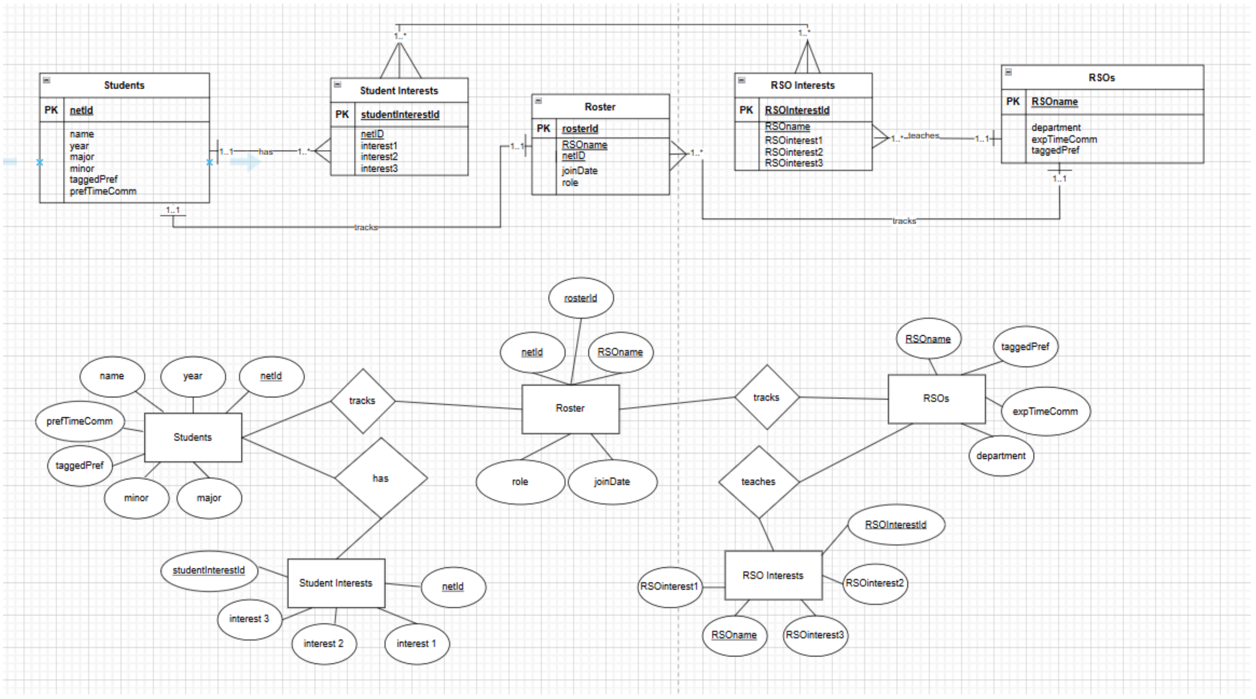
1. **Draw your ER/UML diagram**
    a. **(3.) Your project must involve at least 5 entities**
    b. **(3.) It should involve at least two types of relationships (i.e., 1-1, 1-many, and many-many).**



2. **Explain your assumptions for each entity and relationship in your model. Discuss why you've modeled something as an entity rather than an attribute of another entity. Describe the cardinality of relationships, like why a student is linked to only one advisor. These assumptions might come from customer requirements or application constraints. Please clarify them.**

a. <u>Students</u> is its own entity because each student has different characteristics and preferences. The attributes of this entity are typically not edited once the student makes an account.
    i. Students Attributes:
        1. netId (Primary Key): A unique identifier for each student.
        2. Name: The student's name.
        3. Year: The academic year the student is in (Freshman, Sophomore, Junior, Senior).
        4. Major: The student's major.
        5. Minor: The student's minor.
        6. taggedPref: Represents a predefined preference or categorization for a student organization. Examples of tagged preferences would be categories

like social, technical, and professional. They are meant to be broader categories to help narrow down the search.
7. prefTimeComm: The student's preferred time commitment for an RSO in hours per week.

b. <u>Student Interests</u> is its own entity because it provides a more detailed view of a student's specific interests, which can help recommend suitable student organizations. The student can edit this regularly which is why we separated it from students.
    i. Students Interest Attributes:
        1. StudentInterestId: A unique identifier for each set of student interests that the student can pick from a set of options.
        2. netId**:** A unique identifier for each student. A foreign key for this entity.
        3. Interest 1, 2, 3: The specific interests of the student. Examples of this include robotics, consulting, rocketry, culinary arts. These are meant to be much more specific than taggedPref.

c. <u>Roster</u> is its own entity to properly track which RSO each student eventually chooses to join. This helps ensure that the student and the RSO they join are correctly tracked and that the correct interests were matched.
    i. Roster Attributes:
        1. rosterId: A unique identifier for each roster detailing the club that the student decided to join.
        2. netId**:** A unique identifier for each student. A foreign key for this entity.
        3. RSOname**:** The unique name of the RSO the student is a member of. A foreign key for this entity.
        4. joinDate**:** When the student decided to join the RSO that they chose.
        5. Role: The part that the student plays in their RSO.

d. <u>RSO Interests</u> is its own entity because each organization caters to students with different interests and values. This can be regularly updated by the RSO admin which is why it is separated from the RSO entity.
    i. RSO Interest Attributes:
        1. RSOInterestsId: A unique identifier for the interest an RSO caters to. This can be edited by RSO representative.
        2. RSOname: The unique name of the RSO. This is a foreign key for this entity.
        3. RSOinterest 1, 2, 3: The different interests the RSO caters to. Examples of this include robotics, consulting, rocketry, culinary arts. These are meant to be much more specific than taggedPref.

e. <u>RSOs</u> is its own entity because there are multiple RSOs with various associated preferences and that cater to different students. This information is collected from OneIllinois and is typically not edited by the RSO admin.
    i. RSO's Attributes:

1. RSOname (Primary Key): A unique identifier for each RSO.
2. Department: The department the RSO is most closely associated with.
3. expTimeComm: The expected time commitment for the RSO.
4. taggedPref: Represents a predefined preference or categorization for the student organization. Examples of tagged preferences would be categories like social, technical, and professional. They are meant to be broader categories to help narrow down the search.

f. Students to Student Interests: One-to-many relationship because a student can have multiple interests.
g. Students to Roster: One-to-one relationship since each entry in the Roster corresponds to one student. We are assuming each student is only matched to one RSO for simplicity.
h. RSOs to Roster: One-to-many relationship because many students tracked in the roster may belong to the same RSO.
i. RSOs to RSO Interests: One-to-many relationship because a single RSO can have multiple interests.
j. Student Interests to RSO Interests: Many-to-many relationship because a students interests may match multiple RSO interests and because an RSOs interest may match many students interests.

4. Apply BCNF or 3NF to your schema or show that your schema adheres to one of these normal forms.

$Students = \{netID, name, year, major, minor, taggedPref, prefTimeComm\}$

$R_0 = \{netID \rightarrow name, year, major, minor, taggedPref, prefTimeComm\}$

$Students\ Interests = \{studentInterestID, netID, interest1, interest2, interest3\}$

$R_1 = \{studentInterestID \rightarrow netID, interest1, interest2, interest3\}$

$Roster = \{rosterID, netId, RSOname, joinDate, role\}$

$R_2 = \{rosterID \rightarrow netId, RSOname, joinDate, role\}$

$RSOs = \{RSOname, department, expTimeComm, taggedPref\}$

$R_4 = \{RSOname \rightarrow department, expTimeComm, taggedPref\}$

$RSO\ Interests = \{RSOInterestID, RSOname, RSOinteres1, RSOinterest2, RSOinterest3\}$

$R_5 = \{RSOname \rightarrow RSOname, RSOinteres1, RSOinterest2, RSOinterest3\}$

Because the primary key is a minimal superkey for each relation, each entity table conforms to the BCNF normalization form and no further decomposition is needed.

5. Convert your conceptual database design (ER/UML) to the logical design (relational schema). Note that a relational schema is NOT an SQL DDL command.

Students(

netId:VARCHAR(50) [PK],

name:VARCHAR(100),

year:INT,

minor:VARCHAR(50),

major:VARCHAR(50),

taggedPref:VARCHAR(50),

prefTimeComm:VARCHAR(100))

Student_Interests(

netId:VARCHAR(50) [PK] [FK to Students.netId],

interest1:VARCHAR(100) [PK],

interest2:VARCHAR(100) [PK],

interest3:VARCHAR(100) [PK])

Roster(

rosterId: VARCHAR(50) [PK]

netId:VARCHAR(50) [PK] [FK to Students.netId],

RSO_name:VARCHAR(100) [PK] [FK to RSOs.RSOname],

joinDate :VARCHAR(100),

role: VARCHAR(100))

RSO_Interests(

RSOInterestId: VARCHAR(50) [PK],

RSOname:VARCHAR(100) [PK] [FK to Rsos.RSOname],

RSOInterest1:VARCHAR(100) [PK],

RSOInterest2:VARCHAR(100) [PK],

RSOInterest3:VARCHAR(100) [PK])

RSOs(

RSOName:VARCHAR(100) [PK],

department:VARCHAR(100),

expTimeComm:VARCHAR(100),

taggedPref:VARCHAR(50))