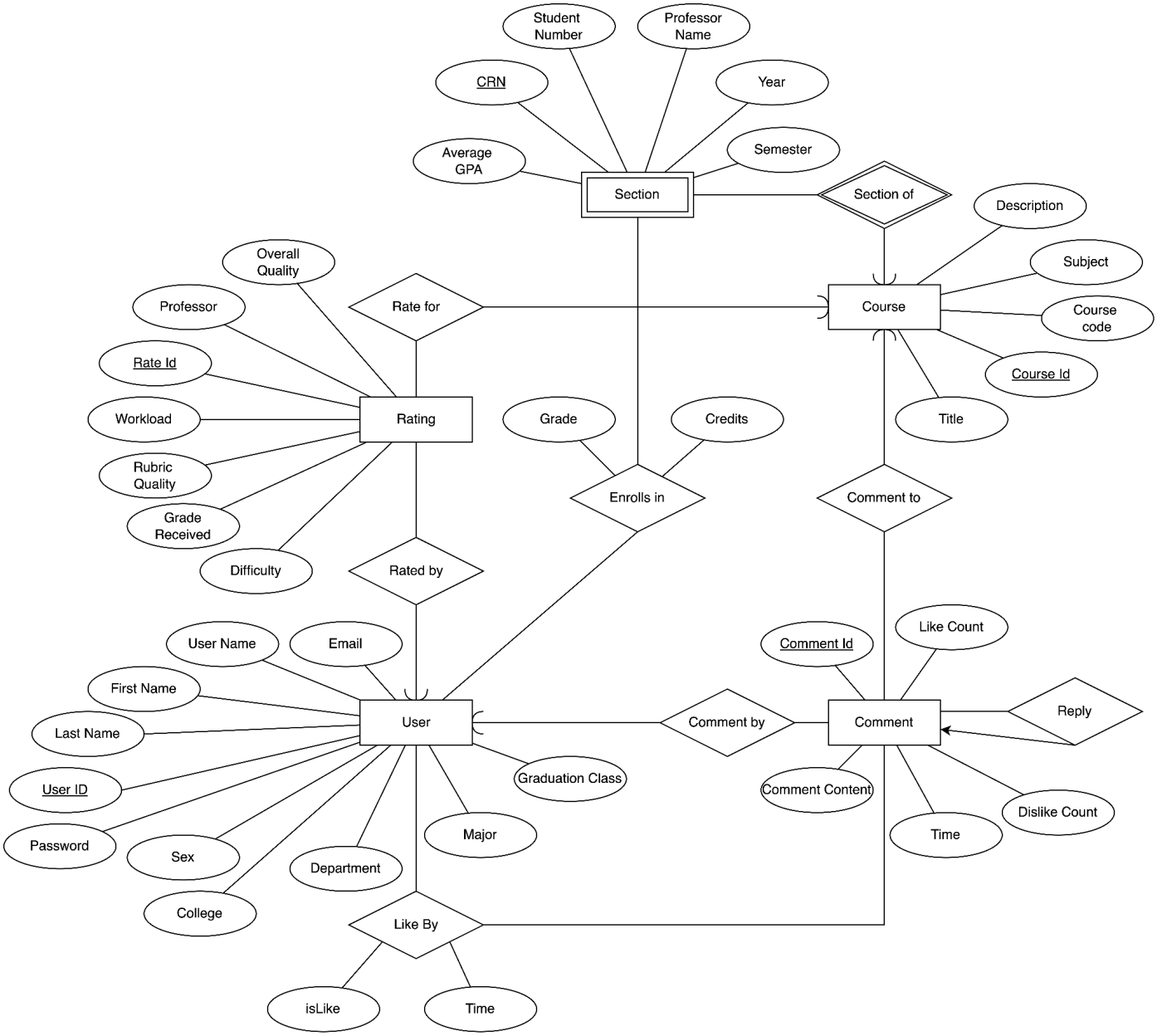
ER Diagram:


Student Number
Professor Name
CRN
Year
Average GPA
Semester
Section
Section of
Description
Subject
Course
Course code
Course Id
Title
Overall Quality
Professor
Rate for
Rate Id
Workload
Rating
Rubric Quality
Grade Received
Difficulty
Rated by
Grade
Credits
Enrolls in
Comment to
User Name
Email
First Name
Last Name
User
User ID
Password
Sex
College
Department
Major
Graduation Class
Like By
isLike
Time
Comment Id
Like Count
Comment by
Comment
Reply
Comment Content
Time
Dislike Count

Schema:

User(userID:INT [PK], userName: VARCHAR (20), email: VARCHAR (50), firstName: VARCHAR(20), lastName: VARCHAR (20), password: INT);

Course (courseID: INT [PK], description: VARCHAR (255), subject: VARCHAR (25), title: VARCHAR (25), courseNumber: INT)

Section(CRN: INT [PK], courseID: INT [PK, FK to Course.courseID], professor: VARCHAR(35), semester: VARCHAR(2), averageGPA: REAL, year: INT, studentNumber: INT)

Comment(commentID: INT [PK], commenter: INT [FK to User.userID], course INT: [FK to Course.courseID], parentID INT: [FK to Comment.CommentID], likes: INT, dislikes: INT, time: DATETIME, content: VARCHAR (1024));

Rating (rateID: INT [PK], rater: INT [FK to User.userID], course: INT [FK to Course.courseID], courseQuality: INT, professor: VARCHAR (35), workload: INT, rubricQuality: INT, gradeReceived: VARCHAR(10), difficulty: INT)

Likes(user: INT [PK, FK to User.User ID], comment: INT [PK, FK to Comment.commentID], isLike: Boolean, Time: DATETIME)

Enrollments(student: INT [PK, FK to User.UserID], CRN: INT [PK, FK to Section.CRN], course_id [PK, FK to Section.courseID], grade: Char, credits: INT)

Assumptions and Descriptions for each table:

We will describe and make assumptions for each table and their attributes individually.

User Table:

This table stores the user's login Information. Every user must create an account using his/her email account. Once an account is created, the application will automatically associate the user account with an unique id.

Attributes:

- userID: Automatically generated by application, the primary key.
- userName: The user's choice of user name. Can't be null.
- firstName: The user's first name, this field can't be null.
- lastName: The user's last name, this field can't be null.
- password: The user's (hased) password, this field can't be null.
- email: The user's email address, can't be null, can be duplicated.

Course:

This table stores basic information of UIUC's courses.

Attributes:

- courseID: Automatically generated by application, the primary key.
- description: The course's description, collected from the UIUC dataset.
- subject: The subject of the course (ex: CS).
- title: The course's title (ex. Database Systems).
- numberNumber: The course number (ex. 411).

Section:

This table presents a single section of a course and is uniquely defined by its course and CRN. A course can have multiple sections, but a section can only belong to one course.

Attributes:

- CRN: The sections' CRN, collected from the UIUC dataset. Primary Key.
- course: The course in which the section belongs to. It is the primary key as well as the foreign key referencing the Course table.
- professor: The full name of the professor teaching this section, not null.
- studentNumber: Number of students enrolled in this section. not Null.
- semester: The semester (spring/fall) in which it was being taught.
- averageGPA: The average gpa of this section.
- year: The year during which the section is taught.

Comment:

This is the comment entity, representing an user's comment to a course. The comment is uniquely identified the ID auto generated by the application. It must be associated to exactly one user who made the comment and to exactly one course it comments about. The comment also can be a reply to at most one other comment.

Attributes:

- commentID: Automatically assigned by the application, primary key.
- commenter: The (exact one) user who made this comment, references to the User table. Commenter can't be null.
- course: The (exact one) course to which it is commenting about, references the Course table.
- parentID: The comment to which it replies. When it is null, it is a comment, otherwise it is a reply and the value of parent_ID references to the Comment table itself (the comment it is replying to).
- content: The actual content of the comment, maximally 1024 bytes of size. This field can't be null.
- likes: The number of times other users liked this comment.
- dislikes: The number of times other users disliked this comment.

Rating:

This table stores a user's rating of all kinds of attributes of a course. Each user can make only one rating, and one rating can only rate for exactly one course.

Attributes:

- rateID: Automatically assigned by the application, primary key.
- rater: User who made the rating. This references to the User entity.
- course: The course to which it rates. This field references to the Course entity.

Other attributes must not be null, and is on a 1-10 integer scale.

Likes:

This is a relation table that stores what users liked/disliked what comments. We assumed that a single user may only likes/dislikes a comment once.

- user: The user who made the like/dislike. Primary key. References to the User entity.
- comment: References the comment that this like is act on. Primary key. Foreign key to the Comment entity.
- isLike: A boolean indicating a like or dislike. True if liked, false if disliked, and null if neither (user cancels his/her like or dislike).
- time: The time at which the action was taken.

Relationship Description/Cardinality:

Like_by (Many-to-Many)

This relation is between the User entity and the Comment Entity. User can like/dislike many comments, and a comment can be liked/disliked by multiple users. However, we assumed that each user may only like or dislike once, and a user may not like and dislike a comment at the same time.

Rated by (One-to-Many):

This relation is between the User entity and the Rating entity. An User may create multiple ratings, but each rating may only be created by one user.

Rate for (One-to-Many):

This relation is between the Rating entity and Course entity. A rating must have exactly one course it rates, and a course may be rated multiple times.

Comment to (One-to-Many):

This relation is between the Comment entity and Course entity. A comment must have exactly one course it comments, and a course may be commented multiple times.

Reply (One-to-Many):

This is between the Comment entity itself. A comment may have many other comments replying to itself, but a comment can only reply to one other comment.

Enrollment (Many-to-Many):

One students can enroll in multiple courses, and one course can have many students.