# Information System of Silver Oaks Cooperative School Design Proposal

1. State mission statement(s) and mission objectives for the client.

1. Silver Oaks Cooperative School Information System is a system that helps the school to manage all the applicants' information during the application process (including subscribers) and allocate those accepted students to different classes and staff (teachers).
2. Apart from those, school can also manage the information of the enrolled students and award the top students by sorting the grades.
3. Besides, by building relationships among the tables, school can easily deal with staff resignation, student transfer and etc..
4. The system can also manage the subscribers who are interested in getting up-to-date information of the school.

2. Finalize ER schema and diagram.

## Entities, Attributes, and Primary Keys:

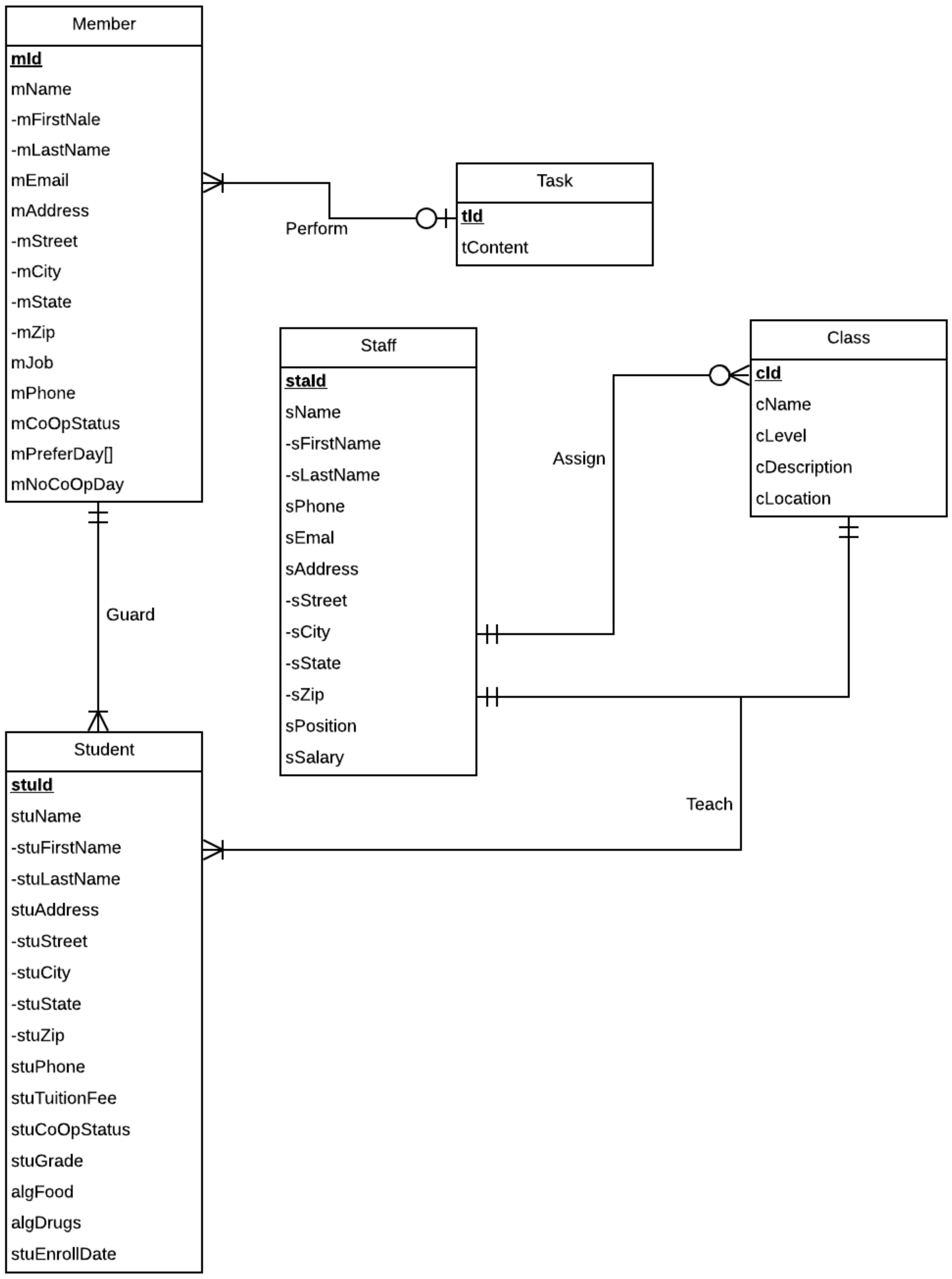
Student: **stuId**, stuName, -stuFirstName, -stuLastName, stuAddress, -stuStreet, -stuCity, -stuState, -stuZip, stuPhone, stuTuitionFee, stuCoOpStatus, stuGrade[], algFood[], algDrugs[], stuEnrollDate
Staff: **staId**, sName, -sFirstName, -sLastName, sPhone, sEmail, sAddress, -sStreet, -sCity, -sState, -sZip, sPosition, sSalary
Member: **mId**, mName, -mFirstName, -mLastName, mEmail, mAddress, -mStreet, -mCity, -mState, -mZip, mJob, mPhone,mCoOpStatus, mPreferDay[],mNoCoOpDay
Class: **cId**, cName, cLevel, cDescription,cLocation
Task: **tId,** tContent

## Relationship, Degrees and Participating Entities:

Teach: ternary relationship
    1 student and 1 staff to 1 class
    1 class and 1 student to 1 staff
    1 class and 1 staff to 1 or many students
Assign: binary relationship
    1 staff to 0 or many classes
    1 class to 1 staff
Guard: binary relationship
    1 student to 1 member
    1 member to 1 or many students
Perform: binary relationship
    1 member to 0 or 1 tasks

1 task to 1 or many members

## ER Diagram:


Member
mId
mName
-mFirstNale
-mLastName
mEmail
mAddress
-mStreet
-mCity
-mState
-mZip
mJob
mPhone
mCoOpStatus
mPreferDay[]
mNoCoOpDay
Perform
Task
tId
tContent
Staff
staId
sName
-sFirstName
-sLastName
sPhone
sEmal
sAddress
-sStreet
-sCity
-sState
-sZip
sPosition
sSalary
Assign
Class
cId
cName
cLevel
cDescription
cLocation
Guard
Teach
Student
stuId
stuName
-stuFirstName
-stuLastName
stuAddress
-stuStreet
-stuCity
-stuState
-stuZip
stuPhone
stuTuitionFee
stuCoOpStatus
stuGrade
algFood
algDrugs
stuEnrollDate

3. Convert ER model into relational schema and identify primary and foreign keys.

## Relational schema

Student(**stuId**, stuFirstName, stuLastName, stuStreet, stuCity, stuState, stuZip, stuPhone, stuTuitionFee, stuCoOpStatus, stuGrade, algFood, algDrugs, stuEnrollDate, *mId*)
Staff(**staId**, sFirstName, sLastName, sPhone, sEmail, sStreet, sCity, sState, sZip, sPosition, sSalary)
Member(**mId**, mFirstName, mLastName, mEmail, mStreet, mCity, mState, mZip, mJob, mPhone,mCoOpStatus, mPreferDay, mNoCoOpDay )
Class(**cId**, cName, cLevel, cDescription,cLocation, *staId*)
Task(**tId**, ***mId*** , tContent)
Teach(***cId, staId, stuId***)

4. Determine functional dependencies and perform normalization to 3NF.

FDs in 3NF

stuId —> stuFirstName, stuLastName, stuStreet, stuCity, stuState, stuZip, stuPhone, stuTuitionFee, stuCoOpStatus, stuGrade, algFood, algDrugs, stuEnrollDate
sId —> sFirstName, sLastName, sPhone, sEmail, sStreet, sCity, sState, sZip, sPosition, sSalary
mId —> mFirstName, mLastName, mEmail, mStreet, mCity, mState, mZip, mJob, mPhone,mCoOpStatus, mPreferDay, mNoCoOpDay, *stuId*
cId —> cName, cLevel, cDescription,cLocation
tId, mId —> tContent
cId, staId, stuId —>

Normalization:
Student(**stuId**, stuFirstName, stuLastName, stuStreet, stuCity, stuState, stuZip, stuPhone, stuTuitionFee, stuCoOpStatus, stuGrade, algFood, algDrugs, stuEnrollDate, *mId*)
Staff(**staId**, sFirstName, sLastName, sPhone, sEmail, sStreet, sCity, sState, sZip, sPosition, sSalary)
Member(**mId**, mFirstName, mLastName, mEmail, mStreet, mCity, mState, mZip, mJob, mPhone,mCoOpStatus, mPreferDay, mNoCoOpDay)
Class(**cId**, cName, cLevel, cDescription,cLocation, staId)
Task(**tId**, ***mId*** , tContent)
Teach(***cId*, *staId*, *stuId***)

5. Generate business rules and determine referential integrity actions.

**Business Rules:**

[R1] When a child was enrolled as a student or change guardian, the corresponding member information should be updated.

[R2] When a student graduates from school or drops out school, the corresponding member information should be deleted.

[R3] When a task is performed by a member, the corresponding task information should be updated .

[R4] When a member stops performing a task, the corresponding task information should be deleted.

[R5] When a staff is assigned to a class, the corresponding class information should be updated.

[R6] When a staff stops teaching a class, the corresponding class information should be deleted.

[R7] When a class is assigned a staff and registered by a student, the class, the staff and the student information cannot be updated or deleted in the database.

## Referential Integrity:

| Relation | Foreign Key | Base Relation | Primary Key | Business Rule | Constraint: ON DELETE | Business Rule | Constraint: ON UPDATE |
|---|---|---|---|---|---|---|---|
| student | mId | member | mId | R2 | SET NULL | R1 | CASCADE |
| task | mId | member | mId | R3 | CASCADE | R4 | CASCADE |
| class | staId | staff | staId | R5 | CASCADE | R6 | CASCADE |
| Teach | staId | staff | staId | R9 | NO ACTION | R9 | NO ACTION |
| Teach | cId | Class | cId | R9 | NO ACTION | R9 | NO ACTION |
| Teach | stuId | Student | stuId | R9 | NO ACTION | R9 | NO ACTION |

6.	Describe sample data for every relation.

```
CREATE TABLE [Student] (
        stuId CHAR(9) NOT NULL,
        stuFirstName VARCHAR(20),
        stuLastName VARCHAR(20),
        stuStreet VARCHAR(40),
        stuCity VARCHAR(20),
        stuState CHAR(2),
        stuZip CHAR(5),
        stuPhone CHAR(12),
        stuTuitionFee DECIMAL(10,2),
        stuCoOpStatus VARCHAR(10),
        stuGrade VARCHAR(10),
        algFood VARCHAR(100)
        algDrugs VARCHAR(100)
        stuEnrollDate DATE,
        CONSTRAINT pk_Student_stuId PRIMARY KEY (stuId),
        CONSTRAINT fk_Student_mId FOREIGN KEY (mId)
                REFERENCES [Member] (mId)
                        ON UPDATE CASCADE
                        ON DELETE SET NULL
         );

CREATE TABLE [Staff] (
        staId CHAR (9) NOT NULL,
        sFirstName VARCHAR (40),
        sLastName VARCHAR (40),
        sPhone CHAR (12),
        sEmail VARCHAR(40),
        sSalary DECIMAL (7,2),
        sStreet VARCHAR(20),
        sCity CHAR(10),
        sState CHAR(2),
        sZip CHAR(5),
        sPosition VARCHAR(20),
        sSalary DECIMAL(7,2)
        CONSTRAINT pk_Staff_staId PRIMARY KEY (staId)
        );

CREATE TABLE [Class]  (
        cId CHAR (9) NOT NULL,
        cName VARCHAR (40),
        cLevel VARCHAR (20),
        cDescription VARCHAR(60),
        cLocation VARCHAR(40),
        staId CHAR(9),
        CONSTRAINT pk_Class_cId PRIMARY KEY (cId),
        CONSTRAINT fk_Class_staId FOREIGN KEY(staId)
                REFERENCES [Staff] (staId)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE
        );

CREATE TABLE [Member] (
        mId CHAR(9) NOT NULL,
        mFirstName VARCHAR(20),
        mLastName VARCHAR(20),
        mEmail VARCHAR(20),
        mStreet VARCHAR(40),
        mCity VARCHAR(20),
        mState CHAR(2),
        mZip CHAR(5),
        mJob VARCHAR(40),
        mPhone CHAR(12)
        stuId CHAR(9),
        CONSTRAINT pk_Member_gId PRIMARY KEY (mId),
         );

CREATE TABLE [Teach]  (
        cId CHAR (9) NOT NULL,
        stuId CHAR (9) NOT NULL,
        staId CHAR(9) NOT NULL,
        CONSTRAINT pk_Teach_cId_stuId_staId PRIMARY KEY (cId,stuId,staId),
        CONSTRAINT fk_Teach_staId FOREIGN KEY(staId)
                REFERENCES [Staff] (staId)
                        ON DELETE NO ACTION
                        ON UPDATE NO ACTION,

        CONSTRAINT fk_Teach_stuId FOREIGN KEY(stuId)
                REFERENCES [Student] (stuId)
                        ON DELETE NO ACTION
                        ON UPDATE NO ACTION,

        CONSTRAINT fk_Teach_cId FOREIGN KEY(cId)
                REFERENCES [Class] (cId),
                        ON DELETE NO ACTION
                        ON UPDATE NO ACTION,

        );

CREATE TABLE [Task]  (
        tId CHAR (9) NOT NULL,
        mId CHAR (9) NOT NULL,
        tContent VARCHAR(40),
        CONSTRAINT pk_Task_tId_mId PRIMARY KEY (tId,mId),
        CONSTRAINT fk_Task_mId FOREIGN KEY(mId)
                REFERENCES [Member] (mId)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE
        );
```