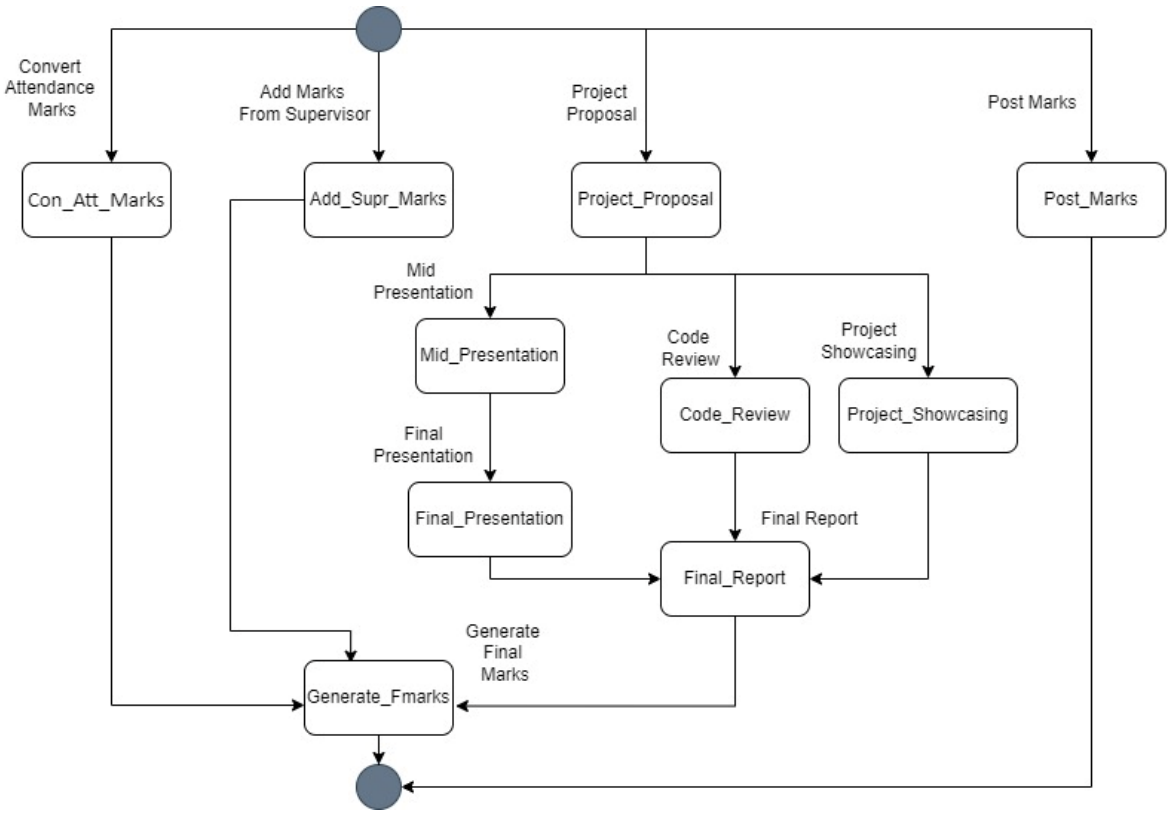
ID: 5

Name: Group



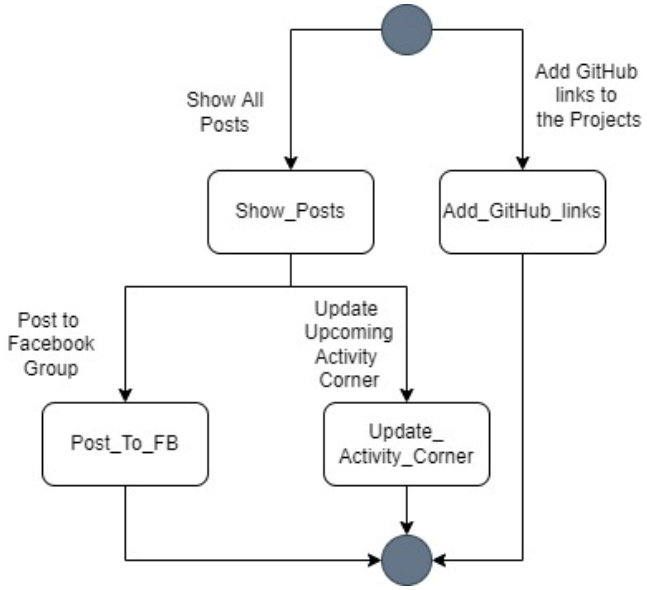
ID: 6

Name: Marks



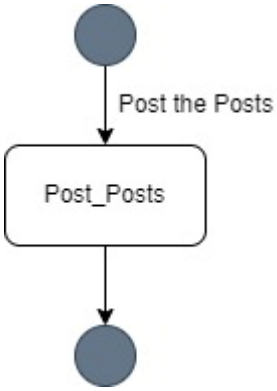
ID: 7

Name: Posts



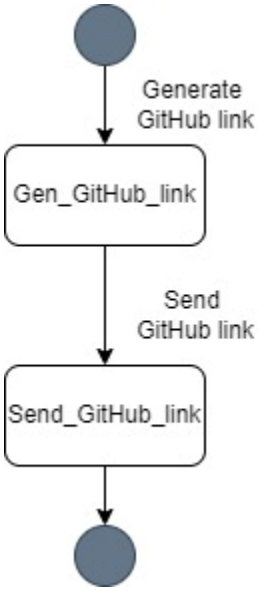
ID: 8

Name: Facebook



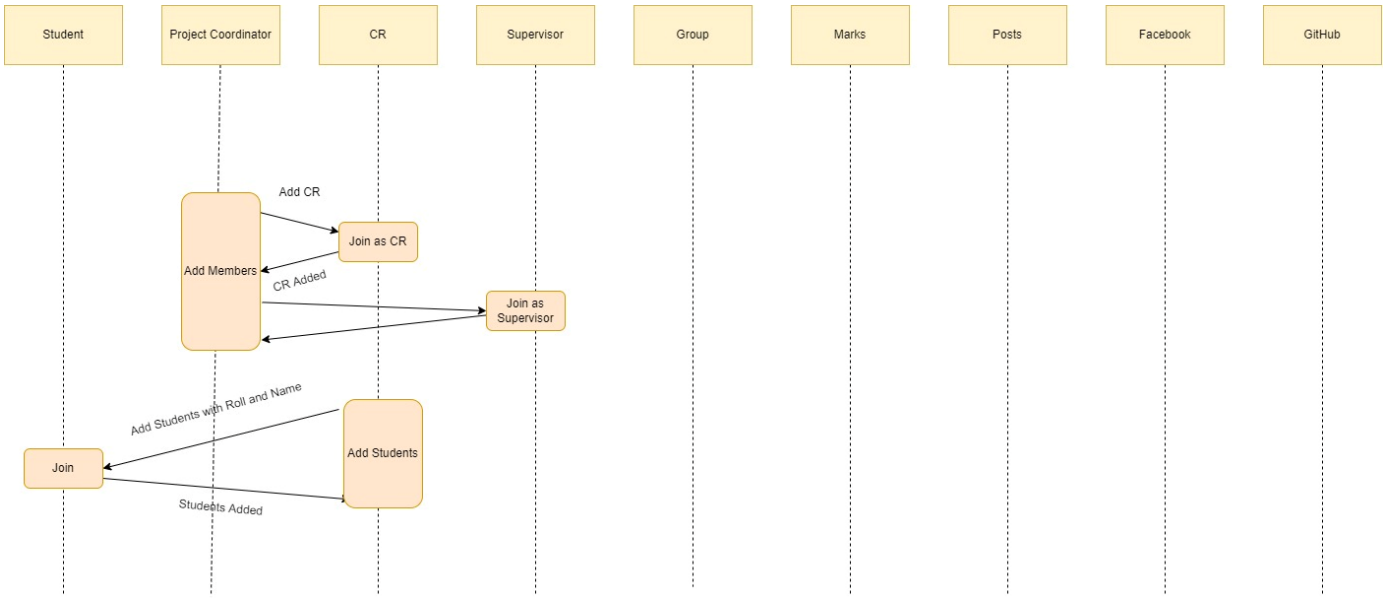
ID: 9

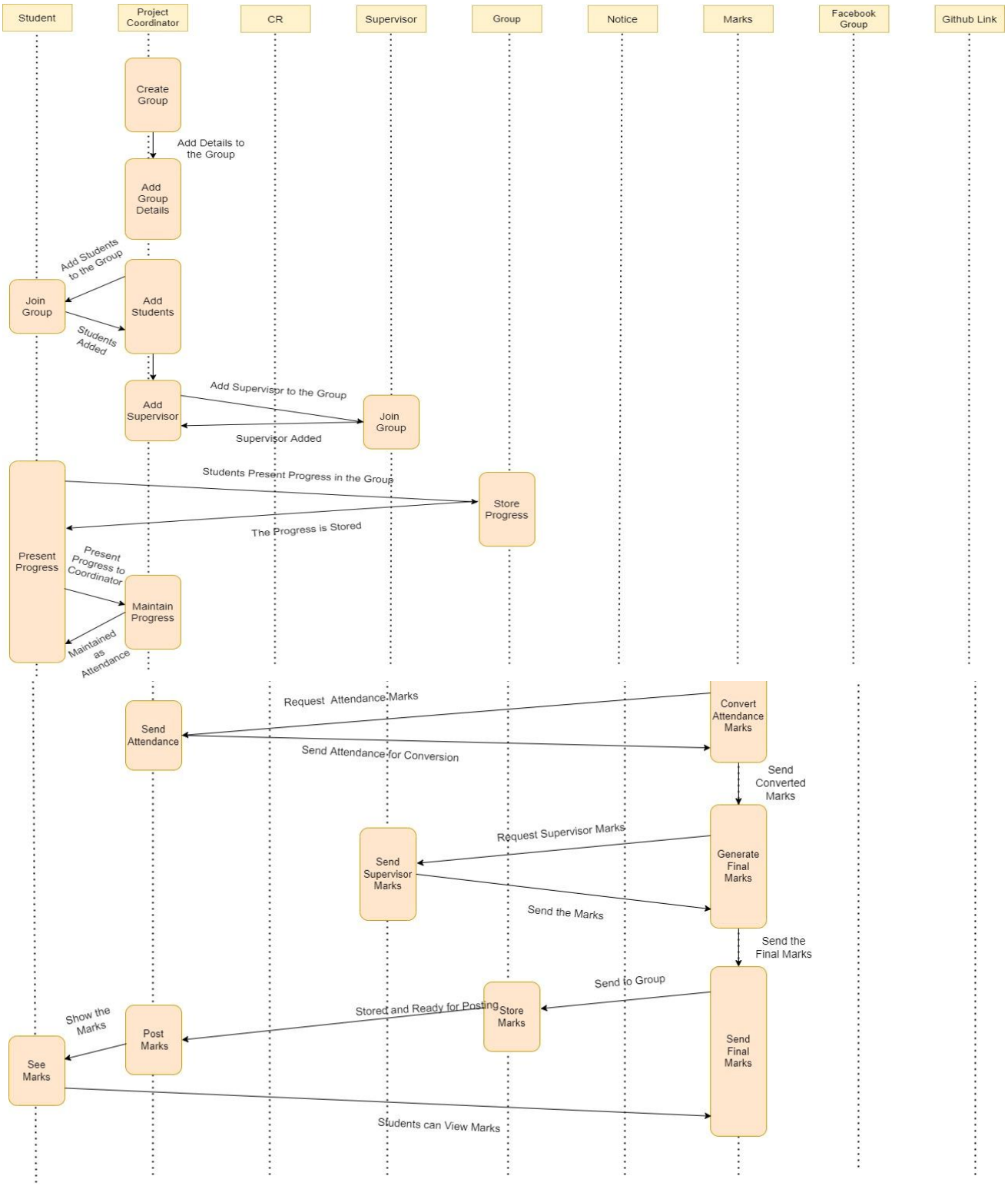
Name: GitHub

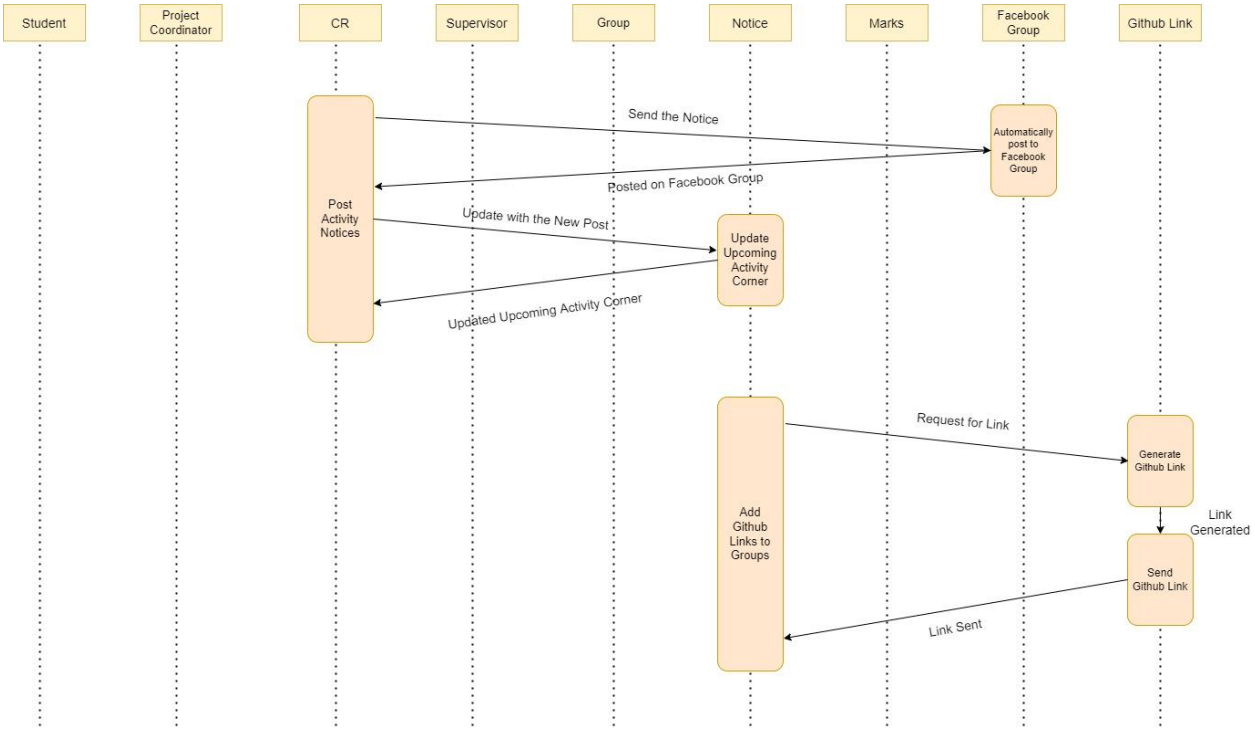


Sequence Diagram

The second type of behavioral representation, called a sequence diagram in UML, is a representation of how events cause flow from one object to another as a function of time. In essence, the sequence diagram is a shorthand version of the use case. It represents key classes and the events that cause behavior to flow from class to class.




Student
Project Coordinator
CR
Supervisor
Group
Notice
Marks
Facebook Group
Github Link
Create Group
Add Details to the Group
Add Group Details
Add Students to the Group
Join Group
Add Students
Students Added
Add Supervisor
Add Supervisor to the Group
Join Group
Supervisor Added
Students Present Progress in the Group
Store Progress
The Progress is Stored
Present Progress
Present Progress to Coordinator
Maintain Progress
Maintained as Attendance
Request Attendance Marks
Convert Attendance Marks
Send Attendance
Send Attendance for Conversion
Send Converted Marks
Request Supervisor Marks
Send Supervisor Marks
Generate Final Marks
Send the Marks
Send the Final Marks
Send to Group
Store Marks
Stored and Ready for Posting
Show the Marks
Post Marks
See Marks
Send Final Marks
Students can View Marks


Student
Project Coordinator
CR
Supervisor
Group
Notice
Marks
Facebook Group
Github Link
Send the Notice
Automatically post to Facebook Group
Posted on Facebook Group
Post Activity Notices
Update with the New Post
Update Upcoming Activity Corner
Updated Upcoming Activity Corner
Request for Link
Generate Github Link
Link Generated
Add Github Links to Groups
Send Github Link
Link Sent