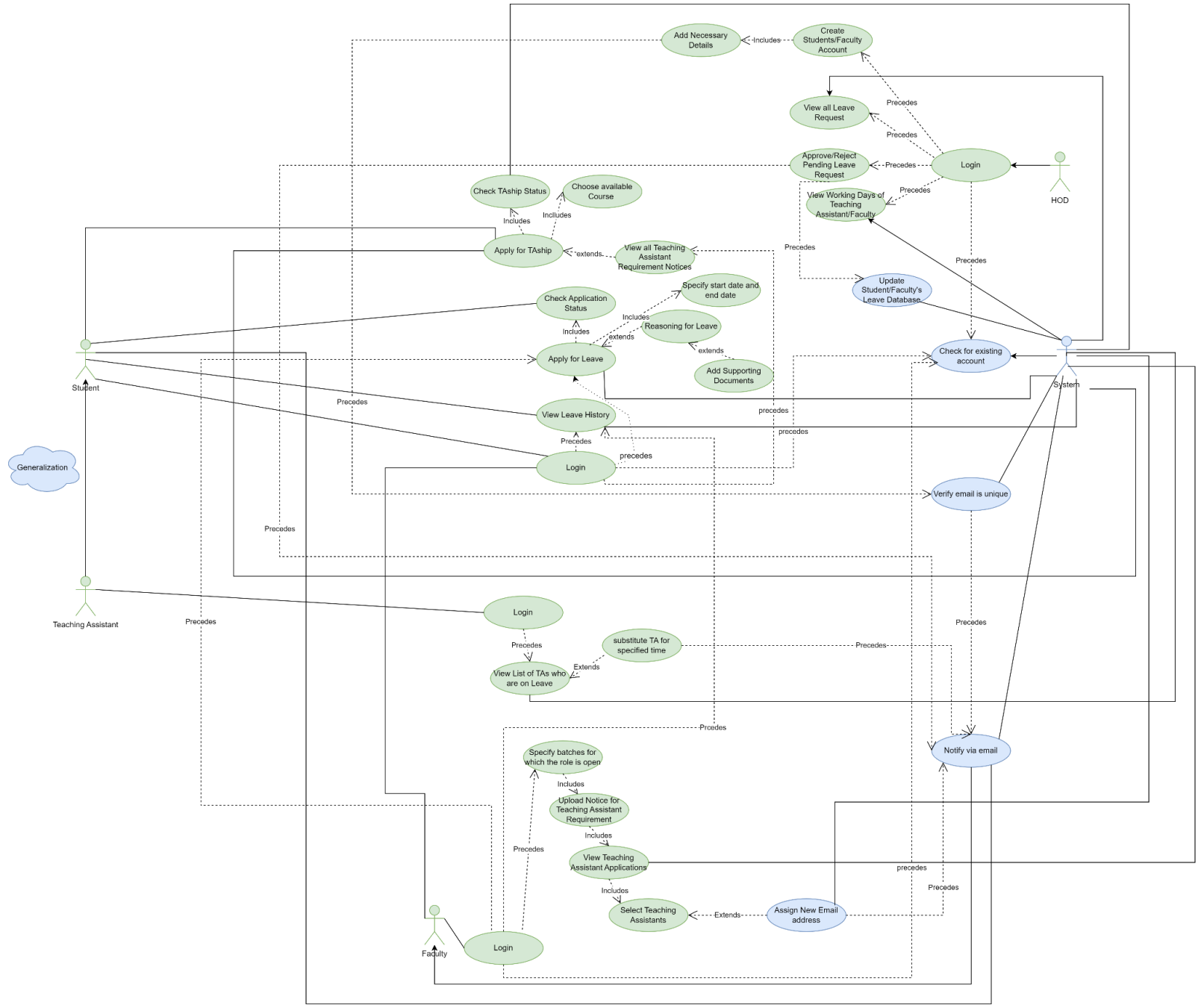
Software Engineering IT314
Project: Student leave and Teaching Assistantship management
Group: 9

- **Group Members:**
  - Preksha Anand - 202001061
  - Maharth Thakar - 202001069
  - **Meet Patel - 202001074**
  - Kartik Dangi - 202001079
  - Harsh Kathiriya - 202001088
  - Kuldipsinh Gohil - 202001092
  - Neha Miglani - 202001097
  - Harsh Anand - 202001101
  - Darshan Kheni - 202001092

Lab : 03

Date : 22-02-2023

Use case diagram:

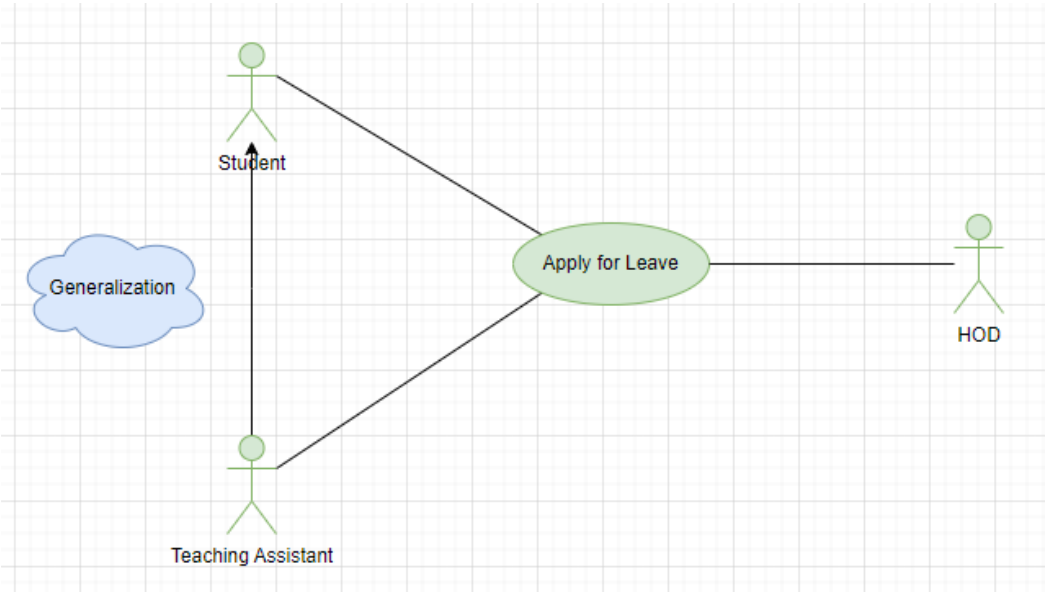


For better view: [Use_Case_diagram]

❖ Relationship among the use cases and actors:

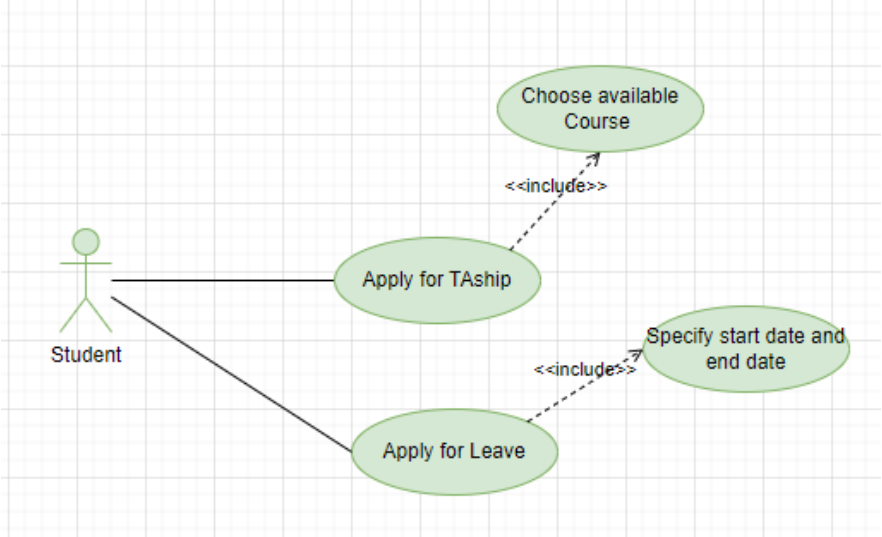
➢ Generalization of an Actor:

Teaching Assistant is generalized with students.

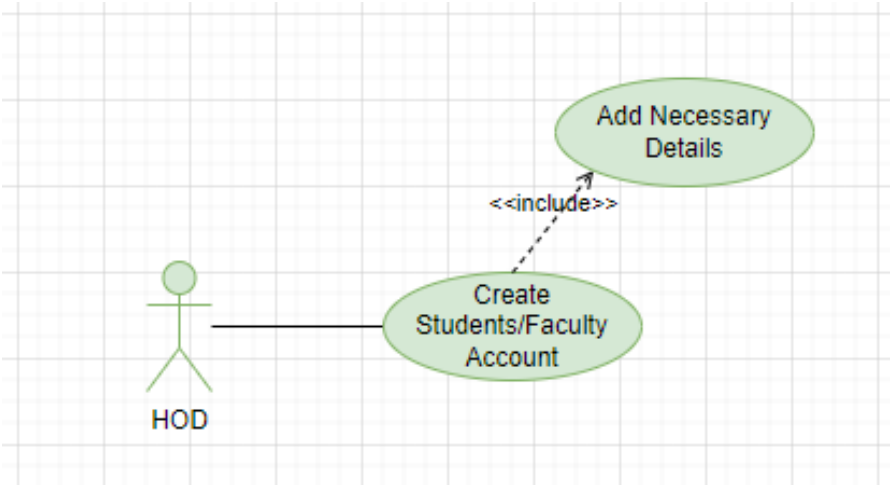


➢ **Include relationship between two use cases**

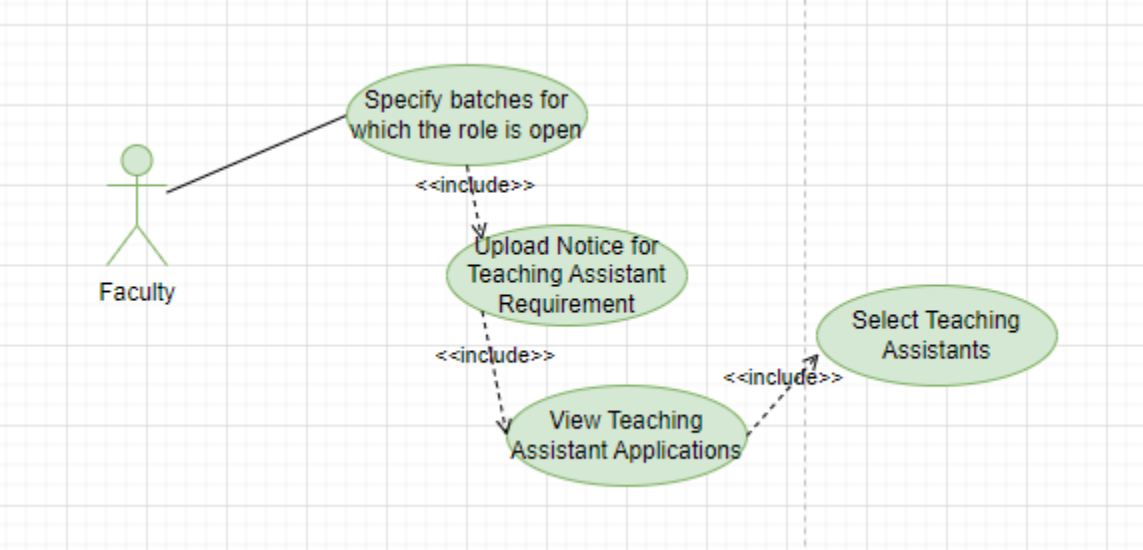
1. Students applying for TAship must choose an available course and if a student is applying for a leave he/she must specify start date and end date.



2. HOD has the responsibility of creating Students / Faculty accounts and upon adding he/she must add necessary information.

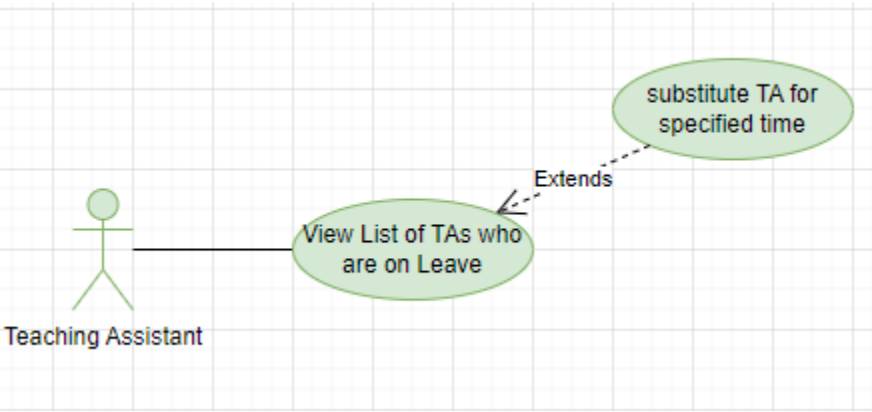


3. Faculty upon specifying batches for which the role is open MUST upload notice for TA requirement and after that he/she must select TAs for the same.

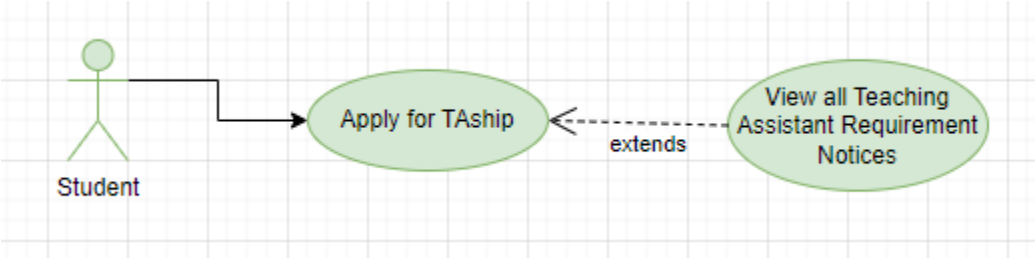


➢ **Extend relationship between two use cases**

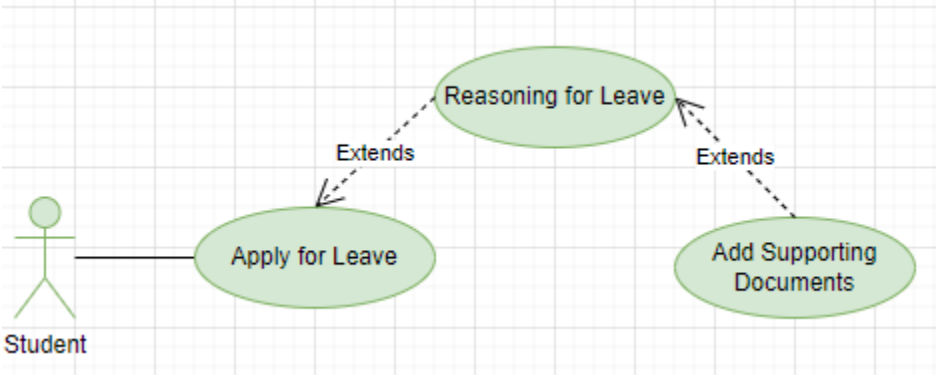
1. If a TA is on leave the other TAs for the same course should look out for a substitute among themselves.



2. If any student wants to apply for a TAship, he/she can view all teaching assistant requirement notices that he/she may want to see.



3. If a student applies for leave then he/she can write the reason for the same and also add the required documents if needed.



(2) use case textual description for each use case:

USE CASE 1

1. **Name:** Leave application
2. **Actors:** HOD, Students, TA, Faculty
3. **Goals:** Apply for leave
4. **Preconditions**: Initially till the current date there are no leave applications to be accepted / rejected by the HOD.
5. **Postconditions**: Either the leave could be approved or it could be rejected.
6. **Description:** Any student, faculty, TA could apply for leave and upon approval by the HOD there are few cases :
    - If a student gets approved corresponding faculties and TAs(of student's chosen courses) should get notified about the same.
    - If the leave of a TA gets approved, corresponding faculties and TAs (of same course) should get informed about the same.
    - If leave of a faculty gets approved, the other faculties of the same courses should get informed about the same.
7. **Trigger**: When a leave request is made by a faculty/TA/student
8. **Summary**: When a particular actor makes a trigger the HOD has two options: either approve it or reject it.
   Upon approval appropriate notifications will be made to different stakeholders.
9. **Main Flow**:
    - Any user first logins into the system and requests leave by describing a particular time period.

- HOD logins into the system and looks upon the pending requests and accepts the legitimate requests .
- Upon approval there are there sub flows
    - If a student leave is authorized, the respective faculties and TAs (of the student's chosen courses) should be informed.
    - If a TA's leave is authorized, the respective faculties and TAs (of the same course) should be notified.
    - If a faculty member's leave is authorized, the other faculty members teaching the same courses should be notified.

10. **Alternate Flows**:
    - If HOD rejects the leave of any user then that particular stakeholder should be notified about the same.

USE CASE 2

1. **Name:** TAship Application
2. **Actors:** Student, Faculty
3. **Goals:** Apply for TAship
4. **PreConditions:** Till date there are not any TAship applications to be accepted/rejected by the faculty.
5. **PostConditions:** There are two choices to be either accepted or rejected at the end.
6. **Description:** A student apply for the TAship then there are the following cases:
    - The application can be accepted by the faculty only of the respective subject and the student be notified for the same.
    - The application if then rejected by the faculty then also it should be notified to the student.

7. **Trigger:** The student applies for the TAship which acts as a trigger for this use case.
8. **Summary:** The student applies for the TAship then the faculty receives the request and then the faculty has the right to accept or reject the application accordingly.
9. **Main Flow:**
    - The student opens the system and enters the credentials to login to the system.
    - Then the student applies/requests for the TAship.
    - The faculty opens the system through the credentials.
    - Then the faculty receives the request and then he accepts the request.

- The student gets notified if he/she has been approved for the TAship.

10. **Alternate Flow:**
    - The application of the student gets rejected by the faculty. The student should be notified regarding the same.

(3) Non-Functional Requirements:

❖ **Performance:**

- The system should be able to handle a large number of requests from different users like students, TA's etc. at the same time without any performance issues.
- Response time should be fast and reliable, and the system should be able to handle peak loads without any downtime.

❖ **Security:**

- The system should be designed with security in mind and should use appropriate measures to protect sensitive data from unauthorized access, such as authentication, access control, and data encryption.
- Different types of users should have different access controls over different sections of the system.
- This will make sure that the system and the data will be well protected even in the case of external attacks.

❖ **Availability:**

- The system should operate continuously, with little downtime required for upgrades and maintenance.
- The system should be available 24/7, as students/TA's may need to submit leave requests outside of normal business hours.
- Any downtime must be planned and announced to the users in advance.
- It should also have a reliable backup and recovery plan to ensure data is not lost in the event of a system failure.

❖ **Usability:**

- The system should have a basic and intuitive user interface that does not need substantial training or technical understanding, making it user-friendly and straightforward to operate.
- All the functionalities available in the system should have proper description so that there is no confusion regarding any functionality among users.

❖ **Scalability:**

- The system should be able to handle an increasing number of users and requests as the organization grows, without compromising its performance or functionality.
- Users should be assured that the system is adaptable to the changing needs and demands.

❖ **Reliability:**

- The software must operate reliably and be bug-and-error-free. It should undergo thorough testing to make sure it performs as planned.
- The software must operate properly even in the case of critical failures.
- Also, if a software problem emerges, for instance, proper maintenance must be carried out within a stipulated time.

❖ **Data Integrity:**

- The system should ensure accuracy, completeness and accuracy of data, with appropriate data validation and error checking mechanisms.
- This is needed to keep the data safe, i.e. if any error or any unavoidable circumstance occurs, no alteration in the data must happen.
- The stored data must be unaffected in every situation.

❖ **Compatibility:**

- The user must have minimum hardware requirements.

❖ **Compliance:**

- The system should comply with all relevant laws, regulations, and industry standards, such as data privacy laws or accessibility guidelines.
- This can help ensure that the institution avoids legal or reputational risks associated with non-compliance.

References:

https://creately.com/blog/diagrams/use-case-diagram-relationships/

https://www.sciencedirect.com/topics/computer-science/case-description