1. List nouns that are candidate classes or attributes

I think these nouns can be candidate classes: faculty, student, course, module, lesson, order, calendar schedule, topic, widget, evaluation, exam, question, registrar's office, semester, section, student progress, everyone, grade, assessment, evaluation of instructor or teaching assistant.

I think these nouns can be candidate attributes: youtube videos, slides, text documents, raw HTML, evaluation, simple essay assignment, submission assignment, exam, essay question, multiple choice question, fill in the blank questions, popularity, fall, spring, full summer, summer 1 and summer 2, seat capacity, faculty, final grade, letter grade, student feedback, username, password, first name, last name, emails, phones, and addresses, benefits, tenure status, parking, bank account info, financial aid info, work-study, scholarship, assignment, rubric.

2. List verbs as candidate relations between classes

I think these verbs can be candidate relations between classes: create, share, author, contain, broken up, rearrange, build, evaluate, answer, enroll, see, assign, teach, keep track of, verify, provide, update, keep an eye on, review.

3. Generalization/specialization (inheritance, if applicable, explain) - show parts of your diagram that specifically illustrates the use of inheritance

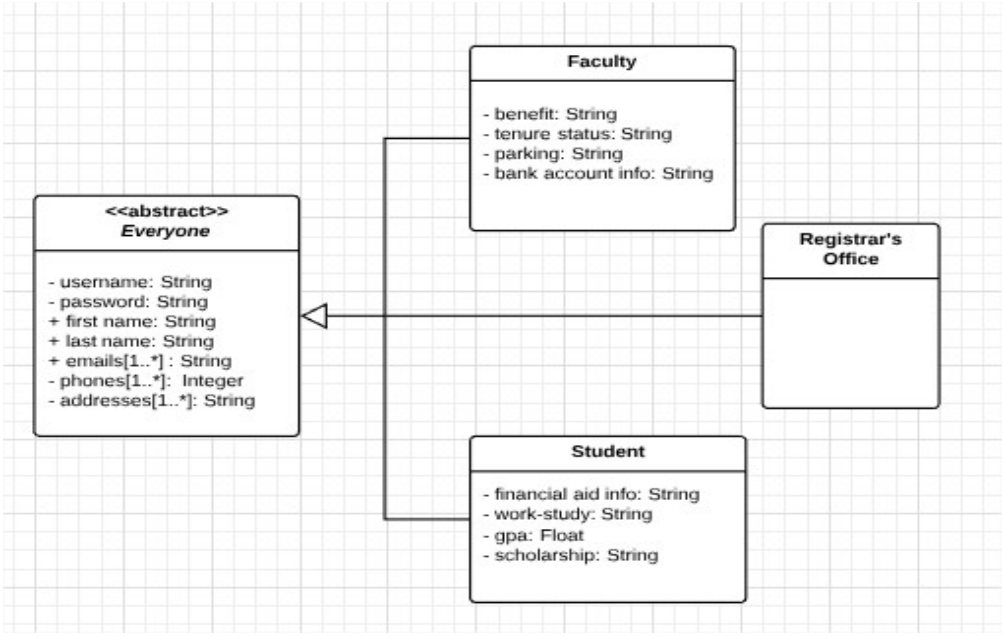


Figure 3.1

The first part I think should be an inheritance shows above in Figure 3.1.

I think class *Everyone* is a superclass, because it contains every individual of the university. And class Student, class Faculty and class Registrar's Office are just collections of different kinds of people who study or work on campus, so they are subclasses.

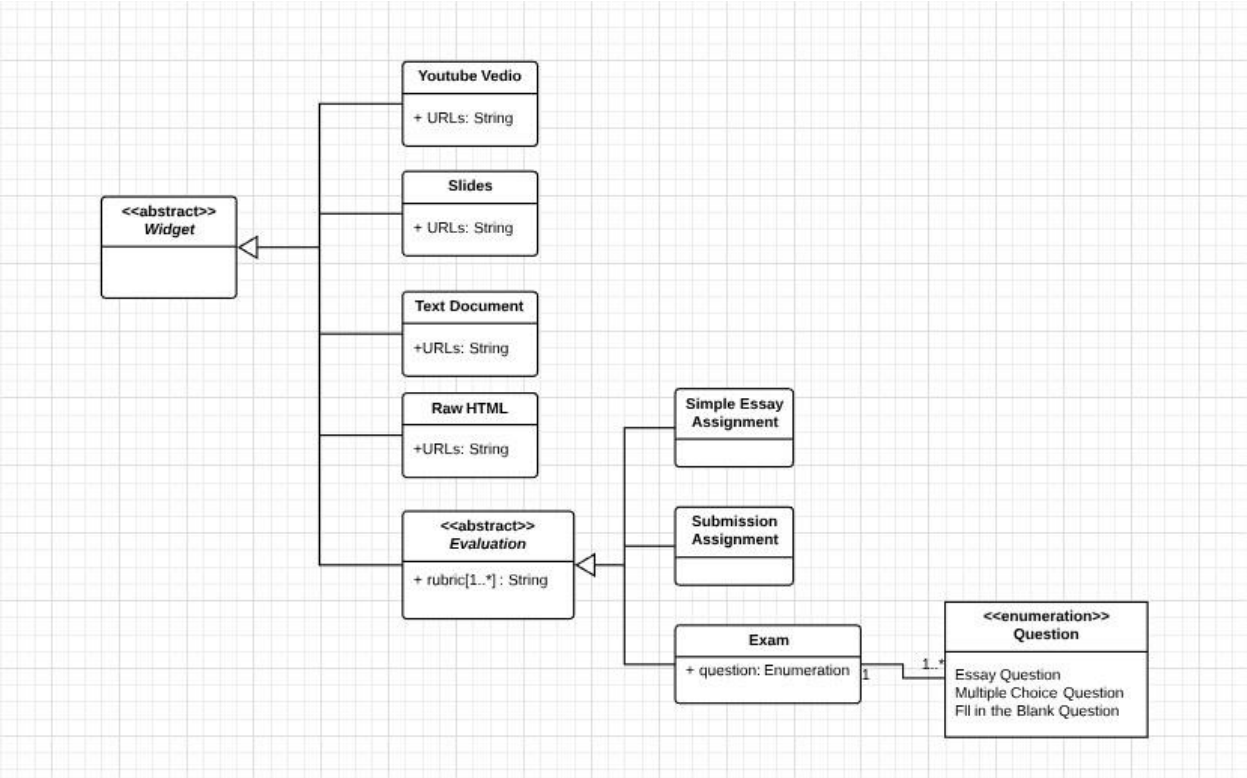


Figure 3.2

The second part I think should be an inheritance shows above in Figure 3.2.

As the problem statement says, youtube videos, slides, text documents, raw HTML and evaluations are just different kinds of widgets. So, class *Widget* is a superclass, and class Youtube Video, class Slides, class Text Document, class Raw HTML and class Evaluation are subclasses.

And the problem statement also says that evaluation widgets can be a simple essay assignment, a submission assignment, or an exam, which means that simple essay assignment, submission assignment and exam are different kinds of evaluations. So, class *Evaluation* is a superclass, and class Simple Essay Assignment, class Submission Assignment and class Exam are subclasses.

4. Associations, aggregation and/or composition, e.g., empty or filled in diamonds (1 to * or 1 to 1..*, if applicable, explain) - capture any lifecycle dependencies between classes using aggregation or composition.

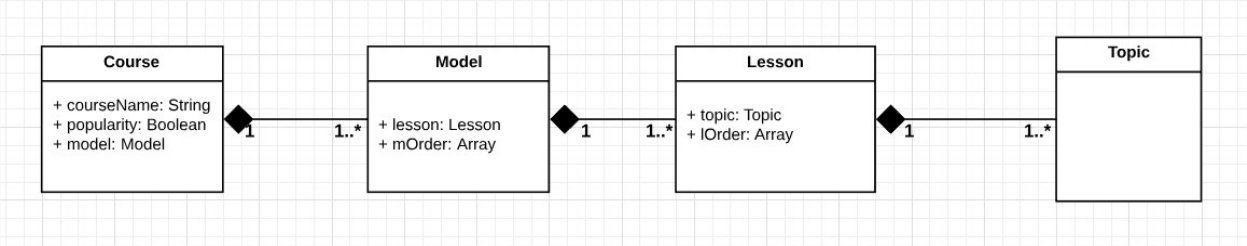


Figure 4.1

As the problem statement says that "faculty can author courses that contain learning modules broken up into lessons", I think the relationships between courses and models, and models and lessons are composition. To be specific, class Course is made up by class Model, and class Model is made up by class Lesson. And from problem statement, we can also tell that class Lesson is made up by class Topic.

Because if there is no course, there is no model. The relationship between class Course and class Model is composition. Similarly, we can infer that the relationship between class Model and class Lesson, and the relationship between class Lesson and class Topic are composition.

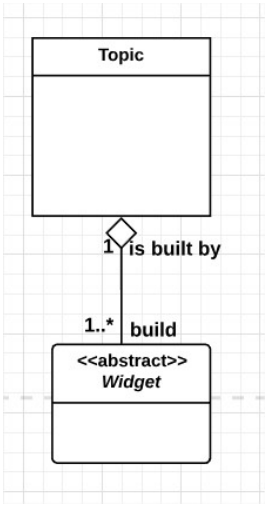


Figure 4.2

Since the topics are built by content widgets, and the lifecycles of widgets are not dependent on topics, we can say that the relationship between class Topic and superclass *Widget* is aggregation.

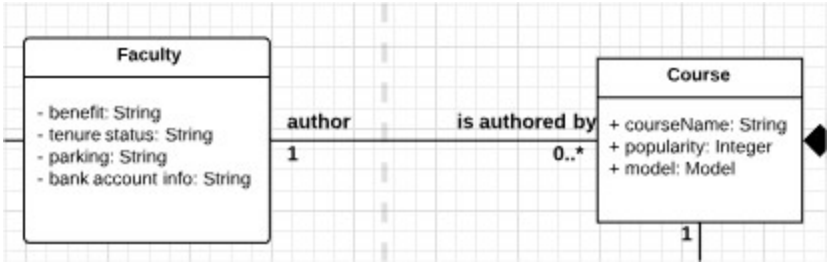


Figure 4.3

Since there are many associations, I will just show one here.

From the problem statement, we can tell the relationship between class Faculty and class Course is association. And since a faculty can author 0 course, the multiplicity is 1 and 0..*.

5. Classes vs. attributes analysis - if you are not familiar with a particular domain, e.g., a particular industry such as education, or nuclear energy, or telecommunication, or astrophysics, you might be unfamiliar with the vocabulary and so any noun might be a potential class. You might create a 'naive' class diagram that makes no distinction between classes and attributes. As you research the domain and learn more about it you realize the relationships between nouns where some might just be describing other nouns or some are compositions of other nouns. Create a naive class diagram and diagram your process of identifying certain classes as actually being attributes of other classes. Or vice versa, where some attributes might actually be better modeled as classes. Document how the class diagram evolved from a naive first approximation to the final result. Explain your decisions and support them with relevant portions of the class diagram. Show your final class diagram as one single diagram.

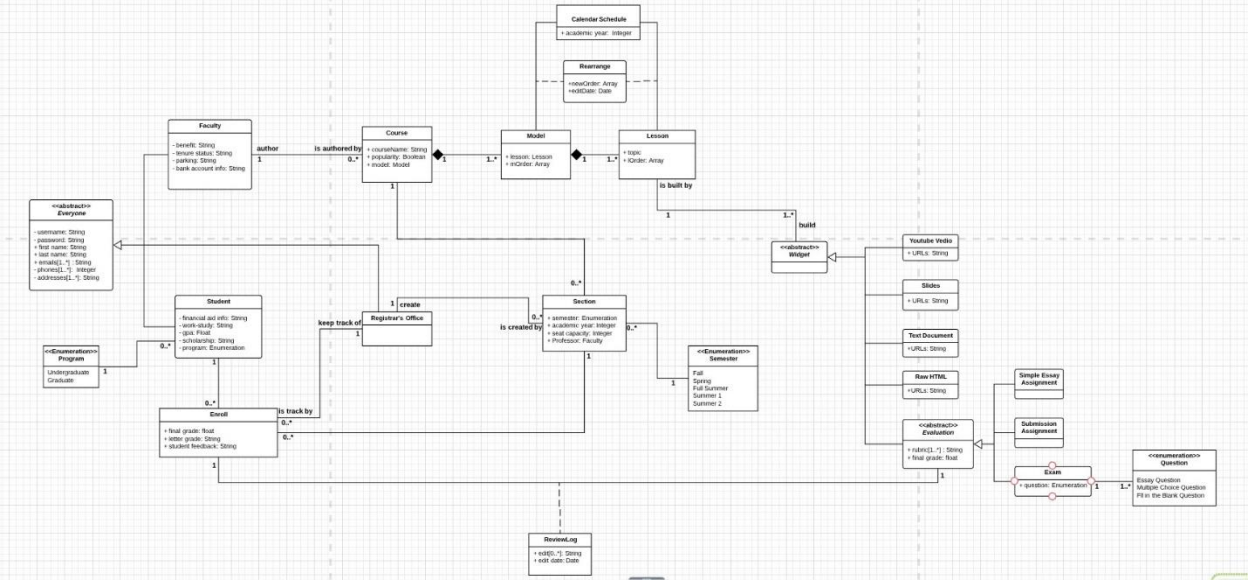


Figure 5.1

The URL of my naive class diagram:

https://www.lucidchart.com/invitations/accept/3eeb5372-aaaf-4ab5-bac6-171b85ca6639

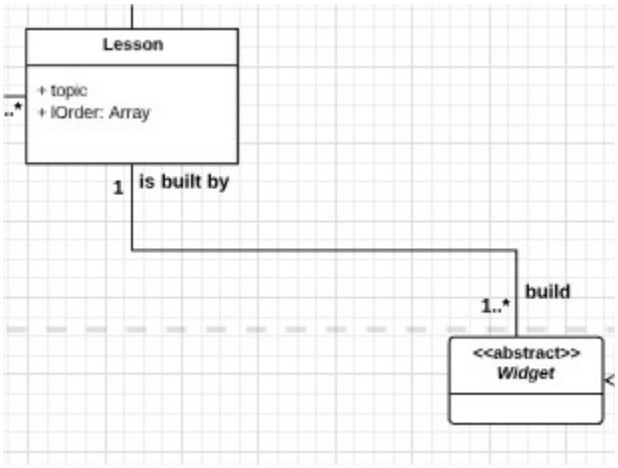


Figure 5.2

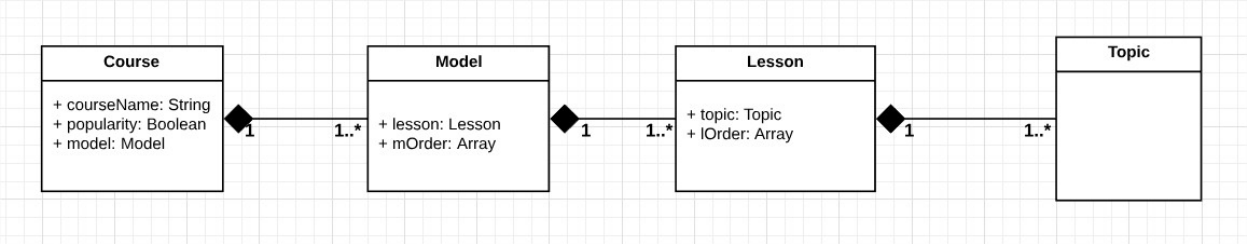


Figure 5.2

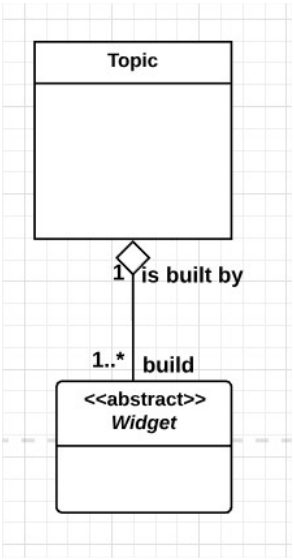


Figure 5.3

At first, I considered topic as an attribute of class Lesson, but then I realized that topics are the compositions of lessons by reading the problem statement again. And there are many kinds of topics since there are many kinds of widgets. And it's hard to represent the data type of topic if topic is just an attribute. So, I decided to model topic as a class.

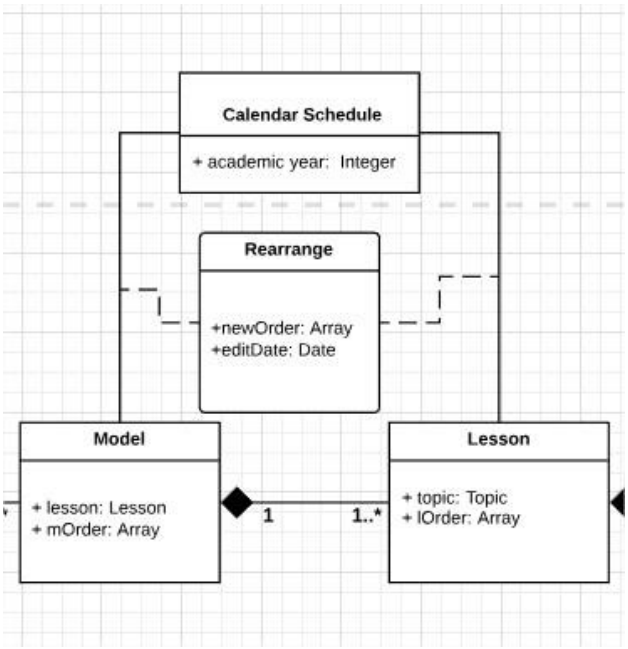


Figure 5.4

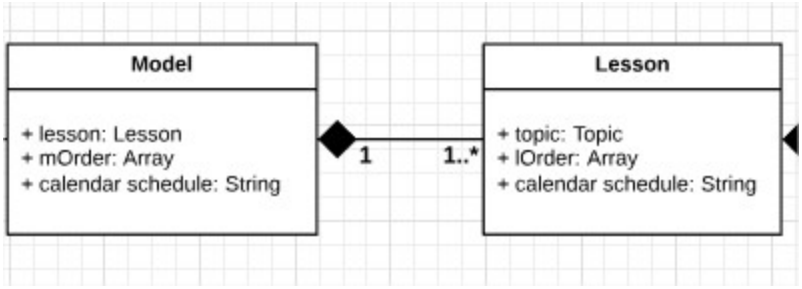


Figure 5.5

I also delete the class Calendar Schedule and class Rearrange. Because even though the order of models and lessons can be rearranged based on calendar schedule, their correspondence are not apparent. So, I considered the calendar schedule as an attribute of class Model and class Lesson. And for class Rearrange, I think it might be a method of class Model and class Lesson.

6. Correct data types, e.g., Date, String, Integer, List, Array, Enumeration, etc.

I correct the data type of attribute phones from Integer to String in the class *Everyone*. That's because there might be some symbols, like "-"and "+" appear in the field of phones.

I correct the data type of attribute professor from class Faculty to String in the class Section.

I also correct the data type of attribute popularity from Boolean to Integer in the class Course. That's because there might be more grades to represent the popularity of a course.

7. Cardinality - for every single association, show the number of instances participating in a relation

For a faculty, there can be 0..* course. Because a faculty can author 0 course or any amount of course. And for a course there is just 1 faculty.

For a student, there can be 0..* enrollment. Because there is no limitation presented in problem statement. And a student can just enroll in 1 section for a particular course.

There are many associations, other cardinality can be seen in my final class diagram.

8. Remove any inadequate or redundant relationships, entities or attributes (if applicable, explain) - if you identify redundant associations, entities, or attributes, explain how/why you removed it. For instance, the problem statement might have irrelevant information that you might need to ignore. Also, the text might describe contradictory or ambiguous descriptions. Finally, the statement might use different terms to refer to the same thing.

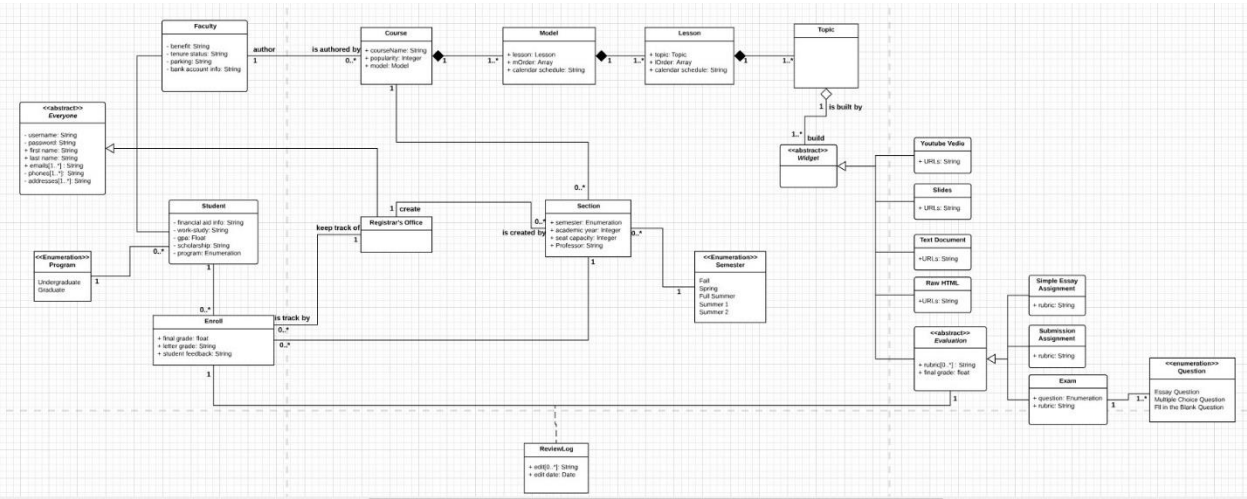
Make sure you make a compelling argument for your decisions to ignore a particular noun or verb as irrelevant or redundant or an overloaded term.

Since I didn't find any inadequate or redundant relationships, entities or attributes, I didn't remove anything.

9. Reify (if applicable, explain) - if you have association classes or other UML artifacts that don't readily map to relational schemas, explain how you transformed it to a concrete relational schema representation

I think my final class diagram can be changed to relational schemas readily.

10. Prose - in proper English, for each of the above, provide a couple of paragraphs that support your decisions and train of thought.



The URL of my final class diagram:

https://www.lucidchart.com/invitations/accept/1866653b-63e3-43bd-ae0e-b48ab84c31f4