Course Compass UML

Course Info
course code
~~Professor Name~~
Course Title

Course description
Credits

Department Name
Course Enrollments

0..*

0..* offered by

1..1

Department
Department Name
Avg GPA
Enrollment
Course Count

GPA data

Average GPA
Number A+
Number A
⋮
Number F
Number W

0..*

Match-result

Response

0..*

rates

0..1

Users
NetID
major
graduating year

Professor info
Professor Name
Average rating
Number 5's
Number 4's
⋮
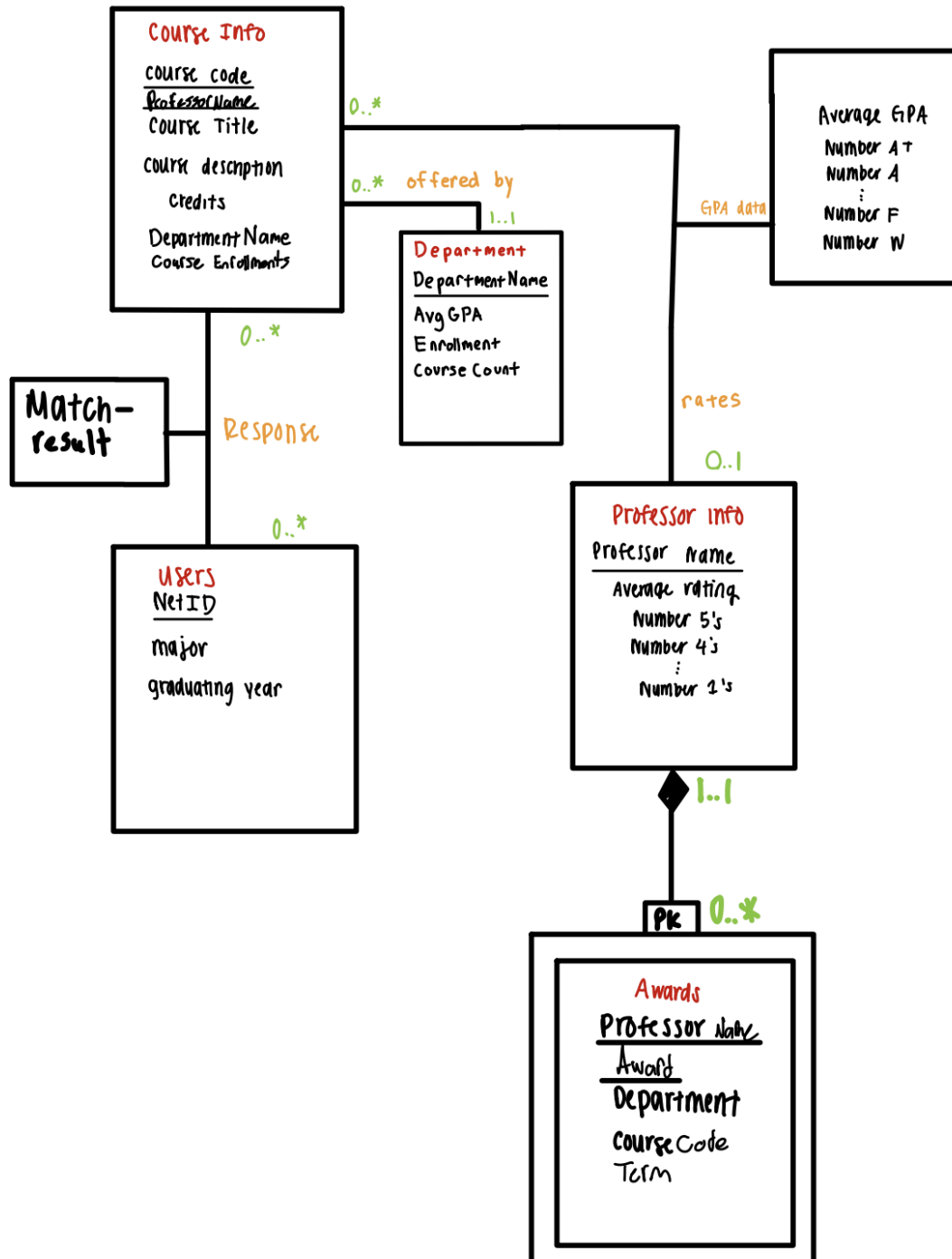Number 1's

1..1

PK 0..*

Awards
Professor Name
Award
Department
Course Code
Term

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 also 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.

Department
We assumed that a department could be uniquely identified by its DepartmentName. We did not need a university as an attribute as this project will only be for UIUC classes. We are also storing additional information about departments such as AvgGPA (average gpa of students in the department), number of students enrolled in the major, and number of courses the department offers.

GPA data Relationship
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 an professor's award can be uniquely identified by the ProfessorName, and the award they got. We assume that professors can have multiple awards (as they can receive the same award across multiple years) and that some professors will not have awards. Additionally, ProfessorAwards data and ProfessorInfo data come from two different datasets (teachers-ranked-as-excellent and RateMyProfessor, respectively).

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

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.

rates: 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.

responses: 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).

Offered by: We assumed that a course is offered by exactly one department. In the event of a cross-listed course, we will use the department given by the data from waf.

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(CourseCode, Course Title, Course Description, Credits, DepartmentName, CourseEnrollments)

Department(<u>DepartmentName</u>, AvgGPA, Enrollment, CourseCount)

Professor Info(<u>ProfessorName</u>, Average Rating, CountRatings_5's, CountRatings_4's, CountRatings_3's, CountRatings_2's, CountRatings_1's)

Awards(<u>ProfessorName</u>, <u>Award</u>, Department, Course Code, Term)

Users(<u>NetId</u>, Major, Graduating Year)

RELATIONAL SCHEMA

CourseInfo(CourseCode:VARCHAR(10)[PK], ProfessorName:VARCHAR(50)[PK, FK to ProfessorInfo:ProfessorName], CourseTitle:VARCHAR(50), CourseDescription:VARCHAR(255), Credits:INT, DepartmentName:VARCHAR(50)[FK to Department:DepartmentName], CourseEnrollments:INT)

Department(DepartmentName: VARCHAR(50)[PK], AvgGPA:FLOAT, Enrollment: INT, CourseCount: INT)

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

Users(NetId:VARCHAR(20)[PK], Major:VARCHAR(50), GraduationYear:INT)

GPA_Data(CourseCode:VARCHAR(10)[PK, FK to CourseInfo:CourseCode], ProfessorName:VARCHAR(50)[PK, FK to ProfessorInfo:ProfessorName], AverageGPA:FLOAT, CountA+:INT, CountA: INT … CountF:INT, CountW:INT)

Awards(ProfessorName:VARCHAR(50)[PK, FK to ProfessorInfo:ProfessorName], Award:VARCHAR(50)[PK], 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, CountRatings_1: INT)