Software Requirement Specification

SPMA

Student Project Management Application

SE 406 : Software Requirement Specification

Submitted by

Md Siam

BSSE Roll No. : 1104

BSSE Session: 2018-19

Supervised by

Dr. Kazi Muheymin-Us-Sakib

Designation: Professor

Institute of Information Technology

Institute of Information Technology

University of Dhaka

31-10-2021

Use Case Diagram:

A Use Case describes the system behavior under various conditions as the system responds to a request from one of its stakeholders. In fact, a use case diagram is a kind of visualization of the system where an end-user has an idea of a specific feature. It 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 or systems 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 system operators. They procedure some information or consume some information. Every user has one or more goals when using the system.

Primary Actor
Primary actors interact directly to achieve the required system function and derive the intended benefit from the system. They work directly with the software. They produce some information and consume some information too.

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

Here is given the use case diagram to observe the non-technical view of the system.

Level: 0
Name: SPMA
Primary Actor:Teacher,Coordinator,Student,CR,Supervisor

Secondary Actor:Facebook,Github

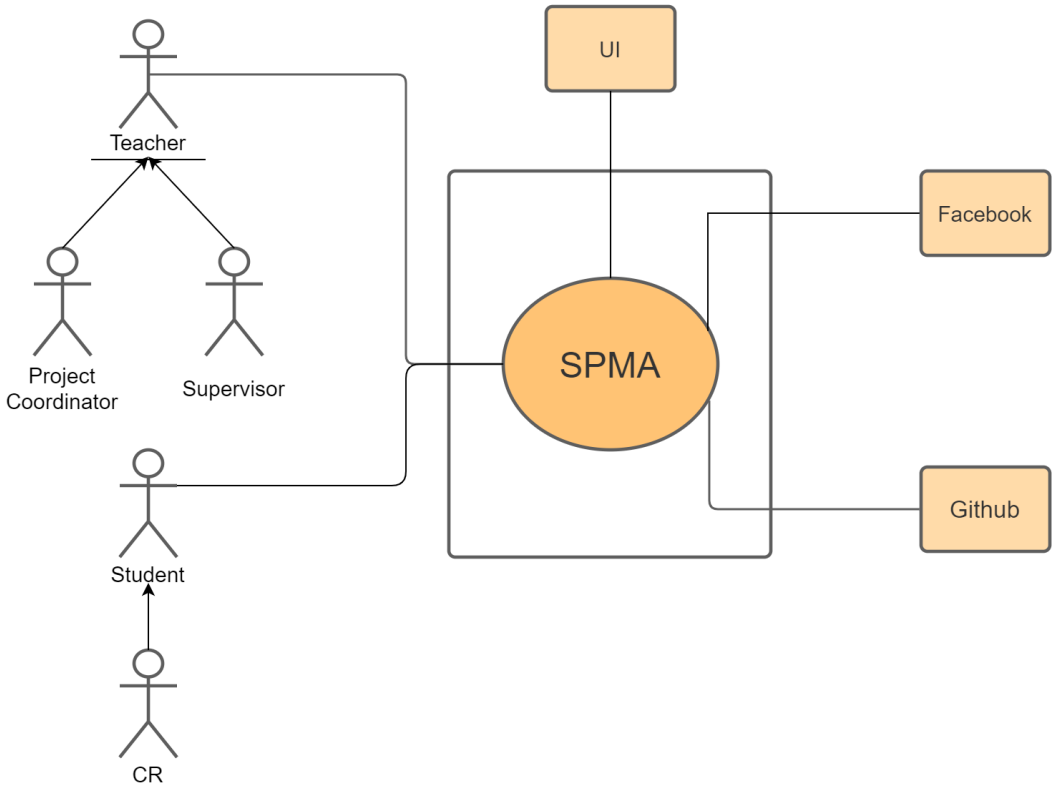


Figure 1: Software Project Management Application

Level: 1
Name: SPMA
Primary Actor:Teacher,Coordinator,Student,CR,Supervisor
Secondary Actor:Facebook,Github

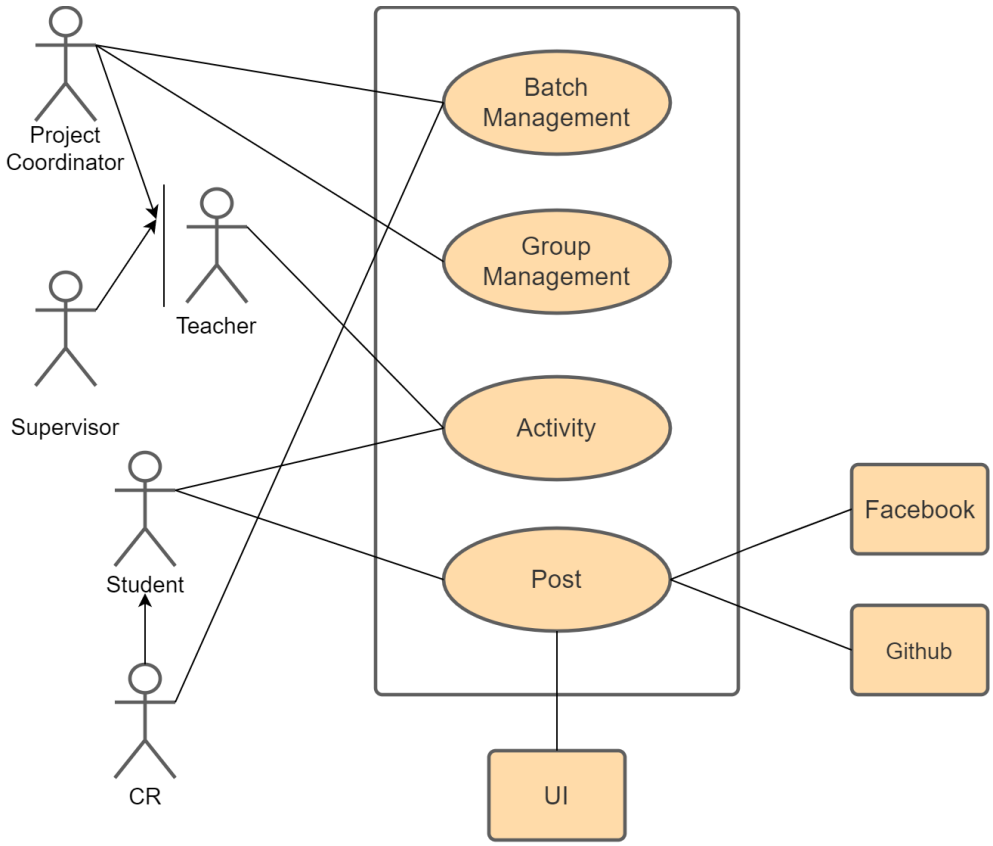


Figure 2: Software Project Management Application (Detailed)

Description of use case diagram level 1:

1.Batch Management: 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.. The admin will register 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 a group entity, and each group should be created by the Project Coordinator. Each group may contain one or multiple students.

3. Activity: Each of the groups is required 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 attendance 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 of 20 and Final report 20. Rest 20 will be given by the supervisor. Coordinator will post the marks from time to time. A student will be able to see her marks.

4. Post: 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). a github link for each of the groups will be available in the application. Any timely activity notice should also be on the upcoming activity corner.

Level 1.1:
Name: Batch Management
Primary Actor: Coordinator, CR
Secondary Actor : NA

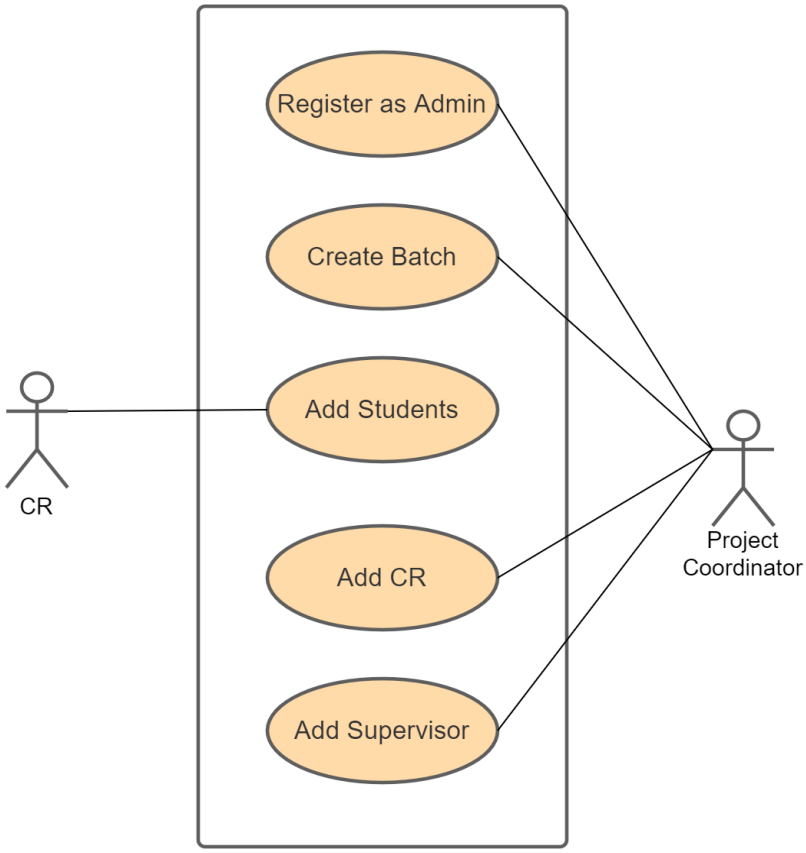


Figure 3: Batch Management

Description of use case diagram level 1.1:

Register as Admin:

The admin will register him/herself by giving his/her Name and ID.

Add Supervisor:

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

Add CR:

The Project Coordinator, an assigned teacher, also the admin of this application, should add one or more students to the application, called CR.

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).

Create Batch:
Admins can create a batch.

Level 1.2:
Name: Group Management
Primary Actor: Admin,Student,Supervisor
Secondary Actor: None

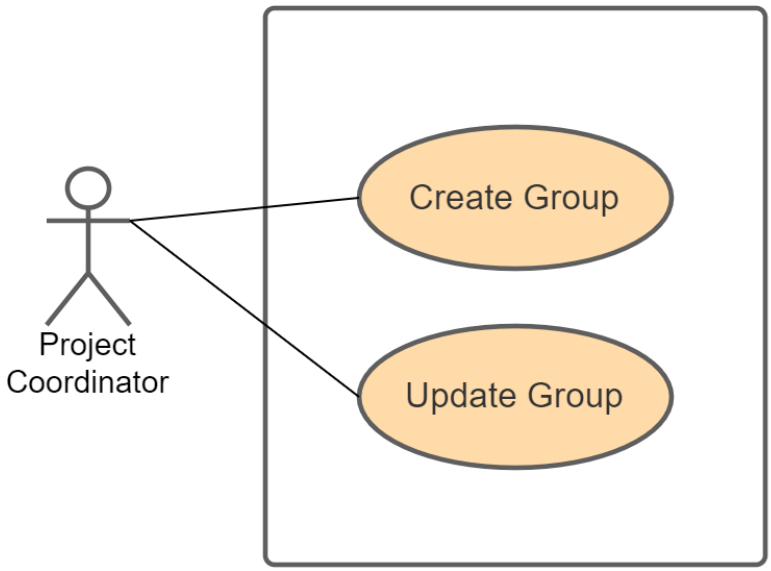


Figure 4: Group Management

Description of use case diagram level 1.2:

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

Update Group : The students will perform their assigned activities and the supervisor will guide and maintain the group.

Level 1.3:
Name: Activity
Primary Actor: Coordinator,Teacher,Supervisor
Secondary Actor : None

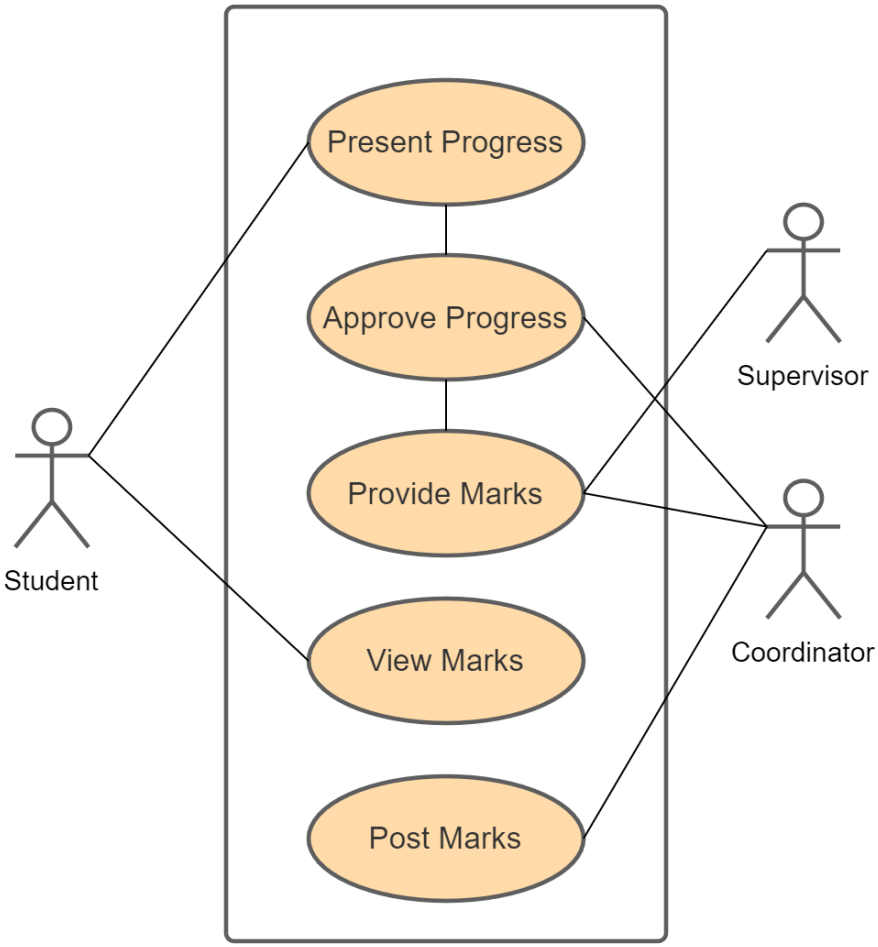


Figure 5: Activity

Description of use case diagram level 1.3:
Present Progress : Each of the group requires to present their progress weekly to the coordinator. Coordinator will maintain the progress
Approve Progress : Coordinator will approve or deny the progress of the student
View Marks : students can view their marks
Provide Marks : teachers will provide marks

Post Marks : The Project Coordinator will post the marks and the student will be able to check it.

Level 1.3.1:
Name: Approve Progress
Primary Actor: Coordinator
Secondary Actor:None

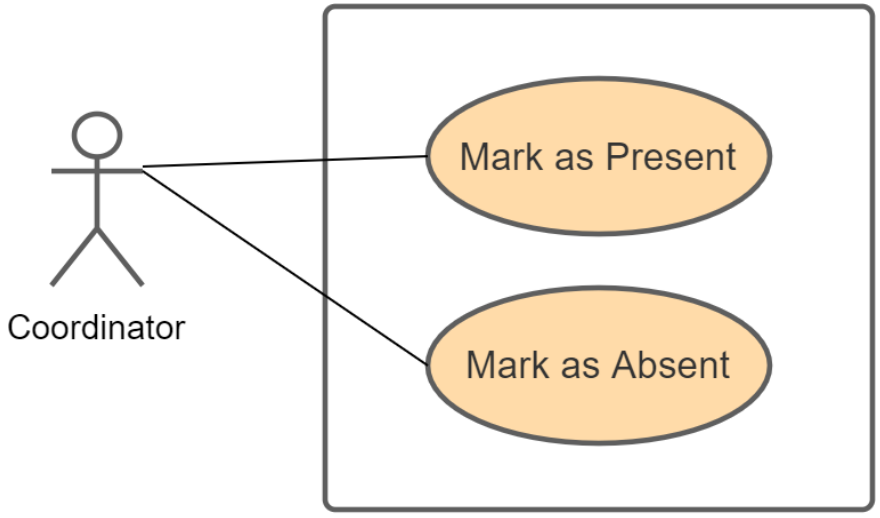


Figure 6: Approve Progress

Description of use case diagram level 1.3.1:
Mark as present : if progress is satisfactory
Mark as absent: if progress is not satisfactory

Level 1.4:
Name: Post
Primary Actor: student**,**CR
Secondary Actor:Facebook,Github, UI

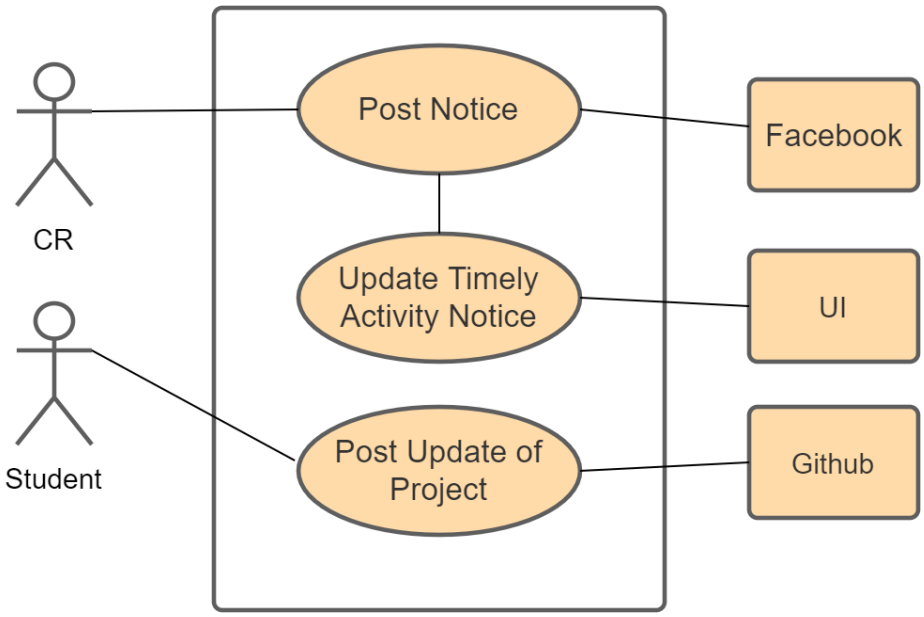


Figure 7: Post

Post Notice : All the activity related notices will be posted by the CR. All the notices should automatically be posted on the Facebook group.
Post update of project : All project related data that is code and reports should be kept in the github. The github link for each of the
groups will be available in the application.
Update Timely Activity Notice: Once the notice have been posted, UI should update automatically.

Activity Diagram:

Level 1:

Name: Software Project Management Application

Reference: Use case Diagram level-1

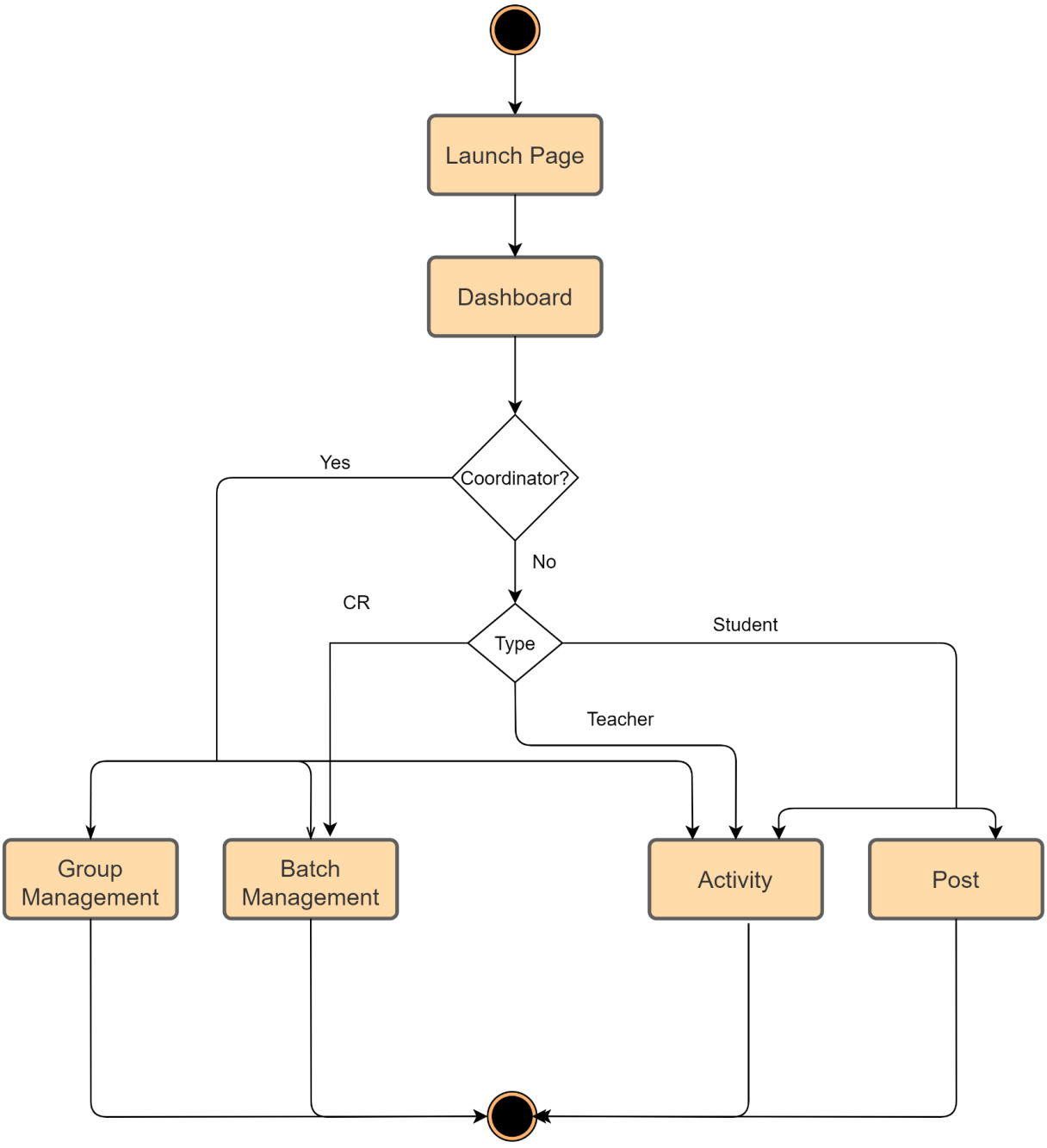


Figure 14: SPMA (Use Case - 1)

Level 1.1:

Name: Batch Management

Reference: Use case Diagram level-1.1

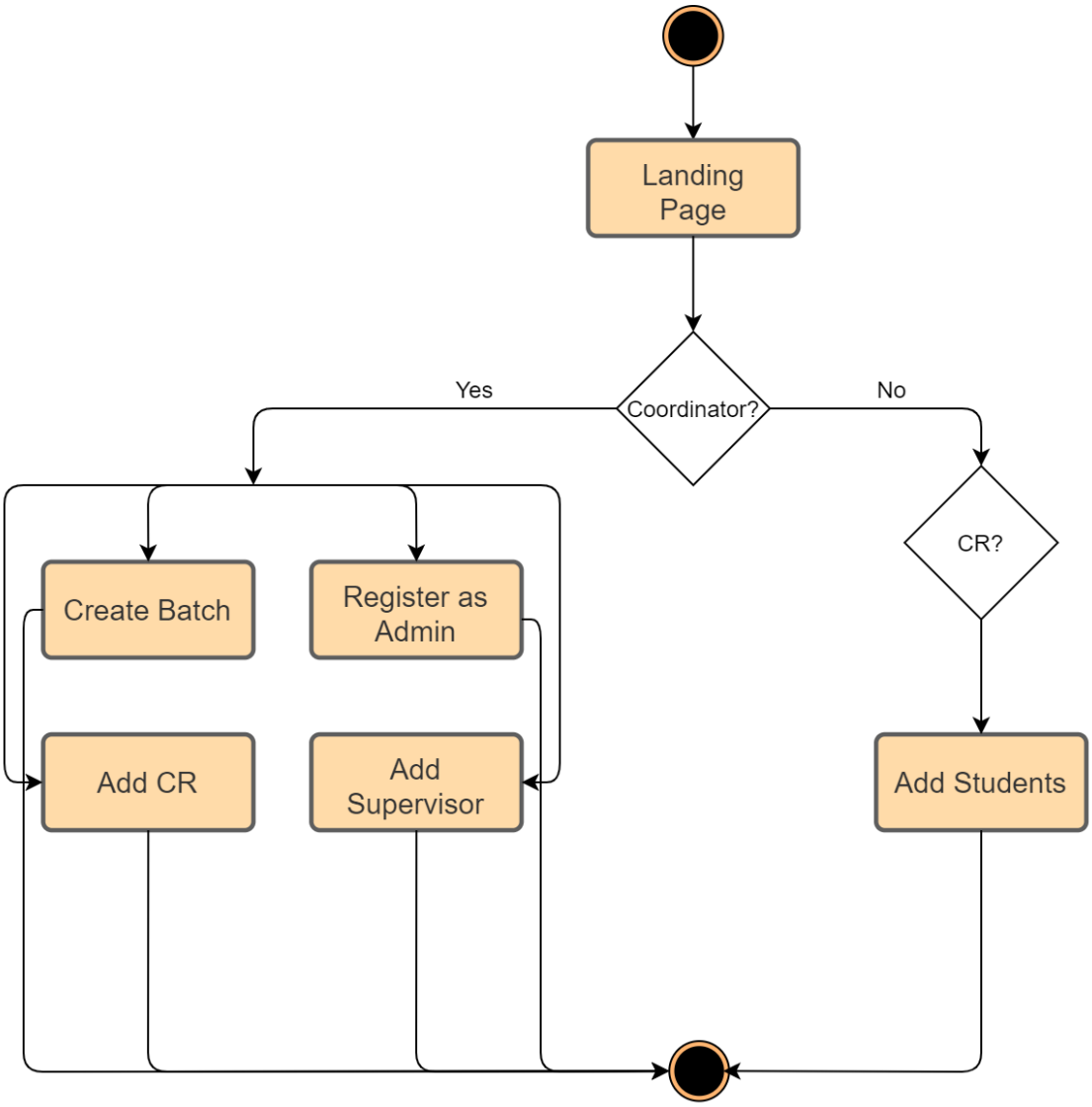


Figure 9: Batch Management (Use Case - 1.1)

Level 1.2:
Name: Group Management
Reference: Use case Diagram level-1.2

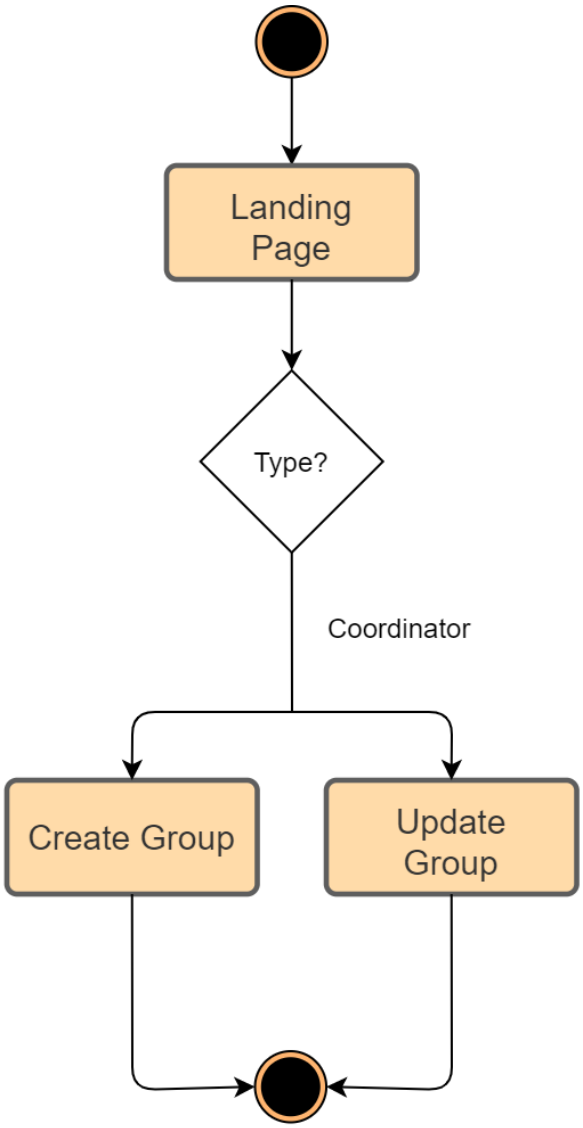


Figure 10: Group Management (Use Case - 1.2)

Level 1.3:
Name: Activity and Marks distribution
Reference: Use case Diagram level-1.3

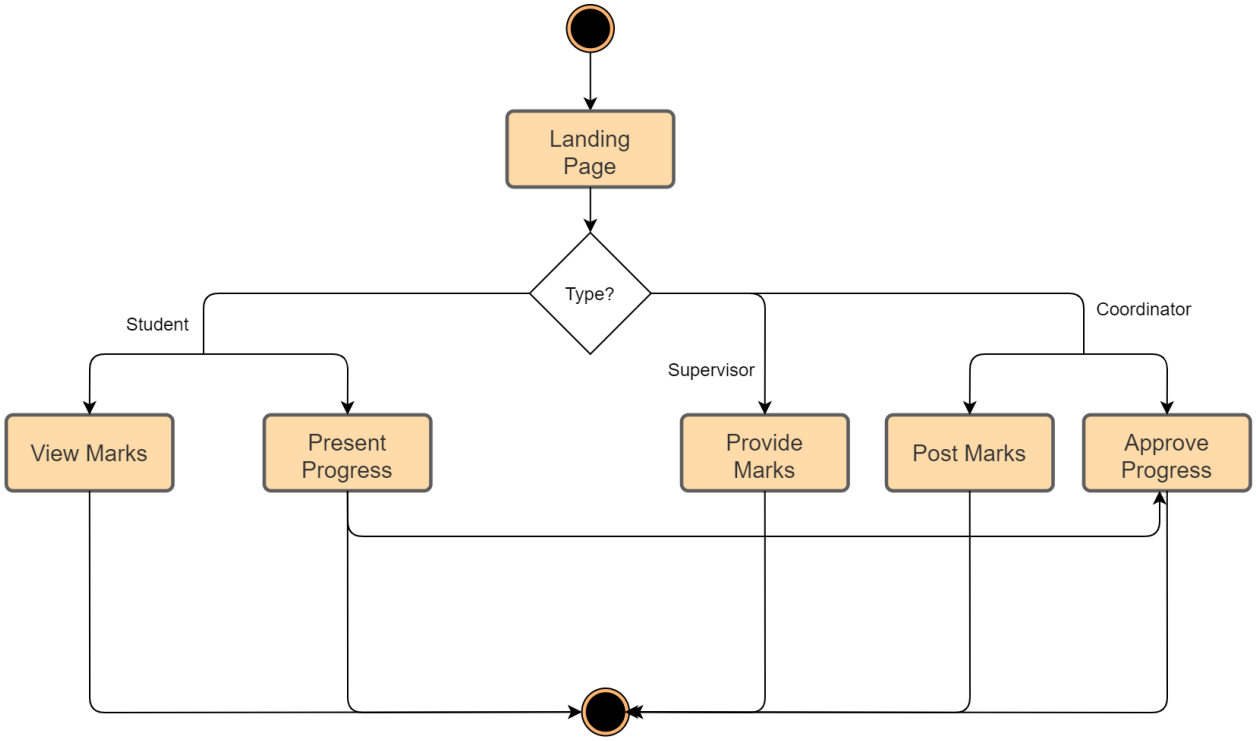


Figure 11: Activity (Use Case - 1.3)

Level 1.3.1:
Name: Approve Progress
Reference: Use case Diagram level-1.3

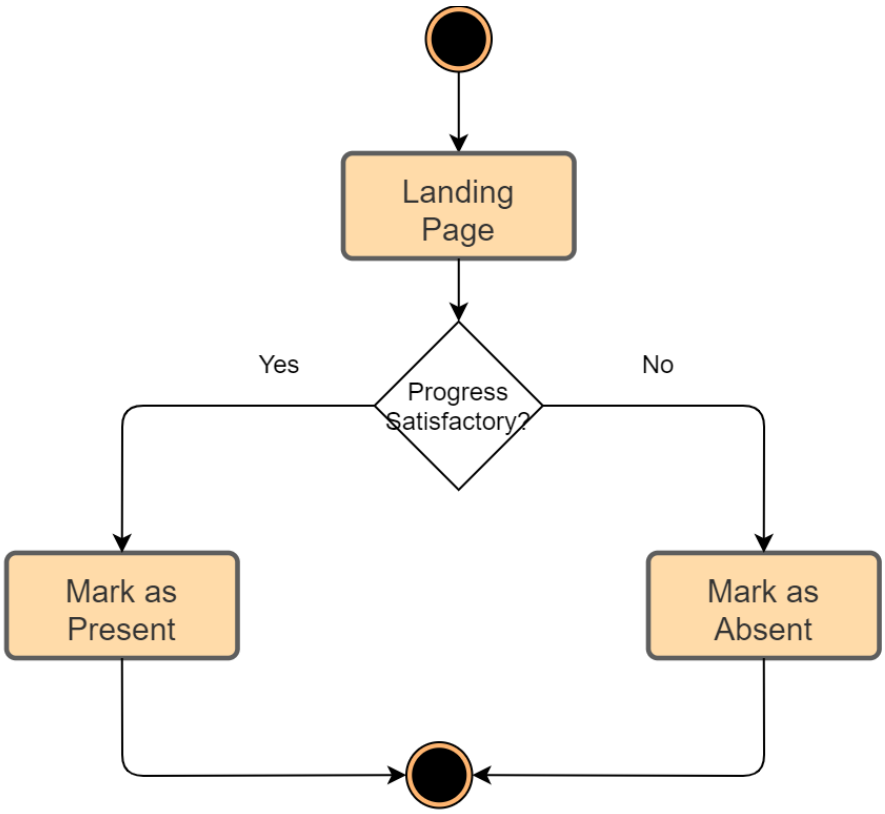


Figure 12: Approve Progress (Use Case - 1.3.1)

Level 1.4:

Name: Post

Reference: Use case Diagram level-1.4

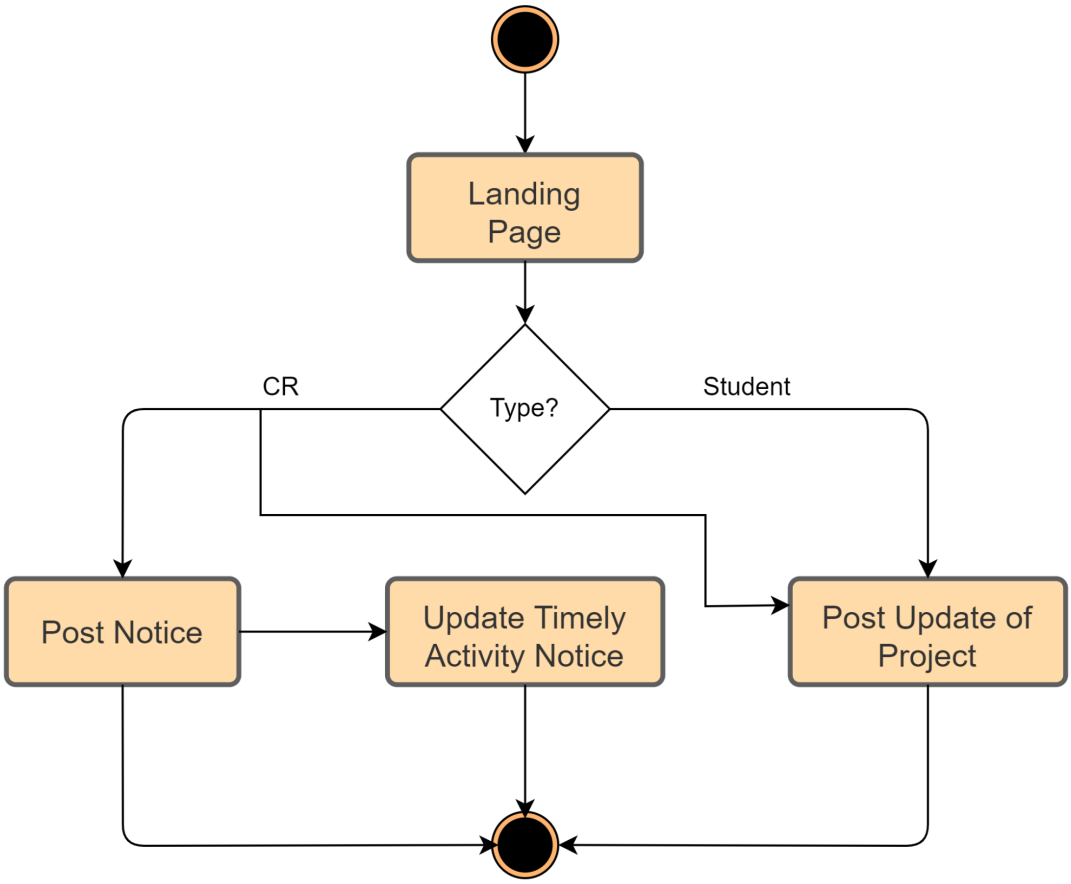


Figure 13: Post (Use Case - 1.4)

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 swimlane 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 : SPMA

Reference: Use case & Activity diagram level-1

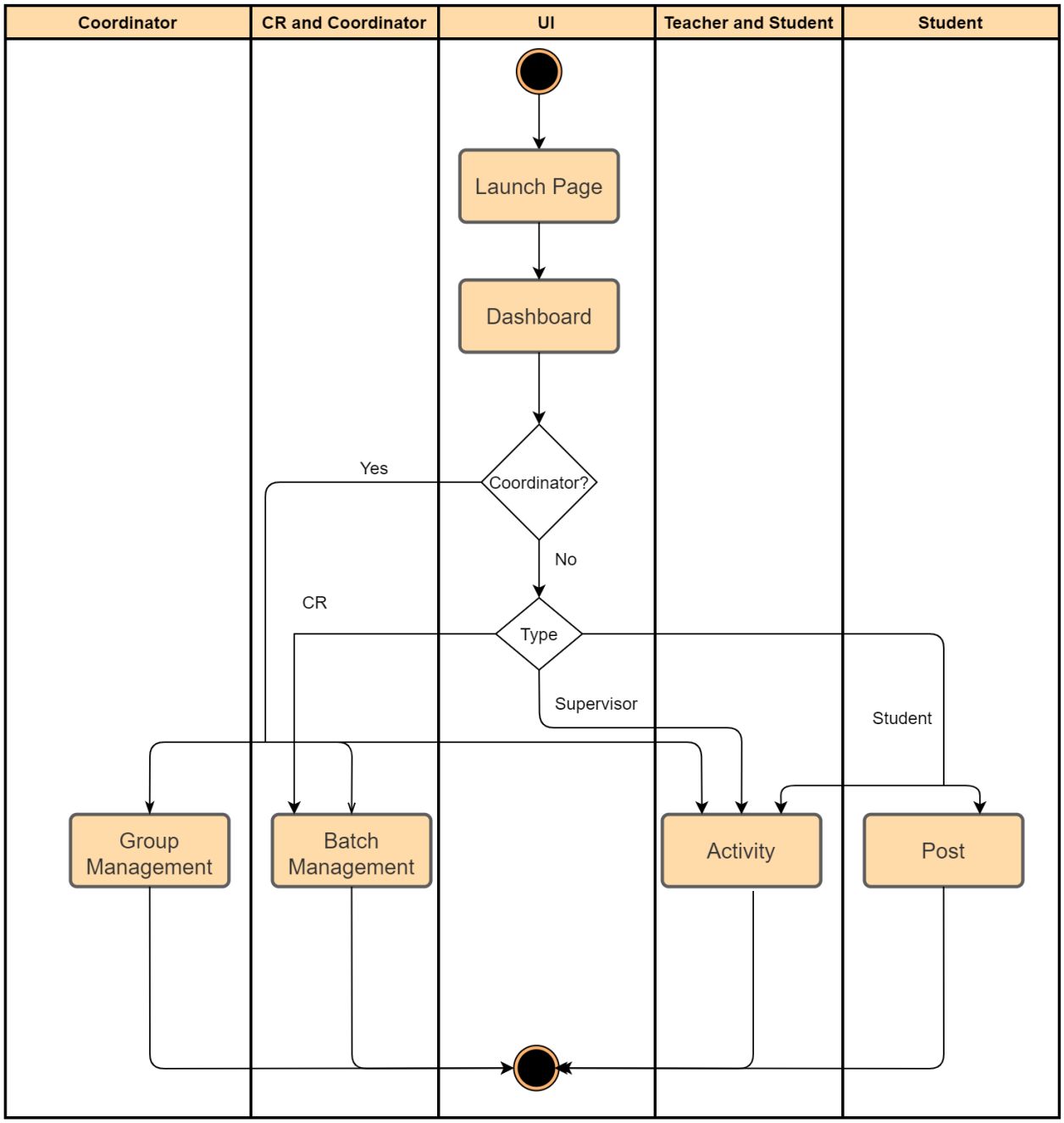


Figure 14: SPMA (Swimlane - 1)

SID (Swimlane ID): 1.1
Name : Batch Management
Reference: Use case & Activity diagram level-1.1

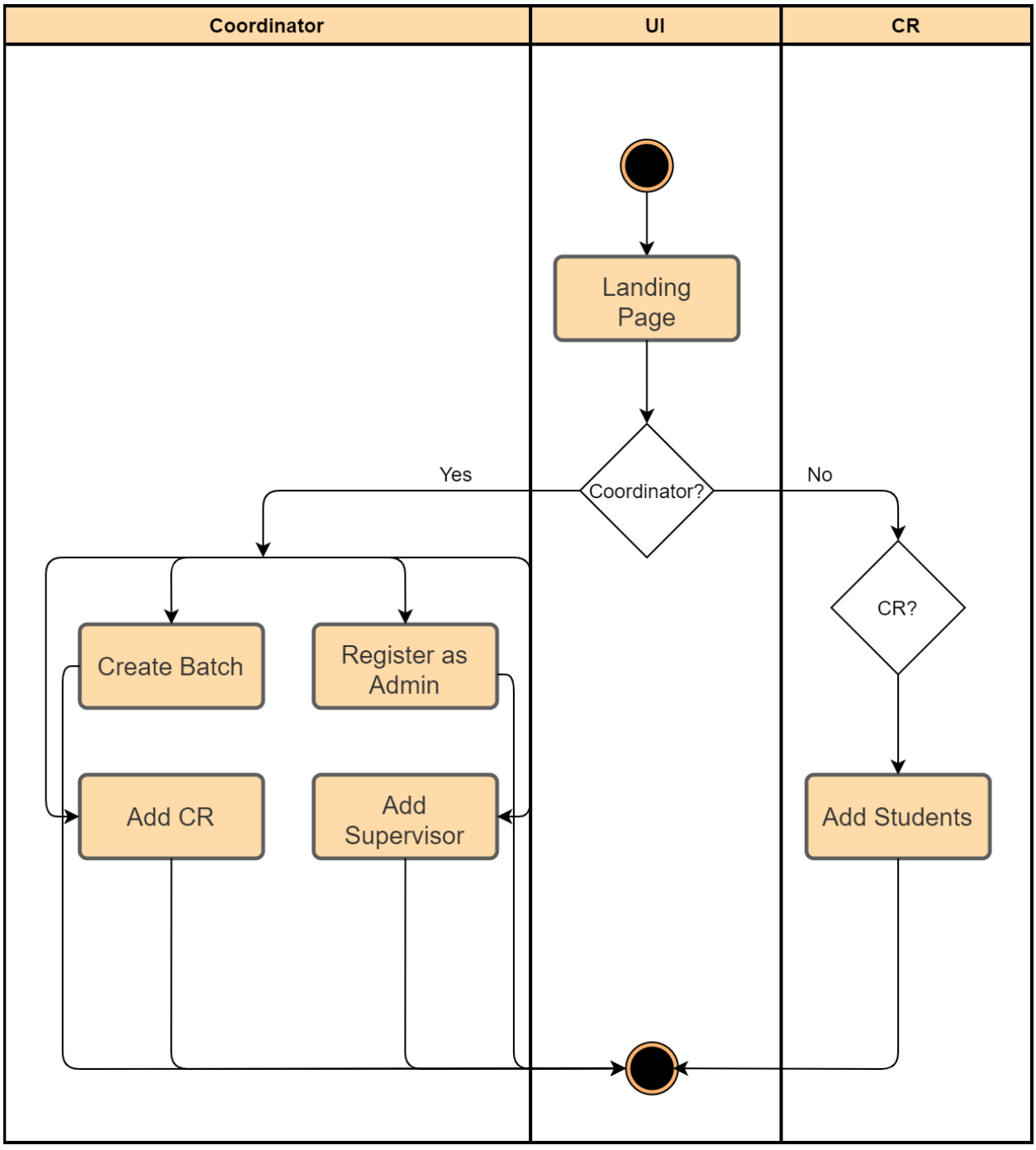


Figure 15: Batch Management (Swimlane - 1.1)

SID (Swimlane ID): 1.2
Name : Group Management
Reference: Use case & Activity diagram level-1.2

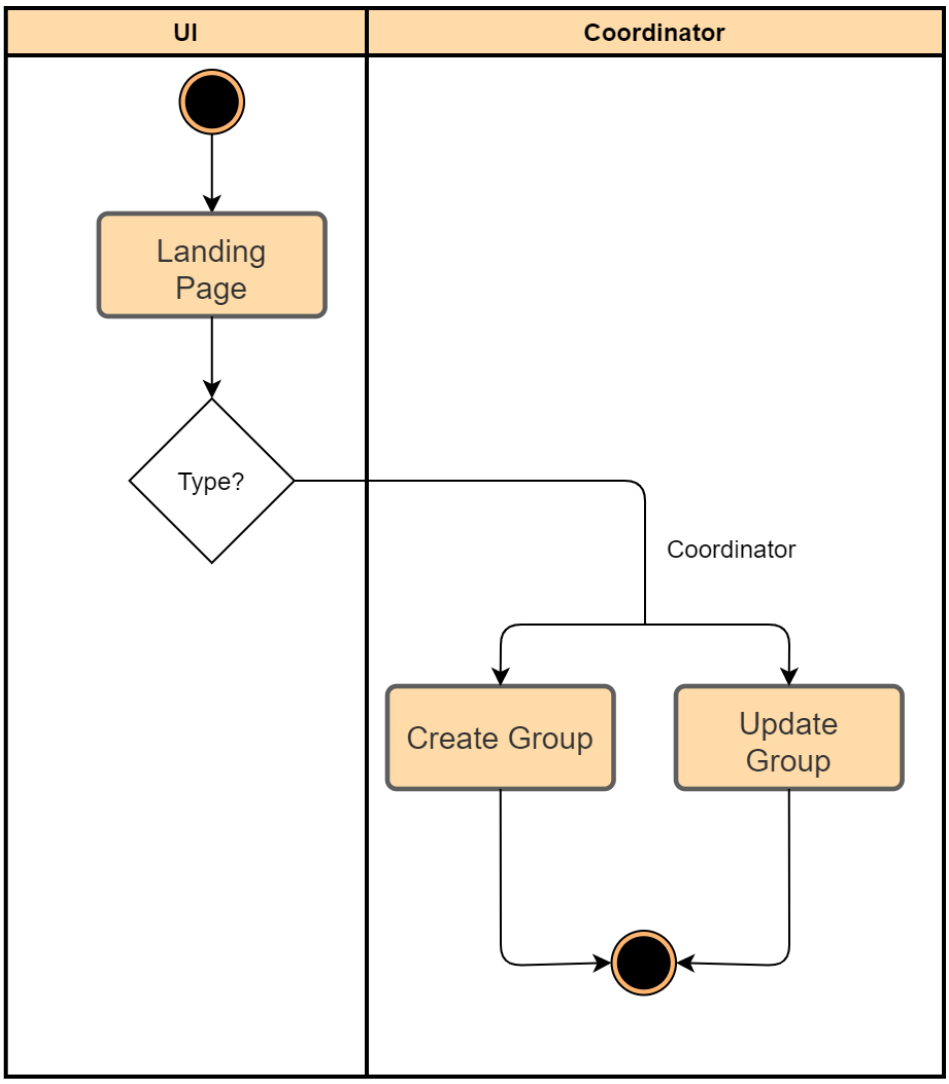


Figure 16: Group Management (Swimlane - 1.2)

SID (Swimlane ID): 1.3

Name : Activity

Reference: Use case & Activity diagram level-1.3

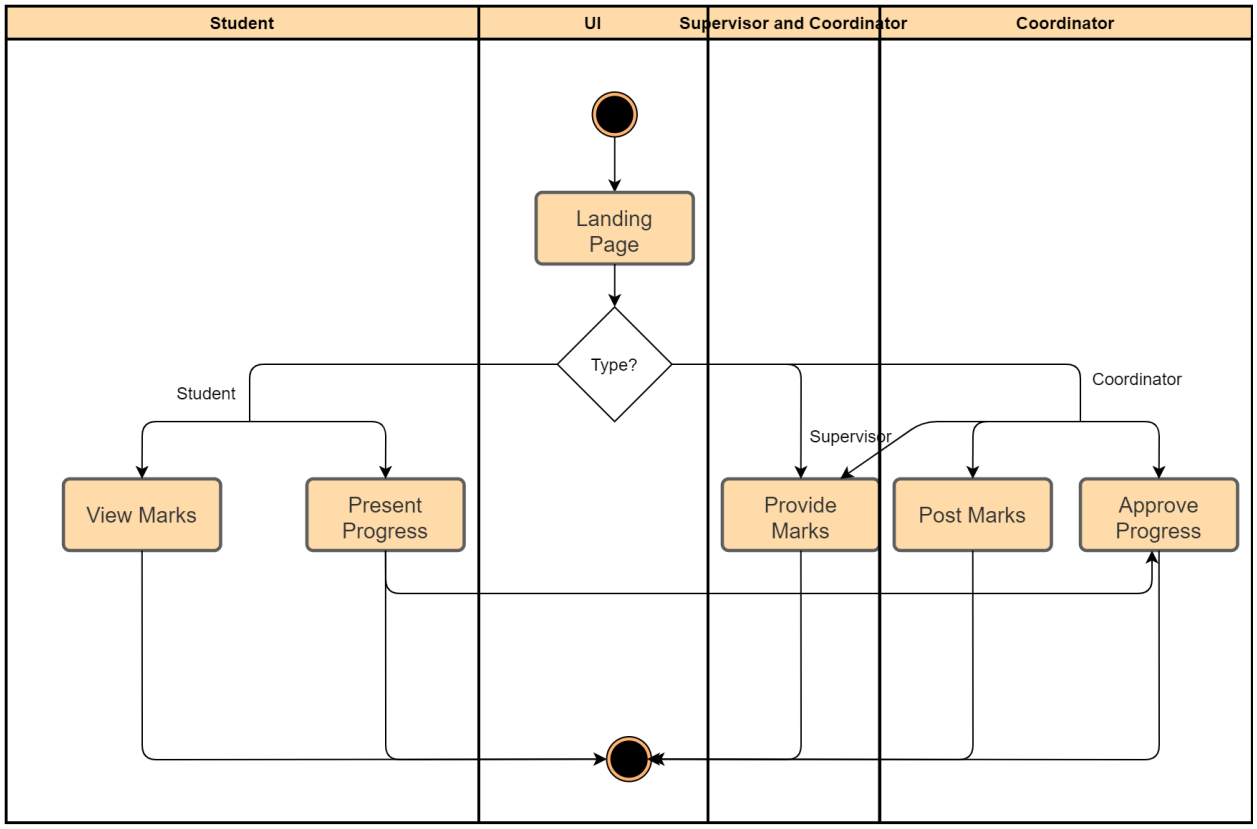


Figure 17: Activity (Swimlane - 1.3)

SID (Swimlane ID): 1.3.1

Name : Approve Progress

Reference: Use case & Activity diagram level-1.3.1

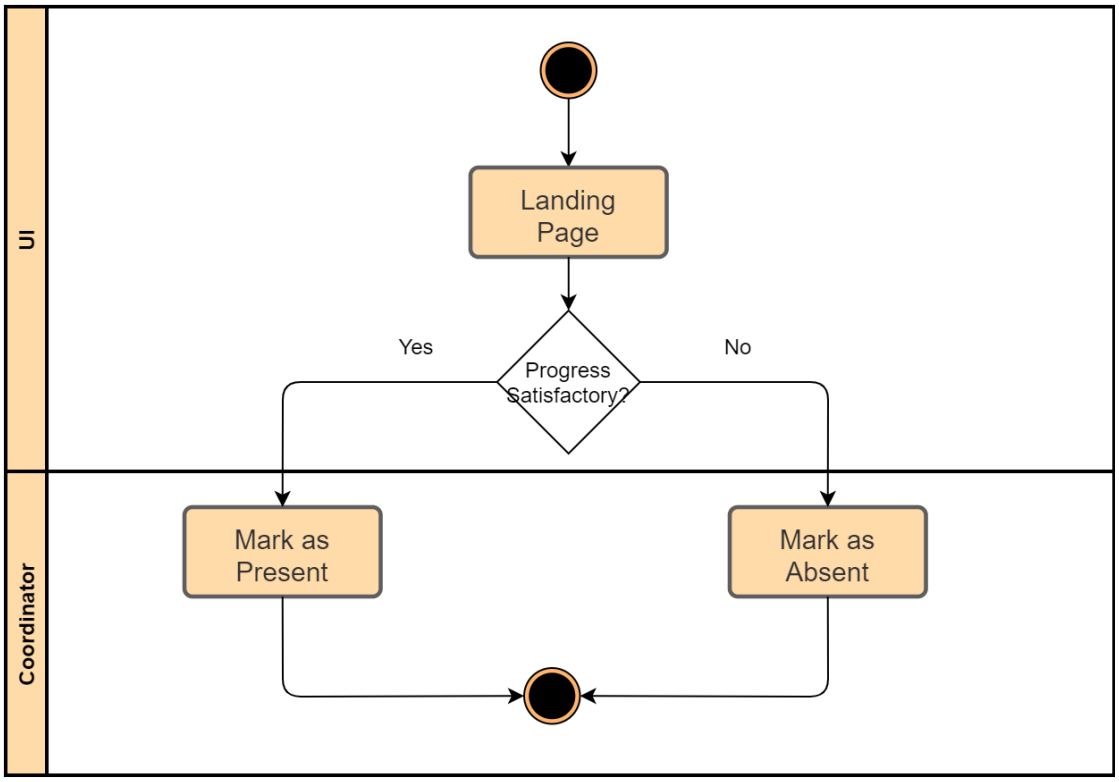


Figure 18: Approve Progress (Swimlane - 1.3.1)

SID (Swimlane ID): 1.4
Name : Post
Reference: Use case & Activity diagram level-1.4

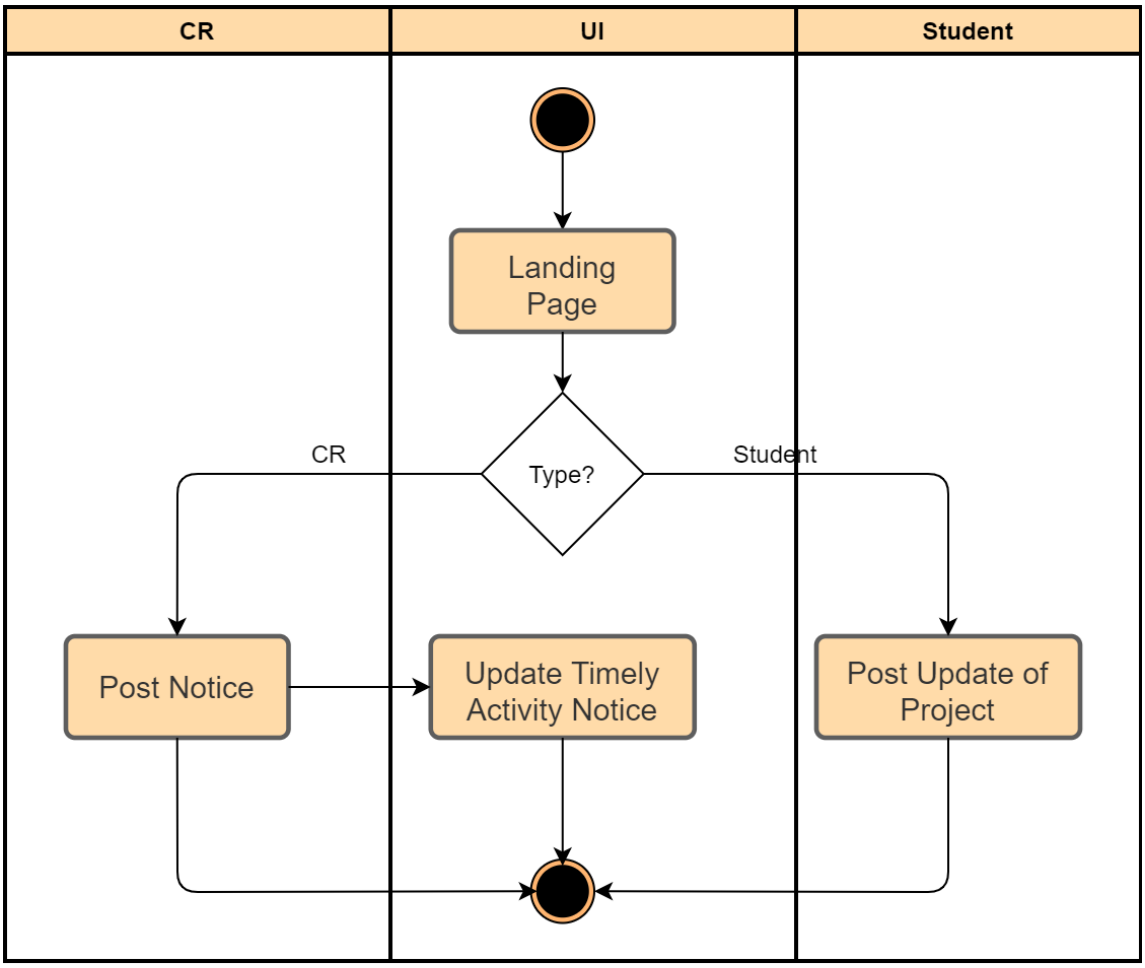


Figure 19: Post (Swimlane - 1.4)

Data Based Modelling

DATA MODELING 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 modeling. 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 Page 60 of 126 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 :

Serial	Noun	Problem/Solution Space	Attributes
1	**Student**	s	5,6,26
2	Member	p	
3	**Coordinator**	s	6,7
4	**CR**	s	5,6,26
5	Roll Number	s	
6	Name	s	
7	ID	s	
8	**Teacher**	s	6,7
9	**Group**	s	1,6,10,11,30
10	Project Title	s	
11	**Supervisor**	s	6,7
12	Progress	p	
13	Attendance	s	

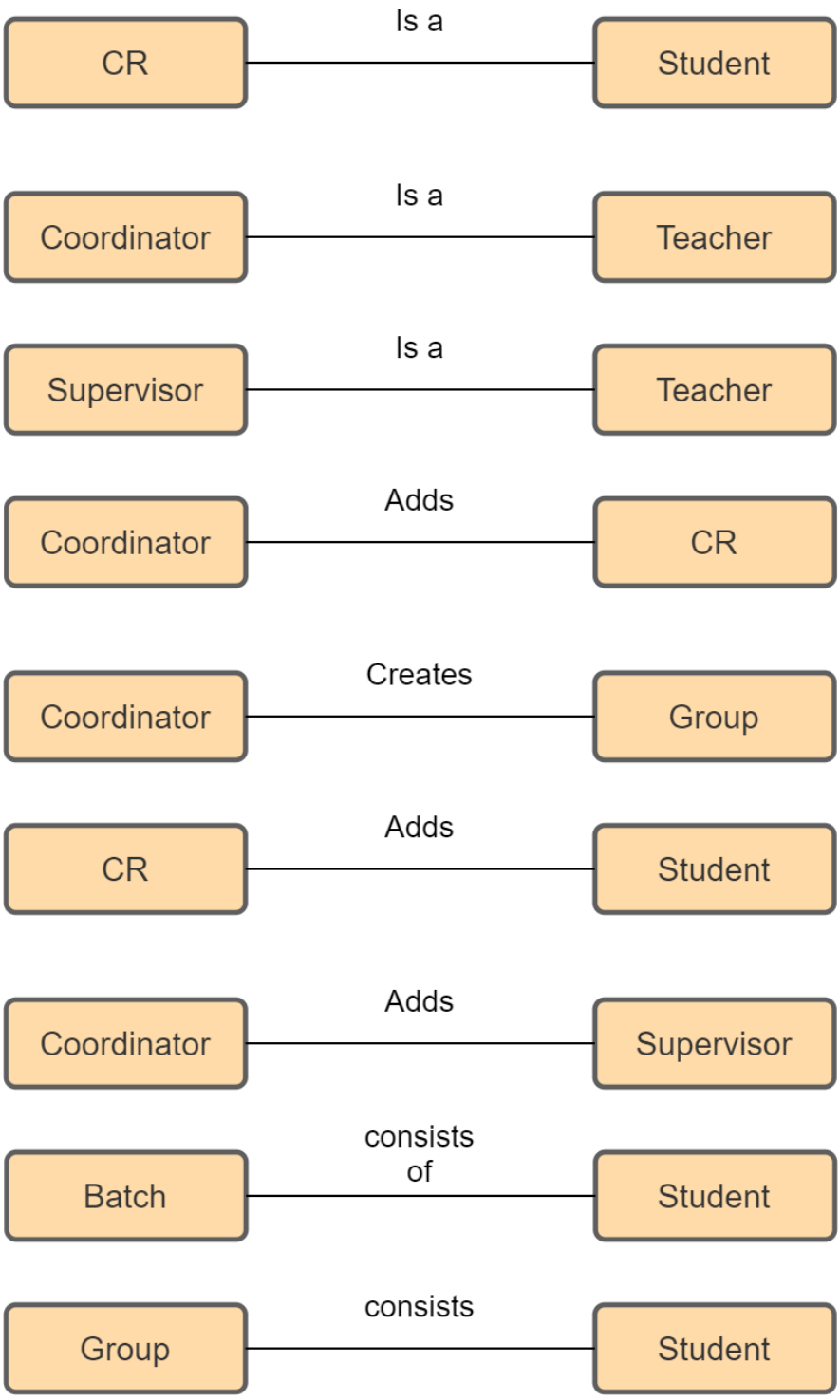
14	Proposal Presentation	s	
15	Mid Presentation	s	
16	Final Presentation	s	
17	Code Review	s	
18	Project Showcasing	s	
19	Final Report	s	
20	**Notice**	s	5,21,31
21	Facebook	p	
22	Github	p	
23	Activity Corner	p	
24	**Marks**	s	5,13,14,15,16,17,18,19
25	Admin(already taken)	s	6,7
26	Github link	s	
27	**Batch**	s	1,10,7,31
28	Deadline (timely activity)	s	
29	**Update**	s	5,22
30	Group ID	s	
31	Batch ID	s	

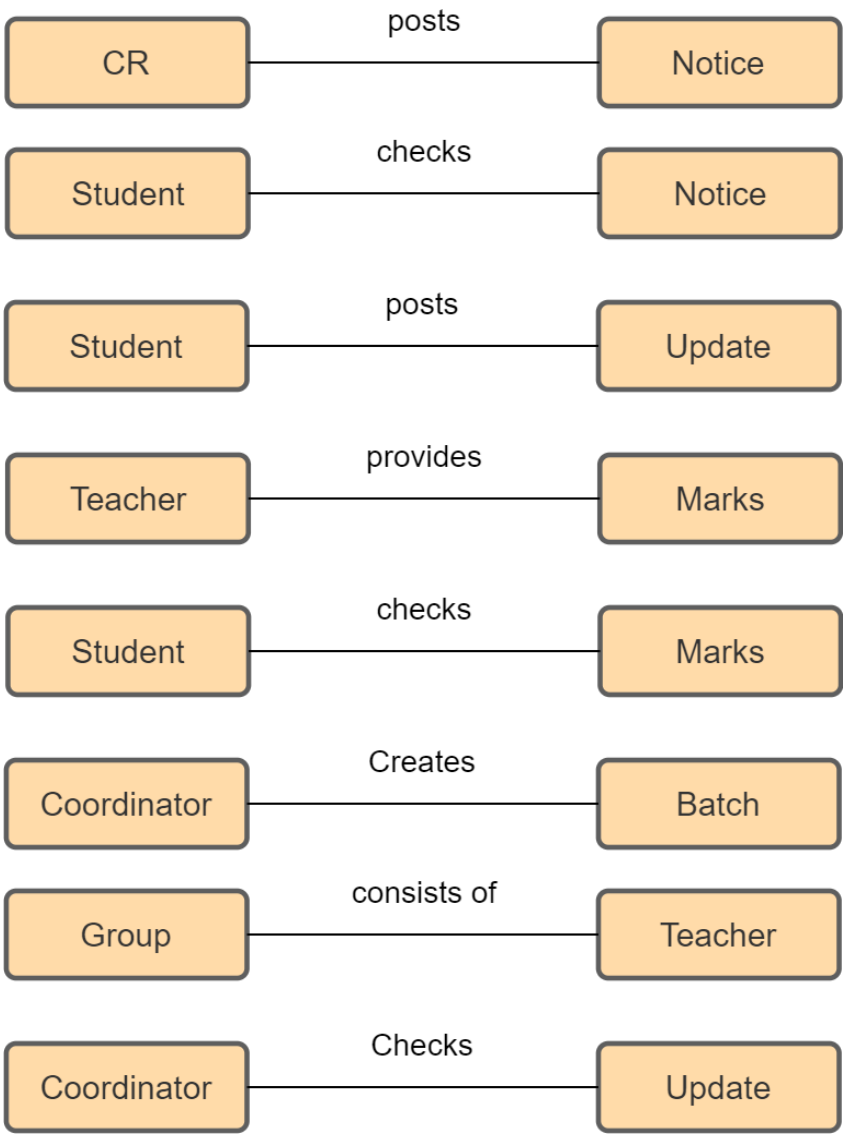
Final Data Object:

1. Student
2. Coordinator
3. CR
4. Teacher
5. Group
6. Notice

7. Marks
8. Update
9. Supervisor
10. Batch

Relationship Between Data Objects:


CR
Is a
Student
Coordinator
Is a
Teacher
Supervisor
Is a
Teacher
Coordinator
Adds
CR
Coordinator
Creates
Group
CR
Adds
Student
Coordinator
Adds
Supervisor
Batch
consists of
Student
Group
consists
Student



ER Diagram:

Definition of ER Diagram : An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.

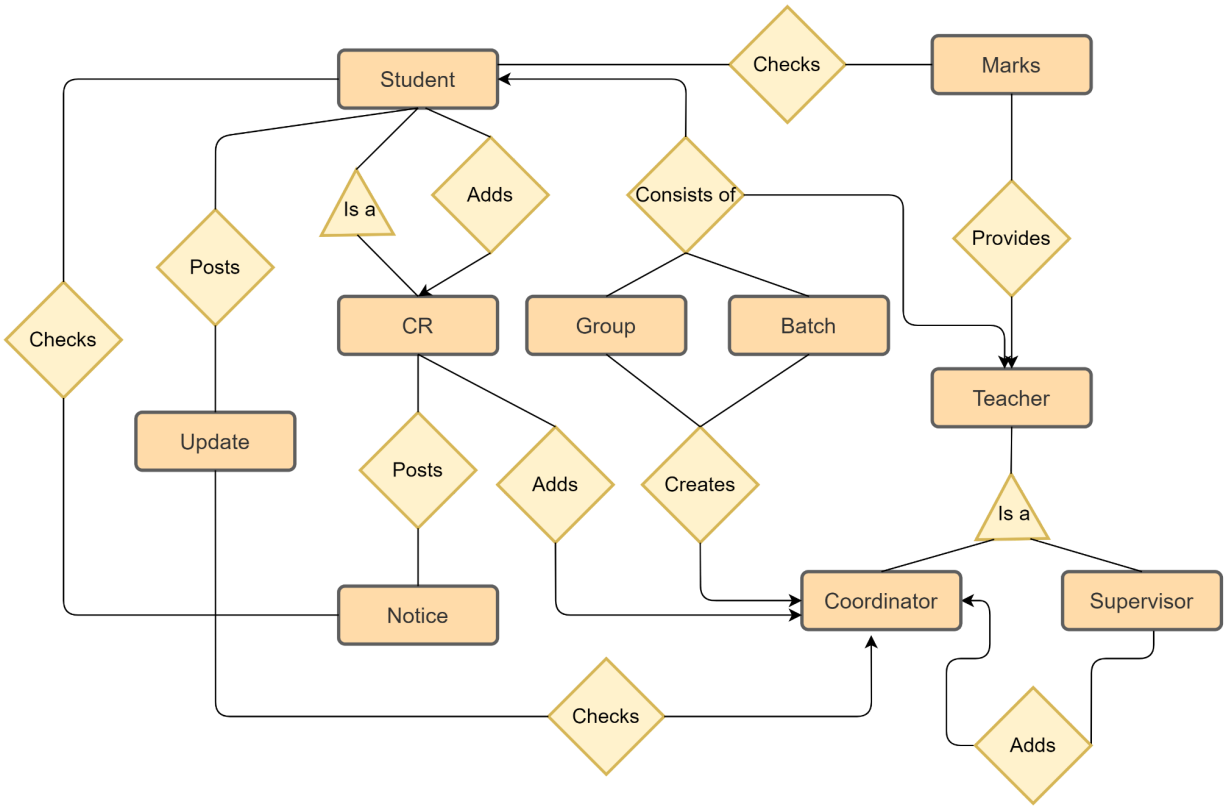


Figure 20: ER Diagram

Schema Diagram:

Data Object	Attribute	Type	Size	Comment
Teacher	Name ID is_Supervisor is_Coordinator	Varchar Number Bit Bit	40 6 1 1	 Omitting object supervisor with this attribute Omitting object coordinator with this attribute
Group	Name Project_Title Group_ID Teacher_ID	Varchar Varchar Number Number	40 40 6 6	 supervisor's ID

Marks	Roll_Number Attendance Proposal_Presentation Mid_Presentation Final_Presentation Code_Review Project_Showcasing Final_Report	Number Number Number Number Number Number Number Number	6 2 2 2 2 2 2 2	
Batch	Teacher_ID Project_Title Batch_ID	Number Varchar Number	6 40 6	Supervisor's ID
Update	Roll Number Github_Post_ID Batch_ID	Number Number Number	6 6 6	
Student	Roll_Number Name Github Link is_CR Group_ID Batch_ID	Number Varchar Number Bit Number Number	6 40 6 1 6 6	 (converting one to many relationship)
Notice	Roll Number Facebook_Post_ID Batch_ID	Number Number Number	6 6 6	Who posted the announcement ID to the post, for ease to show at UI To know which batch should the UI show
checks	-teacher_ID -Roll_number	number number	6 6	
posts	-github_post_id -Roll_Number	Number number	6 6	

Class Based Modeling:

CLASS BASED MODELING CONCEPT : Class-based modeling 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.

Identified Noun:

Serial	Noun
1	Student
2	Member
3	Coordinator
4	CR
5	Roll Number
6	Name
7	Teacher ID
8	Teacher
9	Group
10	Project Title
11	Supervisor
12	Progress

13	Attendance
14	Proposal Presentation
15	Mid Presentation
16	Final Presentation
17	Code Review
18	Project Showcasing
19	Final Report
20	Notice
21	Facebook
22	Github
23	Activity Corner
24	Marks
25	Admin(already taken)
26	Github link
27	Batch
28	Deadline (timely activity)
29	Update
30	Group ID
31	Batch ID
32	Post

Identified Verb:

Serial	Verb

1	Add
2	Register
3	Have
4	Create
5	Have
6	Contain
7	Present Weekly Progress
8	Get Attendance
9	Post
10	Give
11	See
12	Keep
13	Add

General Classification:

Candidate classes are categorized based on the seven general classification. The analysis classes manifest themselves in one of the following ways:

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

A candidate class is selected for special classification if it fulfills three or more characteristics.

Serial	Solution Space Nouns	General Classification
1	Student	4,5,7
2	Coordinator	4,5,7
3	CR	4,5,7
4	Roll Number	2
5	Name	2
6	Teacher ID	2
7	Teacher	4,5,7
8	Group	4,5,7
9	Project Title	2
10	Supervisor	4,5,7
11	Attendance	2,3
12	Proposal Presentation	2,3
13	Mid Presentation	2,3
14	Final Presentation	2,3
15	Code Review	2,3
16	Project Showcasing	2,3
17	Final Report	2,3
18	Notice	2,3
19	Marks	2,3,7
20	Admin(already taken)	4,5,7
21	Github link	2
22	Batch	4,5,7
23	Deadline (timely activity)	2

24	Update	2,3
25	Group ID	2
26	Batch ID	2
27	Facebook	1
28	Github	1
29	Post	2,3,6

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 :

Serial	Solution Space Nouns	Selection Criteria
1	Student	1-5 (Selected)
2	Coordinator	1-5 (Selected)
3	CR	1-5 (Selected)
4	Teacher	1-5 (Selected)
5	Group	1-5 (Selected)
6	Supervisor	1-5 (Selected)

7	Marks	1-5 (Selected)
8	Admin(already taken)	1-5
9	Batch	1-5 (Selected)
10	Post	1,2,6 (Selected)
11	UI	1,2,3(Selected)

Attribute and method identification:

Class	Attribute	Methods
Student	-Name -Roll_Number -Marks	+getName() +getRollNumber() +getMarks() +setName() +setRollNumber() +setMarks()
Coordinator	-Name -ID	+register() +getName() +getID() +setName() +setID() +addCR() +giveMarks() +approveProgress()
CR	-Name -Roll_Number -Marks	+getName() +getRollNumber() +getMarks() +setName() +setRollNumber() +setMarks() +addStudentsToBatch()
Teacher	-Name -ID	+getName() +getID() +setName() +setID() +giveMarks()

Group	-Name -ID -Project_Title -Supervisor -Student[]	+getSupervisor() +getGroupName() +getProjectTitle() +setSupervisor() +setGroupName() +setProjectTitle() -saveCodeAndReports() +getStudentList()
Supervisor	-Name -ID	+getName() +getID() +setName() +setID() +giveMarks()
Marks	-Attendance_Marks -Proposal_PresentationMarks -Mid_Presentation_Marks -Final_Presentation_Marks -Code_Review_Marks -Project_Showcasing_Marks -Final_Report_Marks	+setTotalMarks +getTotalMarks() -countMarks() -postMarks +getAttendanceMarks +getProjectProposalPresentationMarks +getMidPresentationMarks +getFinalPresentationMarks +getCodeReviewMarks +getProjectShowcasingMarks +getFinalReportMarks +setAttendanceMarks +setProjectProposalPresentationMarks +setMidPresentationMarks +setFinalPresentationMarks +setCodeReviewMarks +setProjectShowcasingMarks +setFinalReportMarks
Batch	-Name -ID -Coordinator -Student[]	+getName() +getID() +setName() +setID() +getCoordinator() +setCoordinator() +getStudentList() +updateStudentList()
Post	-type -Post_ID	+showPost() +getType() +getPostID()

		+setType()
UI		+showCurrentStatus() +getInput() +showNotices()

Analysis

All classes included in the class based diagram are selected as classes for our system.

CRC card

Class	Responsibility	Collaborator
Student	• View Marks • Present Progress • Post Progress on Github • Be in Group	Post, CR, Batch, Group, Marks
Coordinator	• Add CR • Approve Progress • Post Marks • Create Group • Give Marks	CR, Student, Batch, Group, Marks
CR	• Adding Students • Posting Notice	Student, Post
Teacher	• Give Marks	Marks, Student,Coordinator, Supervisor
Group	• Presenting Progress • Saving Reports	Coordinator, Student, Supervisor
Supervisor	• Give Marks • Supervising Group	Group, Marks, Student
Marks	• Save the Marks • Update the Marks • Show the Marks to Student	Teacher, Student
Batch	• View Batch Status • Add student to batch • Update batch info	CR, Student, Supervisor, Coordinator
Post	• Post Notice on Facebook	CR, Students

	• Contain the Notices on Sidebar • Post Progress on Github	
UI	• Show Notices • Get inputs • Show Marks	Student, Teacher, Marks, Group, Batch, Post

Class Card:

Student	
Attribute	Method
-Name -Roll_Number -Marks	+getName() +getRollNumber() +getMarks() +setName() +setRollNumber() +setMarks()
Responsibilities	Collaborator
• View Marks • Present Progress • Post Progress on Github • Be in Group	Post, CR, Batch, Group, Marks

Teacher	
Attribute	Method
-Name -ID	+getName() +getID() +setName() +setID() +giveMarks()
Responsibilities	Collaborator
• Give Marks	Marks, Student, Coordinator, Supervisor

CR	
Attribute	Method
-Name -Roll_Number -Marks	+getName() +getRollNumber() +getMarks() +setName() +setRollNumber() +setMarks() +addStudentsToBatch()
Responsibilities	Collaborator
• Adding Students • Posting Notice	Post, CR

Supervisor	
Attribute	Method
-Name -ID	+getName() +getID() +setName() +setID() +giveMarks()
Responsibilities	Collaborator
• Give Marks • Supervising Group	Group, Marks, Student

Coordinator	
Attribute	Method
-Name -ID	+register() +getName() +getID() +setName() +setID() +addCR() +giveMarks() +approveProgress()

Responsibilities	Collaborator
• Add CR • Approve Progress • Post Marks • Create Group • Give Marks	CR, Student, Batch, Group, Marks

UI	
Attribute	Method
	+showCurrentStatus() +getInput() +showNotices()
Responsibilities	Collaborator
• Show Notices • Get inputs • Show Marks	Student, Teacher, Marks, Group, Batch, Post

Post	
Attribute	Method

-type -Post_ID	+showPost() +getType() +getPostID() +setType()
Responsibilities	Collaborator
• Post Notice on Facebook • Contain the Notices on Sidebar • Post Progress on Github	CR, Students

Batch	
Attribute	Method
-Name -ID -Coordinator -Student[]	+getName() +getID() +setName() +setID() +getCoordinator() +setCoordinator() +getStudentList() +updateStudentList()
Responsibilities	Collaborator
• View Batch Status • Add student to batch • Update batch info	CR, Student, Supervisor, Coordinator

Group

Attribute	Method
-Name -ID -Project_Title -Supervisor -Student[]	+getSupervisor() +getGroupName() +getProjectTitle() +setSupervisor() +setGroupName() +setProjectTitle() -saveCodeAndReports() +getStudentList()
Responsibilities	Collaborator
• Presenting Progress • Saving Reports	Coordinator, Student, Supervisor

Marks	
Attribute	Method
-Attendance_Marks -Proposal_PresentationMarks -Mid_Presentation_Marks -Final_Presentation_Marks -Code_Review_Marks -Project_Showcasing_Marks -Final_Report_Marks	+setTotalMarks +getTotalMarks() -countMarks() -postMarks +getAttendanceMarks +getProjectProposalPresentationMarks +getMidPresentationMarks +getFinalPresentationMarks +getCodeReviewMarks +getProjectShowcasingMarks +getFinalReportMarks +setAttendanceMarks +setProjectProposalPresentationMarks +setMidPresentationMarks +setFinalPresentationMarks

	+setCodeReviewMarks +setProjectShowcasingMarks +setFinalReportMarks
Responsibilities	Collaborator
• Save the Marks • Update the Marks • Show the Marks to Student	Teacher, Student

CRC Diagram

Diagram Id: 1

Name: Student

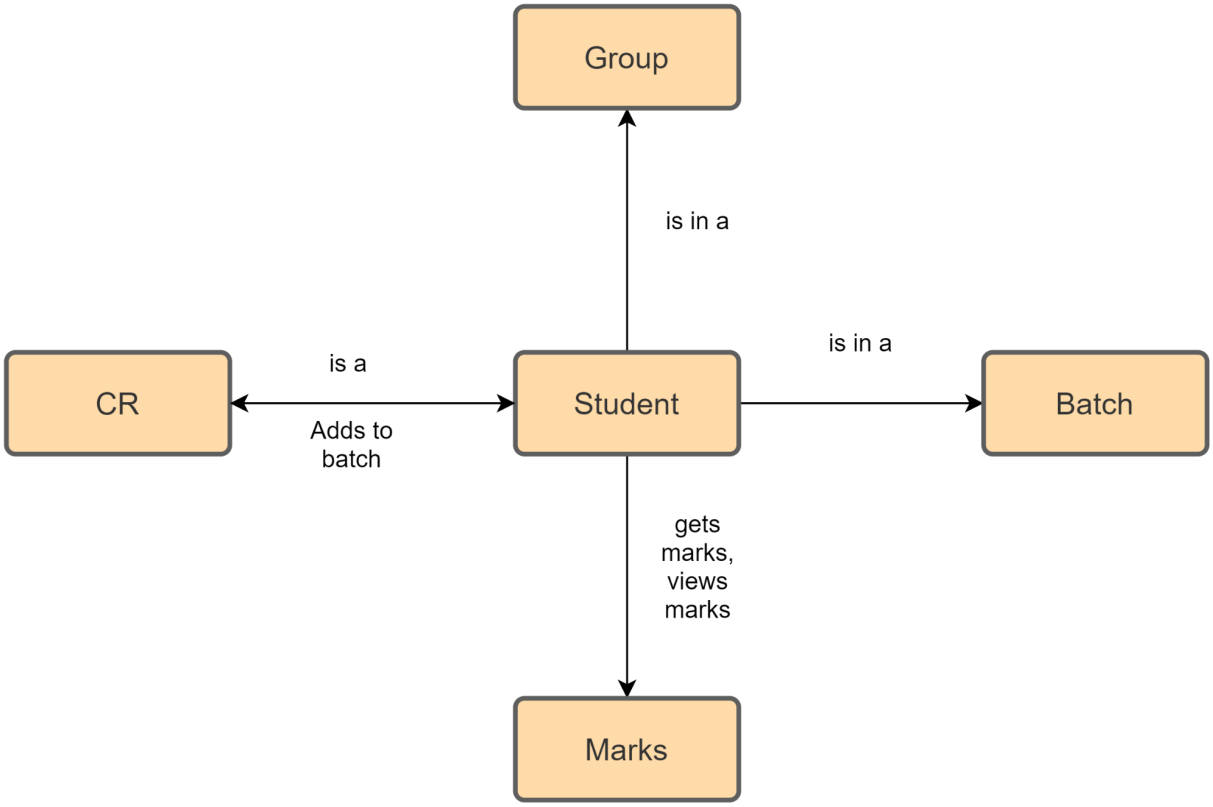


Figure 21: CRC (Student)

Diagram Id: 2
Name:CR

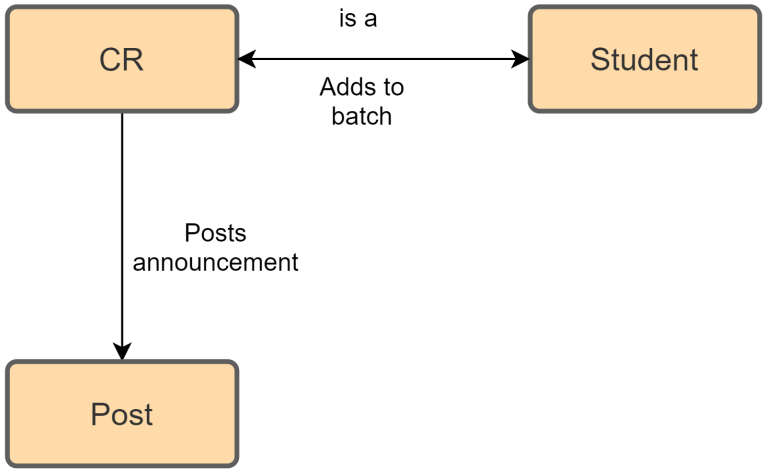


Figure 22: CRC (CR)

Diagram Id: 3
Name:Teacher

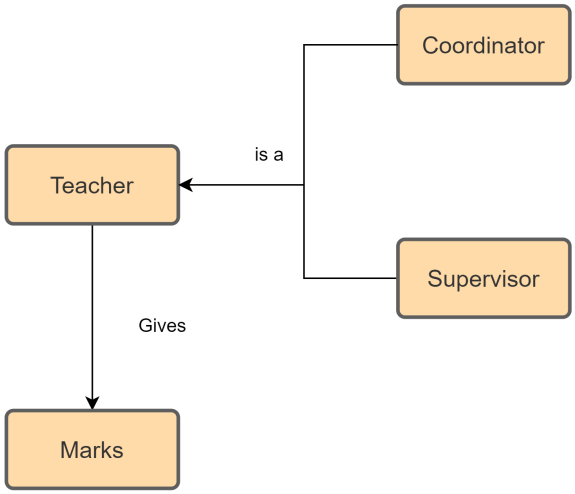


Figure 23: CRC (Teacher)

Diagram Id: 4
Name: Coordinator

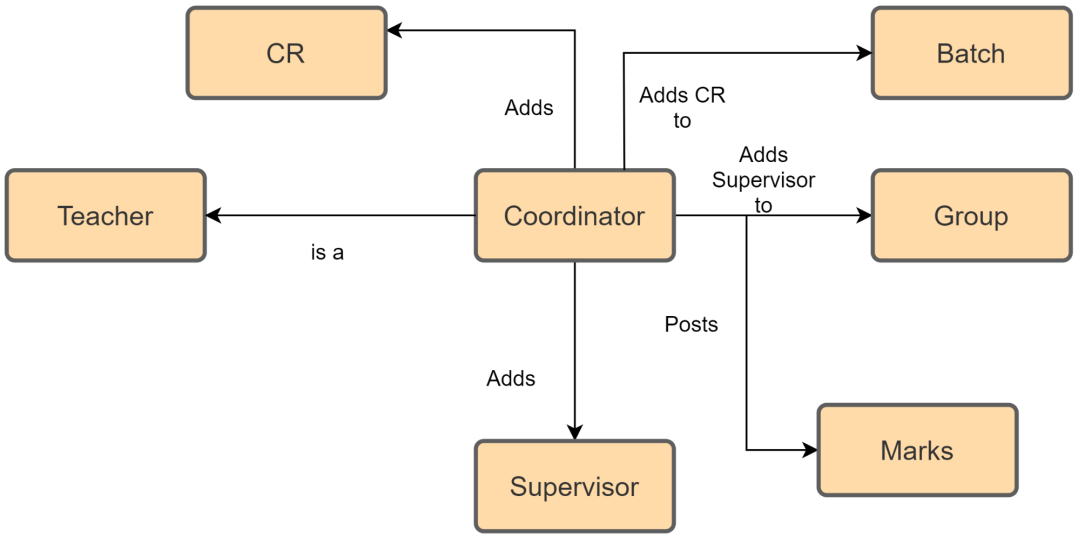


Figure 24: CRC (Coordinator)

Diagram Id: 5
Name: Supervisor

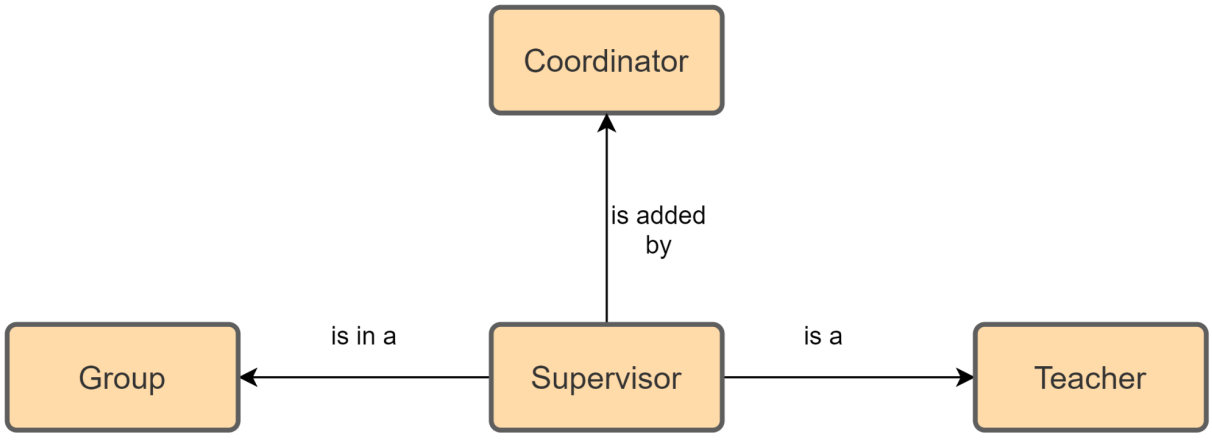


Figure 25: CRC (Supervisor)

Diagram Id: 6

Name: Group

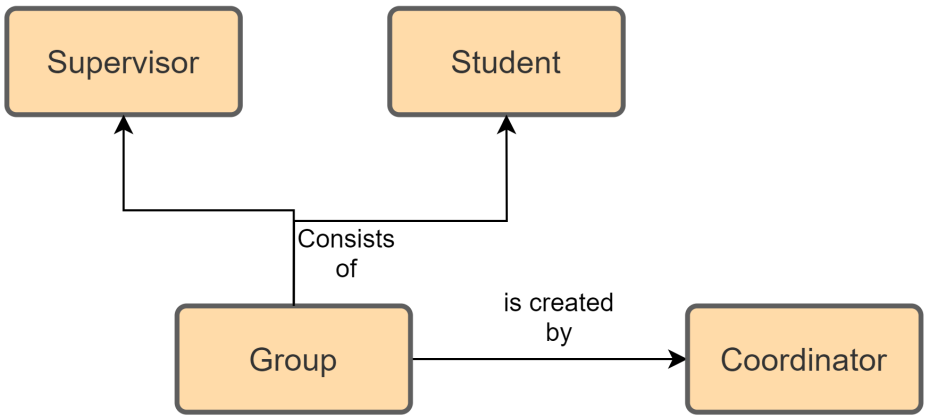


Figure 26: CRC (Group)

Diagram Id: 7
Name: Batch

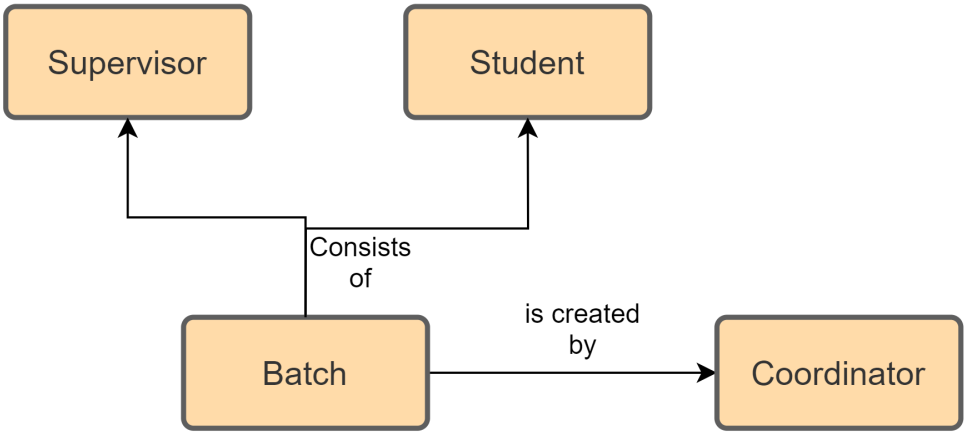


Figure 27: CRC (Batch)

Diagram Id: 8

Name: Marks

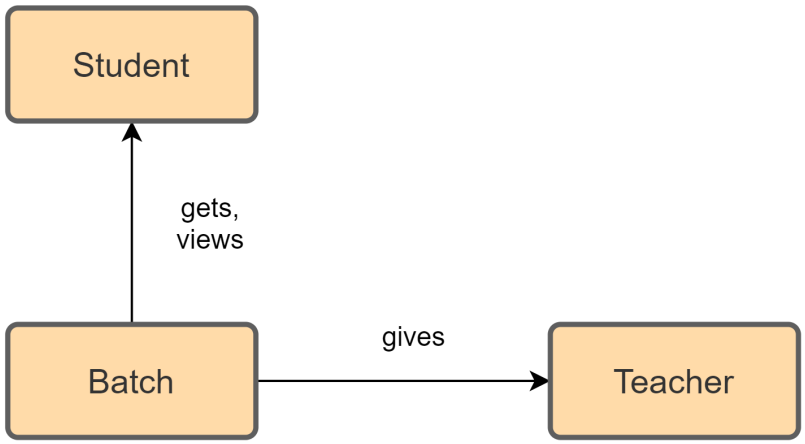


Figure 28: CRC (Marks)

Diagram Id: 9
Name: Post

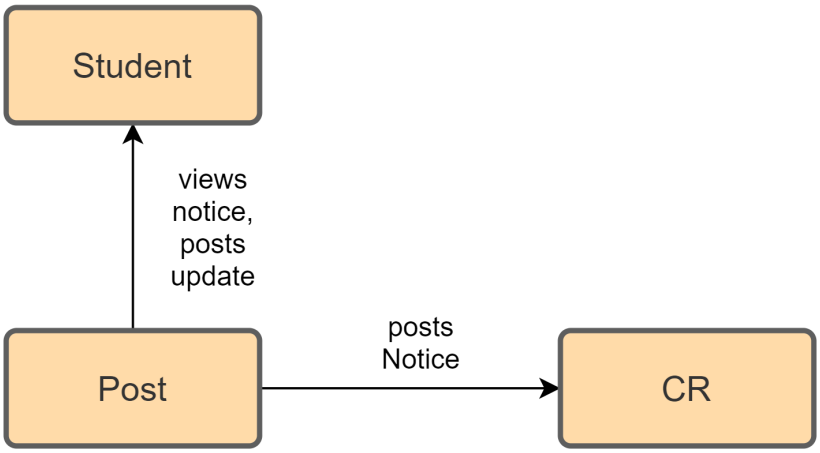


Figure 29: CRC (Post)

Diagram Id: 10

Name: Group

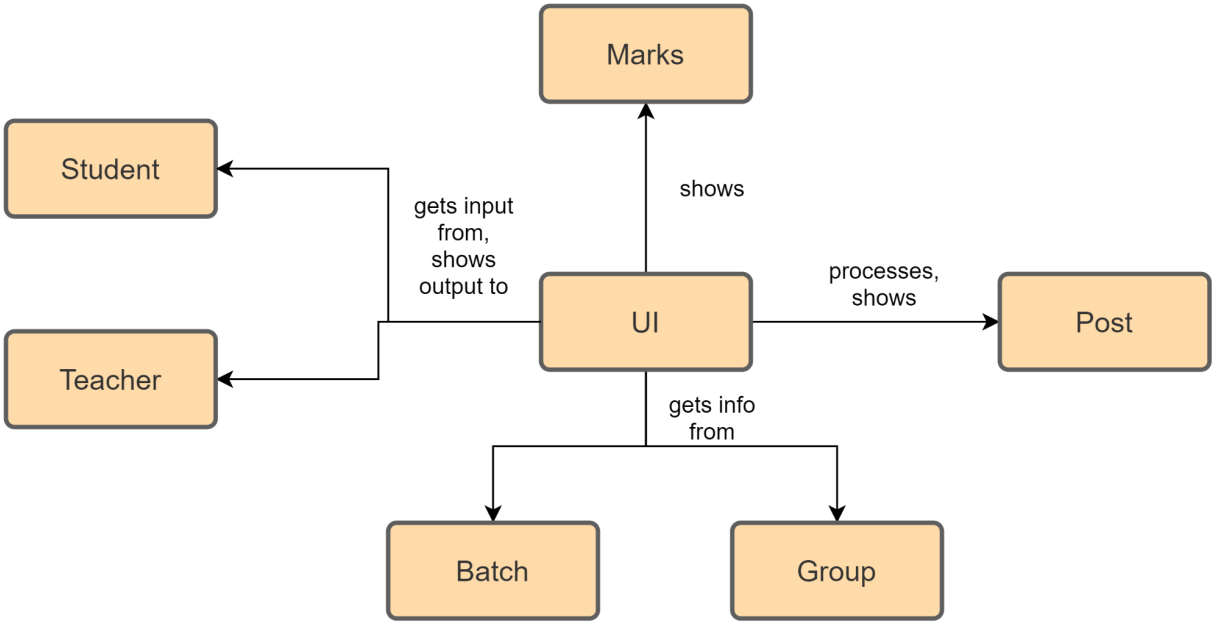


Figure 30: CRC (UI)

BEHAVIORAL MODELING OF SPMA

STATE TRANSITION DIAGRAM :

State diagram represents active states for each class of events (triggers). For this we identified all the events, their initiators and collaborators.

Event Table

SL No.	Event	Event Name	Initiator	Collaborator	Associated Method
1	Launch	launch_app	Teacher, Student	UI	+launch()

2	Register	Register_as_Admin	Coordinator	UI	+register()
3	Add Member	add_Student	CR	Batch, UI	+add_students()
4	Add CR	add_CR	Coordinator	UI, Batch	+add_CR()
5	Add Supervisor	add_Supervisor	Coordinator	UI, Batch	+add_supervisor()
6	Create Group	create_Group	Coordinator	UI, Group	+create_group()
7	Provide Credentials	provide_credentials	Student, Teacher	UI, Batch, Group, Marks, Post	+provide_credentials()
8	Present Progress	present_Progress	Student	UI, Coordinator	+present_progress()
9	Evaluate Progress	evaluate_progress	Coordinator	UI, Student, Marks	+approve_progress() +update_marks() +update_attendance()
10	Choose Marks Content	choose_option	Teacher	UI	+choose_option()
11	Give Marks	provide_marks	Teacher	Student, Marks, UI	+provide_marks()
12	Post Notice	post_notice	CR	Student, UI	+post_notice() +update_actiivty_corner() +post_on_facebook()
12	Post Progress	post_progress	Student	Student, UI, Teacher	+post_progress() +update_github()
13	View Marks	view_marks	Student	UI, Marks	+view_marks()
14	Post Marks	post_marks	Coordinator	UI, Marks, Students	+post_marks()

State Transition

ID: 1

Name:Student

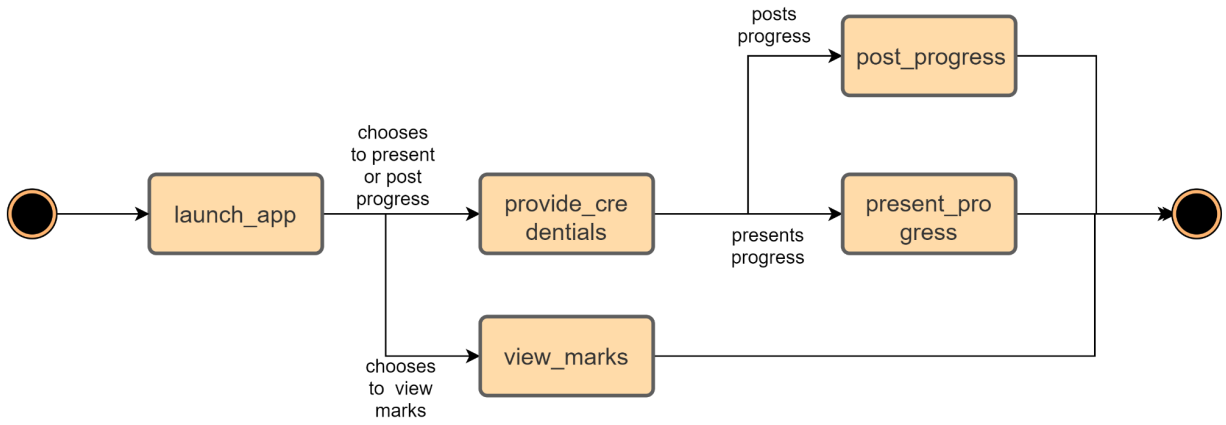


Figure 31: State Transition Diagram (Student)

ID: 2

Name:CR

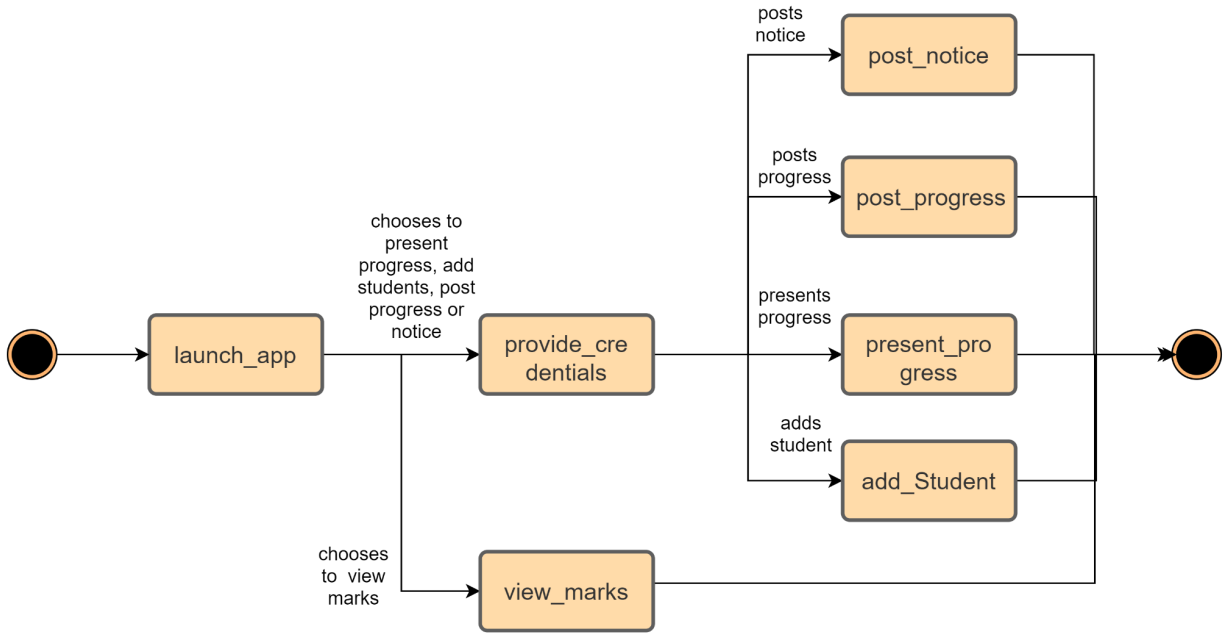


Figure 32: State Transition Diagram (CR)

ID: 3

Name: Teacher

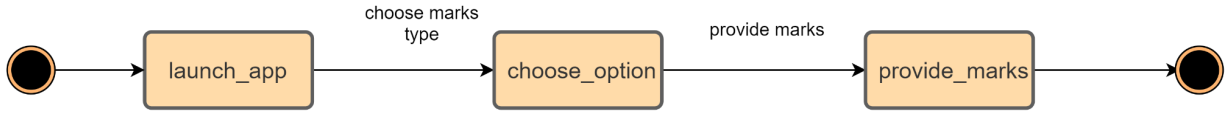


Figure 33: State Transition Diagram (Teacher)

ID: 4

Name: Supervisor

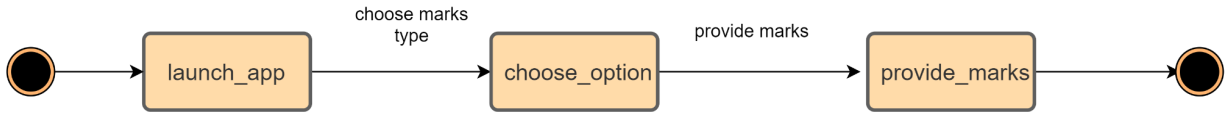


Figure 34: State Transition Diagram (Supervisor)

ID:5

Name: Coordinator

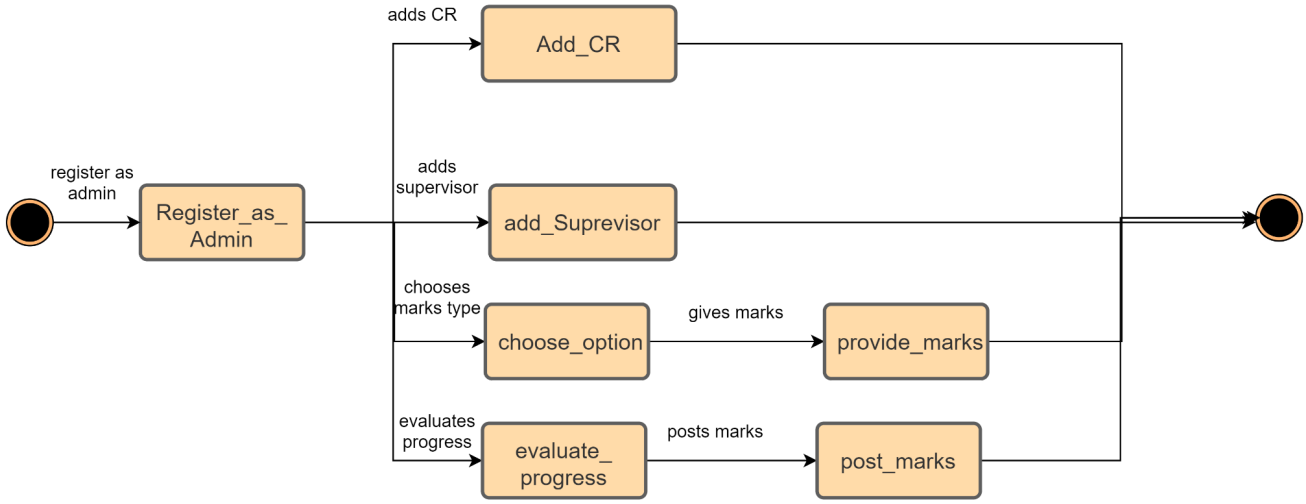
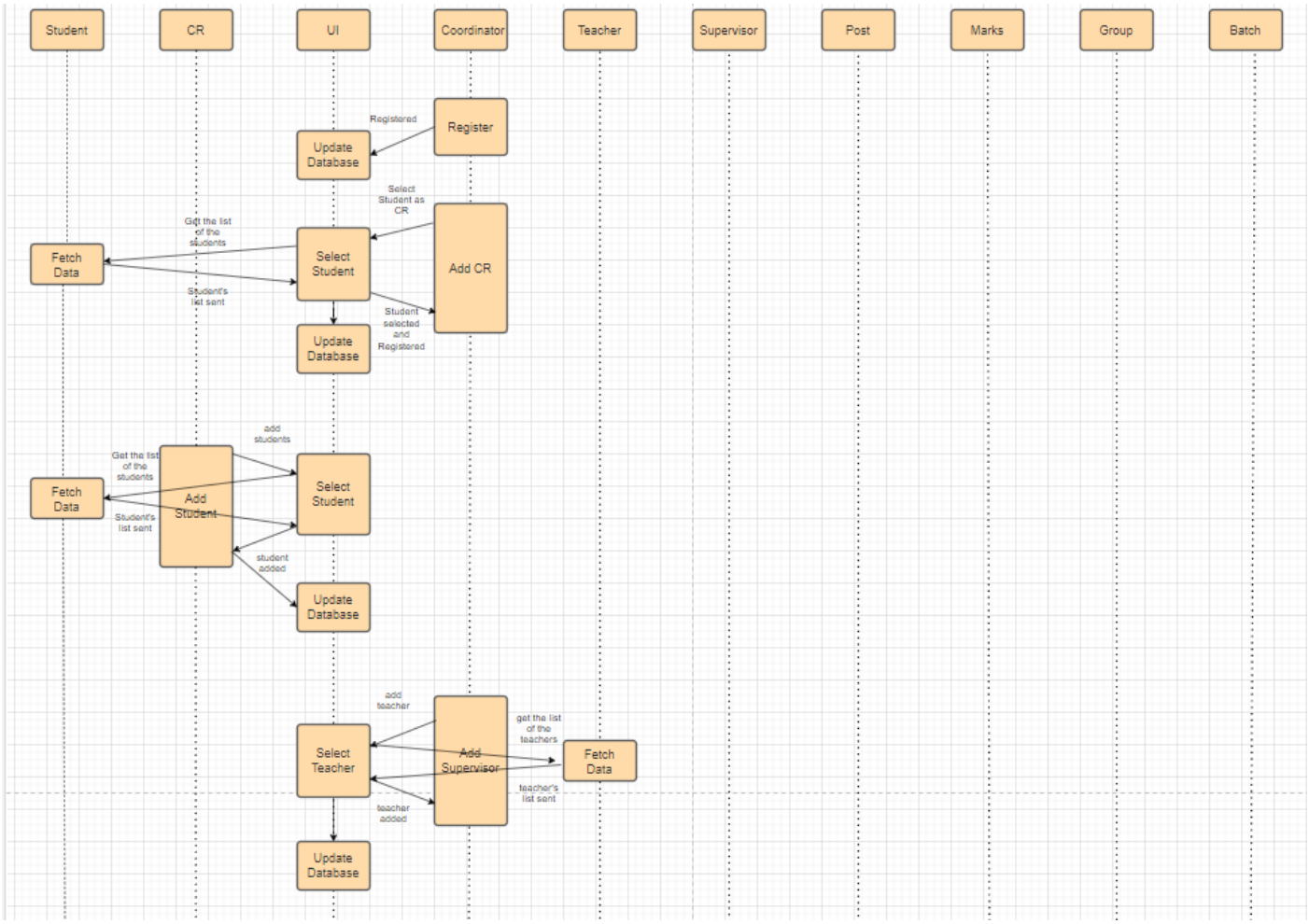
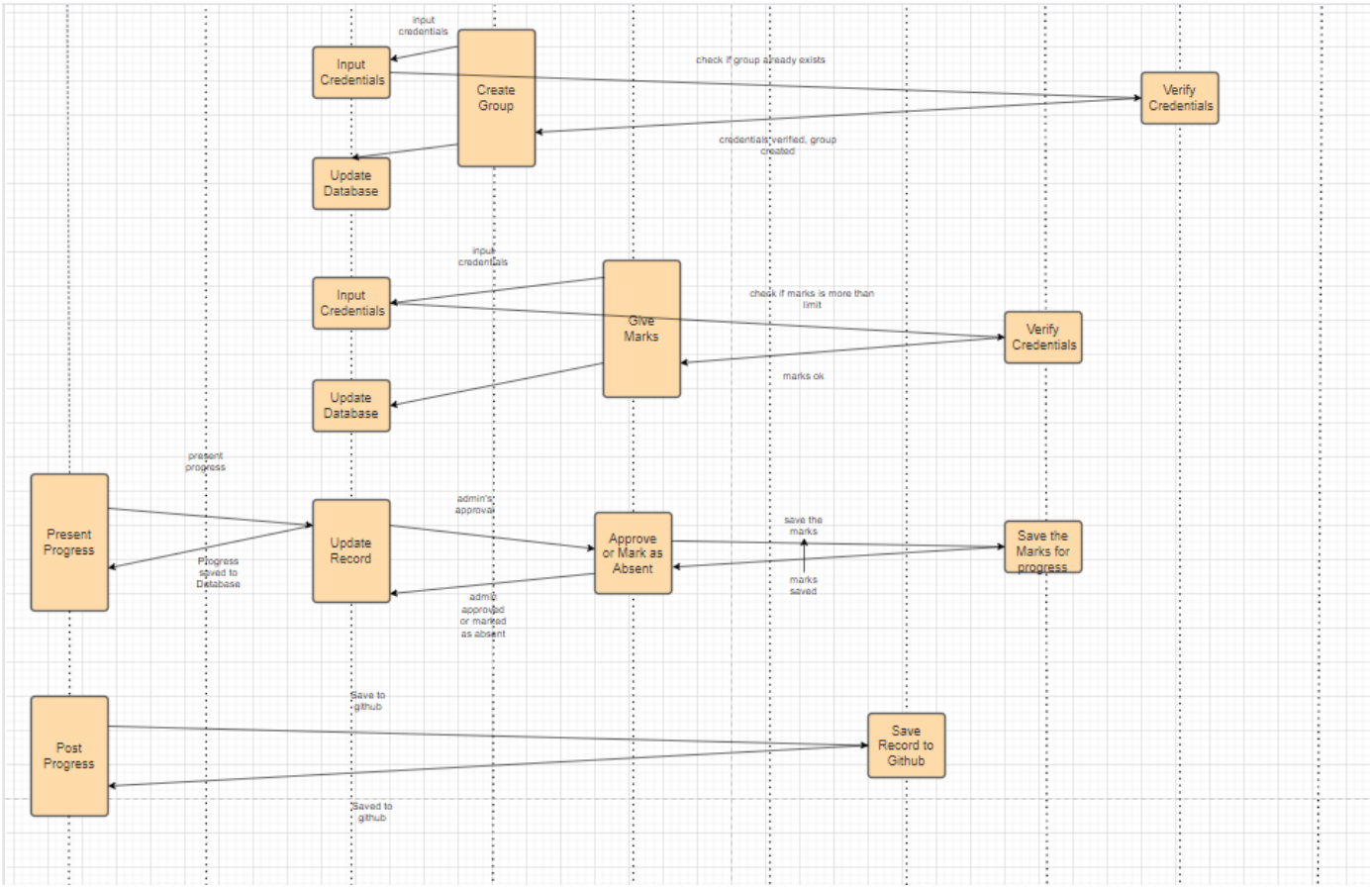


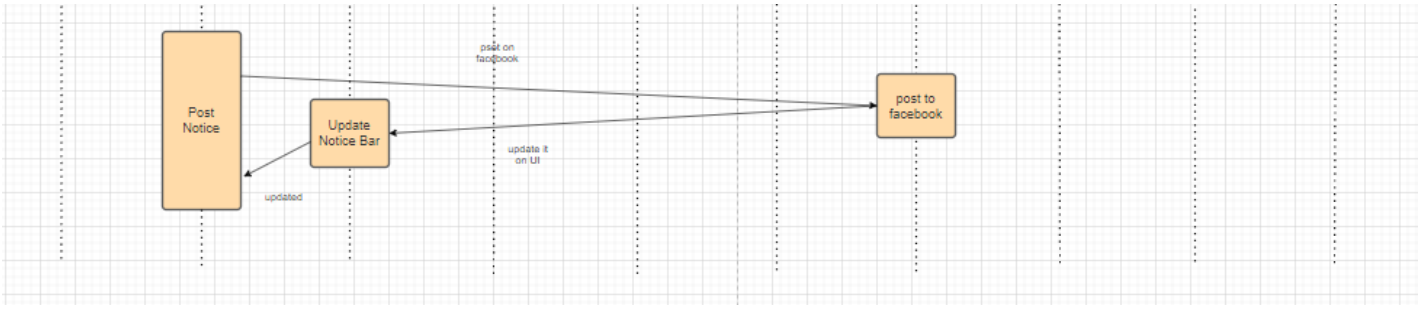
Figure 35: State Transition Diagram (Coordinator)

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
CR
UI
Coordinator
Teacher
Supervisor
Post
Marks
Group
Batch
Register
Registered
Update Database
Select Student as CR
Add CR
Get the list of the students
Fetch Data
Select Student
Student's list sent
Student selected and Registered
Update Database
add students
Get the list of the students
Fetch Data
Add Student
Student's list sent
Select Student
student added
Update Database
add teacher
get the list of the teachers
Select Teacher
Add Supervisor
Fetch Data
teacher's list sent
teacher added
Update Database


input credentials
Input Credentials
Create Group
check if group already exists
Verify Credentials
credentials verified, group created
Update Database
input credentials
Input Credentials
Give Marks
check if marks is more than limit
Verify Credentials
marks ok
Update Database
present progress
Present Progress
Progress saved to Database
Update Record
admin's approval
Approve or Mark as Absent
save the marks
Save the Marks for progress
marks saved
admin approved or marked as absent
Save to github
Post Progress
Save Record to Github
Saved to github


Post Notice
post on facebook
Update Notice Bar
post to facebook
update it on UI
updated