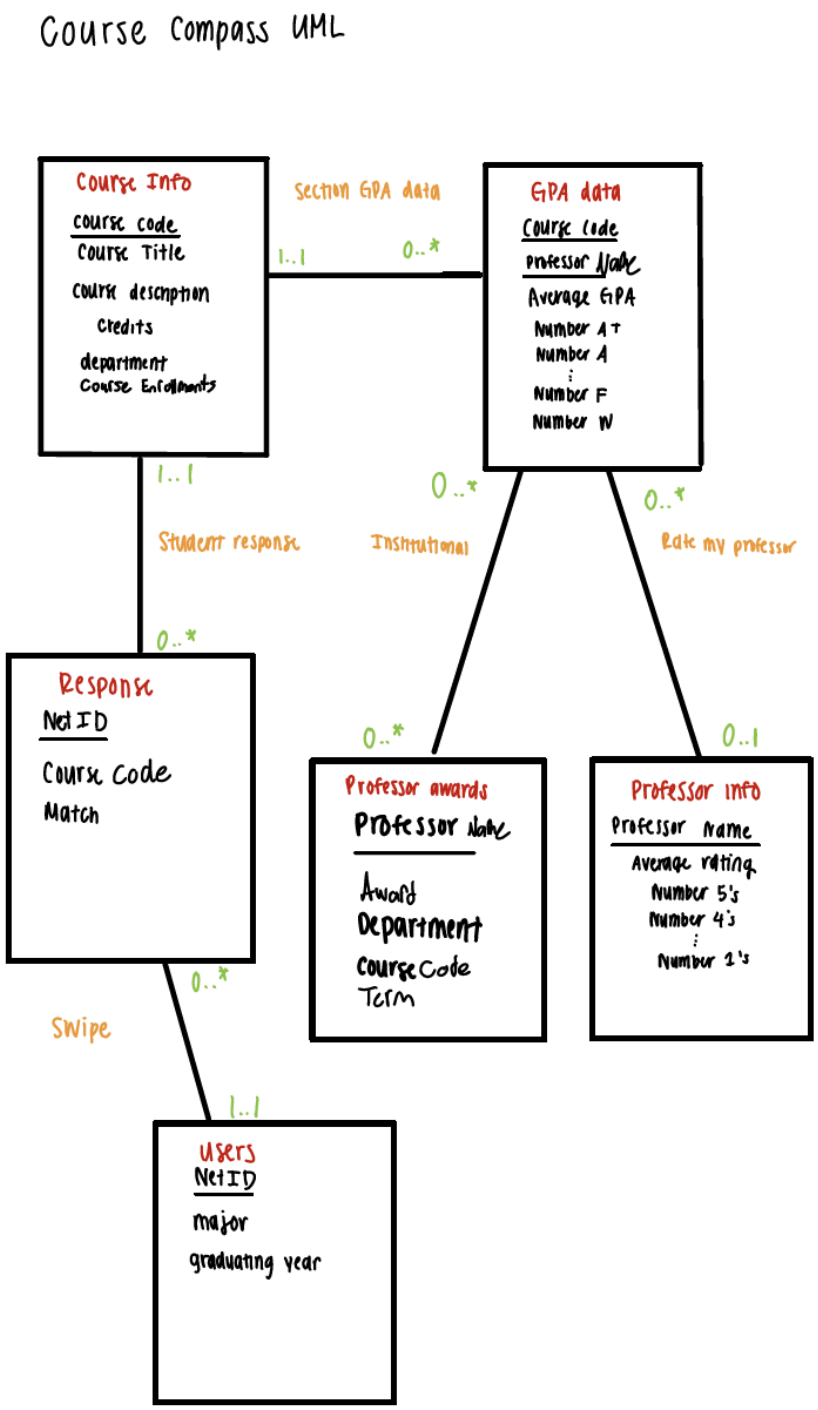
UML DIAGRAM


Course Compass UML
Course Info
course code
Course Title
course description
Credits
department
Course Enrollments
Section GPA data
1..1
0..*
GPA data
Course Code
Professor Name
Average GPA
Number A+
Number A
Number F
Number W
1..1
Student response
0..*
0..*
Institutional
0..*
Rate my professor
Response
NetID
Course Code
Match
0..*
Professor awards
Professor Name
Award
Department
Course Code
Term
0..1
Professor Info
Professor Name
Average rating
Number 5's
Number 4's
Number 1's
0..*
Swipe
1..1
Users
NetID
major
graduating year

ASSUMPTIONS AND EXPLANATIONS

CourseInfo
We assumed that a course can be uniquely identified by a course code, which is a concatenation of the department and a number (ex: CS411, MATH241), thus we make CourseCode a primary key. We assumed that we can encapsulate the details of a course by including CourseTitle, CourseDescription, Credits, CourseDepartment, and CourseEnrollments (# of students who enrolled in the course) as attributes.

GPA data
We assume that each course's GPA can be uniquely identified by a pair of CourseCode and a ProfName, because multiple professors can teach the same course. Additionally, we assume that a course's GPA can be encapsulated by the distribution of letter grades received in that class.

Response
We assume that each user will only "swipe" on a course one time, from there the information will be stored. When swiping, a match/like will be represented by true and a pass/dislike will be represented by false.

ProfessorAwards
We assume that a professor's name can uniquely identify their award. We assume that professors can have multiple awards, as they can receive the same award across multiple years, hence the reason why ProfessorAwards and ProfessorInfo are separated. We also assume that some professors will not have awards.

ProfessorInfo
We also assume that users can rate multiple courses, but each course can only be rated once per user. Furthermore, each course can be uniquely identified by a course code. No two professors teaching the same course have the same name (FirstName, LastName).

We have two different tables for Professors (Professor Data and Professor Awards), as a professor can have multiple awards, therefore we need to separate the table to encapsulate the professors information in one, and another to account for all of the awards a professor has won.

We have two different tables for Course Info and GPA data, because a course can have different GPAs based on the professor teaching the course.

Relations

Section GPA Data: A course can have 0 to many gpa_data values. This is because a course may be taught by multiple professors. A single GPA data value must link to one and only one. This is because gpa data for a course can only be linked to a single course.

Institutional: For a single course, the professor may have 0 to many (0..*) awards. This is because a professor can be nominated for multiple awards. However, a class can have 0 to 1 (0..1) primary instructors listed for that class. A class will not have an instructor listed if the professor teaching that class does not have an award listed for them.

Rate My Professor: A professor can have 0 to many (0..*) classes, as a professor can teach multiple classes. By a similar reasoning, a class can have 0 to 1 (0..1) primary instructors listed for that class. A class will not have an instructor listed if the professor teaching that class does not have a RateMyProfessor page listed for them.

Student Response: A response on our app relates to a single course (1..1) (design decision). A single course can have no responses to many responses (0..*), based on how many students rate the class (all courses will start out at zero student ratings).

Swipe: Each response by the user corresponds to a single course (1…1) which is what the user provides feedback for. A user can have anywhere from zero to multiple responses (0…*), depending on how many swipes they've made, if any.

NORMALIZATION

We used BCNF instead of 3NF as a design choice because it's more restrictive, simple, and intuitive for the database design that we implemented.

Course Info(Course Code, Course Title, Course Description, Credits, Department, Course Enrollments)

GPA Data(Course Code, ProfessorName, Average GPA, CountA+, CountA, CountA-, CountB+, CountB, CountB-, CountC+, CountC, CountC-, CountD+, CountD, CountD-, CountF, CountW)

Response(NetID, Course Code, Match)

Professor Awards(ProfessorName, Award, Department, Course Code, Term)

Professor Info(ProfessorName, Average Rating, CountRatings_5's, CountRatings_4's, CountRatings_3's, CountRatings_2's, CountRatings_1's)

Users(NetId, Major, Graduating Year)

RELATIONAL SCHEMA

CourseInfo(CourseCode:VARCHAR(10)[PK], CourseTitle:VARCHAR(50), CourseDescription:VARCHAR(1000), Credits:INT, Department:VARCHAR(50), CourseEnrollments:INT)

Response(NetId:VARCHAR(20)[PK], CourseCode:VARCHAR(10)[FK to CourseInfo.CourseCode])

Users(NetId:VARCHAR(20)[FK to Response.NetId, Major:VARCHAR(50), GraduationYear:INT)

GPA_Data(CourseCode:VARCHAR(10), ProfessorName:VARCHAR(50)[PK], AverageGPA:Decimal, CountA+:INT, CountA: INT … CountF:INT, CountW:INT)

ProfessorAwards(ProfessorName:VARCHAR(50)[PK],Award:VARCHAR(50),Department: VARCHAR(50), CourseCode:VARCHAR(10), Term:VARCHAR(10))

ProfessorInfo(ProfessorName:VARCHAR(50)[PK], AverageRating: float, CountRatings_5: INT, CountRatings_4: INT, CountRatings_3: INT, CountRatings_2: INT, CountRatings1_1: INT)