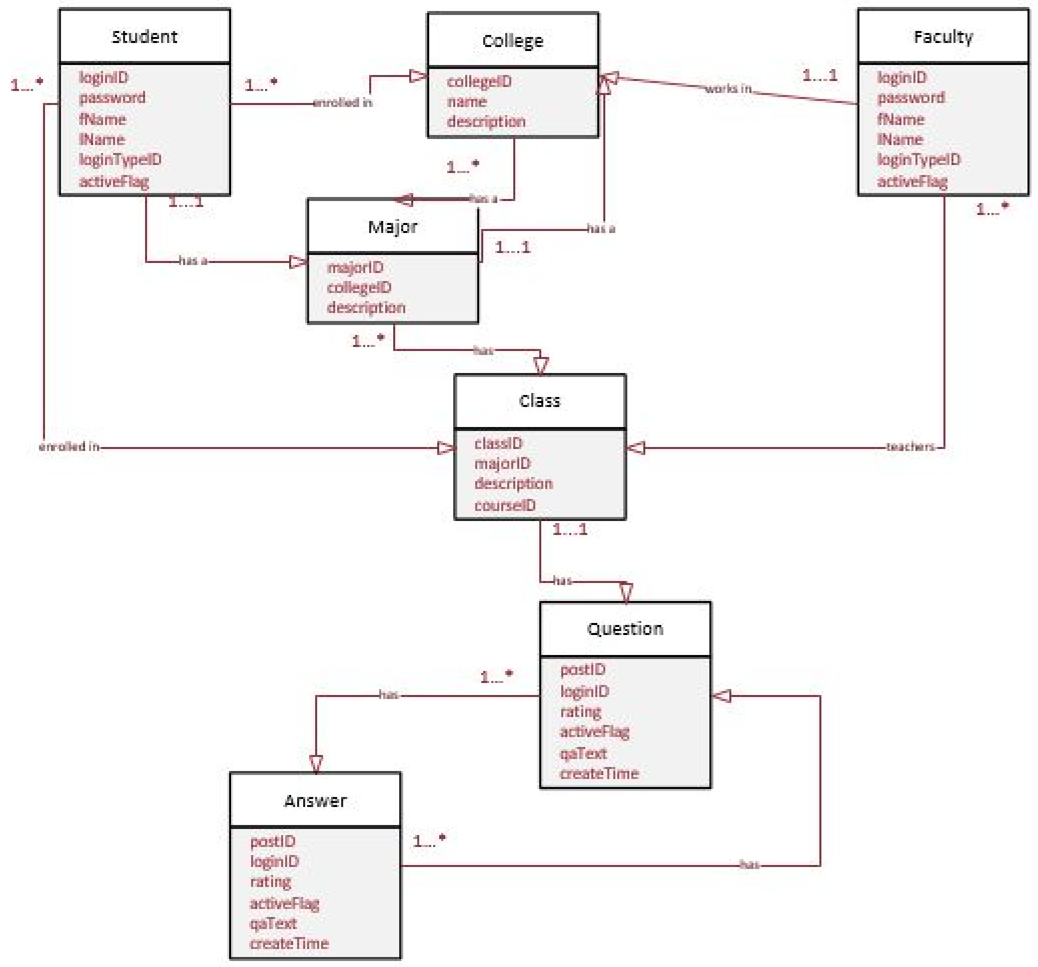
Table of Contents

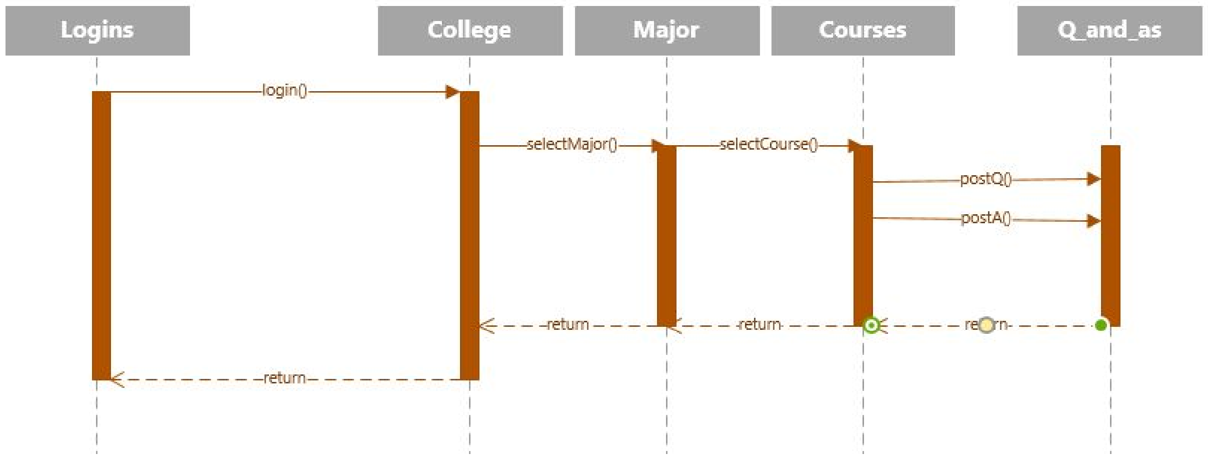
Design Discussion

For the first iteration of the app development we decided to implement posting a question, posting an answer, removing a question and removing an answer. For this iteration there will be no login capabilities. You will simply be able to go to the site and be able to post questions and answers with the ability to edit or destroy the questions. The site hasn't been broken down into the sections based on college, major, or class that we envision the site having yet. Those will be done in later iterations. We have the framework for those things already built into the app, but they are just frameworks.
The classes that were created for this iteration all tie back to the four use cases that we implemented with some basic setup classes. The Logins and Login_types store the logins of users and the different types of logins. All the use cases require that a user is logged in. Q_and_as stores questions and answers that are posted. These connect to the post a question and post an answer use cases. They also connect to a remove a question and answer use cases as well. Courses stores all courses in the systems. Colleges stores all the colleges in the university. Majors stores all majors from all of the colleges. Enrolled_classes stores the classes that the students are enrolled in. _Below are the diagrams for each class.

Design Class Diagram



Design level Sequence Diagrams



Design Decisions

For the design decisions we first sat down and figured out how we wanted the database to look. From there we decided how the classes were going to interact with each other and then the coding began.

Database Design

The database is broken down into similar tables as the classes that we have : Logins to logins, LoginTypes to login_types, QandAs to q_and_as, Classes to courses, Colleges to colleges, Majors to majors, and EnrolledClasses to enrolled_classes. The first is the class and the second is its equivalent in the database. Rails allows for a scaffold to build the database for you, so we used that scaffold. Rails saves the question and answers as title:string, body:text. The foreign keys were generated as author:string, body:text, question:reference. By using this model it doesn't generate all files associated, only the model/class file. The scaffold generates all files. The following are the commands that were used:

- Rake db:migrate

To add columns

- Rails generate add_author_to_questions author:string
- Rake db:migrate