Solving Problems Using Multiple Classes

So far...

We know how to:

- *Use* objects
- Write our own objects with fields and methods
- Make choices using conditionals
- Repeat processes using iteration
- Work with lots of data using arrays

All of this is to say: we can write complex programs. So, let's solve complex problems.

Next Step

Future projects (and exam questions 👀) will have us implement systems that...

- Manage lots of different data
- Coordinate interactions between different entities
- Respond to user interaction

This week, we'll get practice designing systems that let us do all of this.

Today: CRC Modeling

There are many ways to codify how different entities of a project will interact.

Common:

- UML Diagrams (detailed, brittle, hard)
- Story-driven modeling (intuitive, imprecise)
- **Class-responsibility-collaboration cards** (flexible, easy to write)

Example Problem Statement

> We're desgining a Learning Management System that will blend features of PennInTouch and Canvas. Students should be able to request enrollment in courses. When enrollment is approved, the student should be able to submit work for the course they're enrolled in and see the grades that they've received from that course.

-- Our imaginary client

Solving the Problem

In order to implement a full LMS as requested, we'll need to manage different objects that store data and interact together.

How do we design each of the classes for these objects, let alone decide which ones we'll even need?

CRCs can help us accomplish this.

The Basics of CRC Modeling

Represent each class of a system as an index card. On each card, write:

1. The **class name**
2. The **responsibilities** of that class
3. The **collaborators** that the class will need to accomplish its responsibilities.

Example: `Student` Class in a Learning Management System.

Name: Student	
Responsibilities	**Collaborators**
Register for classes Submit work See Grades Personal ID Information	Course Registrar

Classes

Classes represent the blueprints for how similar objects will behave. (We already know this!)

In the `Student` example, we might have a student Dana and a student Phillip. They'll be in different courses and get different grades; nevertheless, they're both still students.

Responsibilites

Either a piece of **information** that a class tracks OR a **task** that a class must be able to do.

In this case, a student has to:

- know their unique ID and grades (i.e. **information**)
- be able to register for classes and submit work (i.e. **tasks**)

Collaborators

A class might not have all the resources on its own to meet its responsibilties. For example, a `Student` usually can't grade themself; they need an instructor to report their grade. They'll need to check their `Course`'s gradebook to do that.

Generally, class `A` collaborates with `B` if

- `A` needs information from `B` to complete its responsibilities, or
- `A` shares information with `B` so that `B` can fulfill its responsibilites.

Procedure for CRC Modeling

While you're not satisfied:

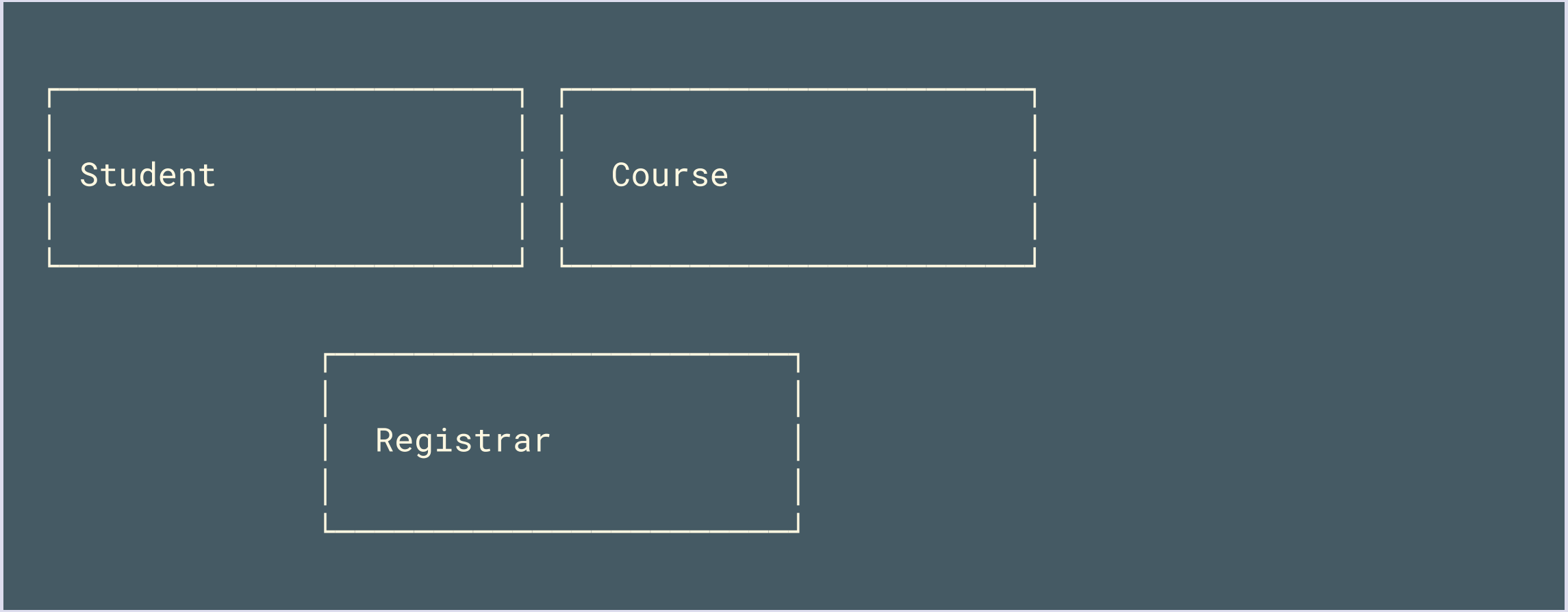
1. Identify new classes
2. Find the responsibilities of these classes
3. Define the collaborators (based on the responsibilities)
4. Arrange your CRC cards so that closely related classes are near each other.
5. Check: did the above steps reveal the need for more responsibilities or classes? If so, start from the top!

Other tips

- For the first pass, try to find three or four main classes.
- Remember that you can always come back if you think something's missing.
- It can be helpful to think of example objects from these classes: *you* are a `Student`, CIT 591 would be a `Course`.
- Collaborations are broadly symmetric
  - e.g. `Courses` have `Students`, `Students` are in `Courses`.
  - You can decide how useful it is to stick strictly to this pattern.

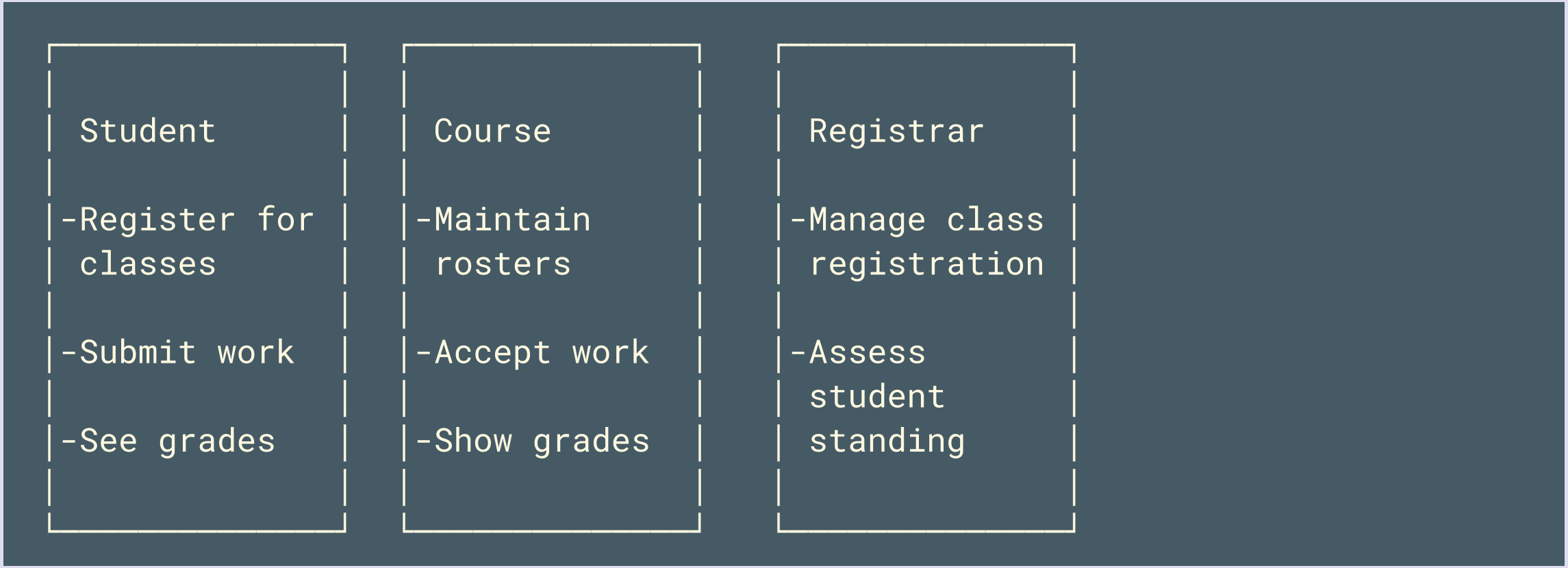
Practicing CRC: Learning Management

Step 1: Find Classes



Practicing CRC: Learning Management

Step 2: Find Responsibilities



Practicing CRC: Learning Management

Step 3: Define Collaborations

- A student can't register for class without registrar approval.
- A student's work must be submitted to a course, and the course stores the grade.
- A registrar needs to see which courses the student is registered for to assess their standing.

Practicing CRC: Learning Management

Step 3: Define Collaborations
So,

Class	Collaborators
Student	Course, Registrar
Course	Registrar, Student
Registrar	Course, Student

Practicing CRC: Learning Management

Step 4 & 5: Arranging and Assessing

We have three classes, each of which collaborate extensively with each other. Maybe with some extra detail, we can tease out this mess a little better.

What else might we add?

Practicing CRC: Learning Management

Step 4 & 5: Arranging and Assessing

We have three classes, each of which collaborate extensively with each other. Maybe with some extra detail, we can tease out this mess a little better.

What else might we add?
Gradebook, Roster, Schedule

Practicing CRC: Learning Management

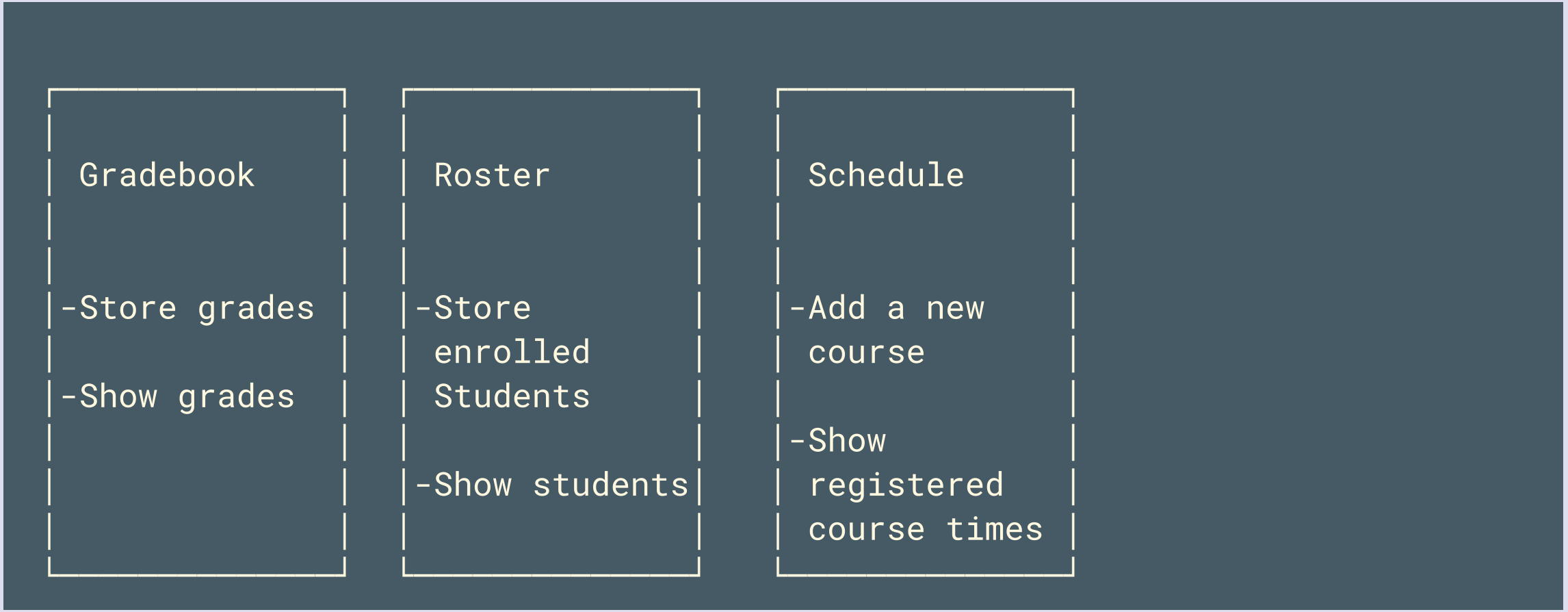
Step 1: Finding Classes (again)

Maybe we should create entities for the storage of different pieces of data to better specify interactions!

For example: `Gradebook`, `Roster`, `Schedule`

Practicing CRC: Learning Management

Step 2: Finding Responsibilities (again)



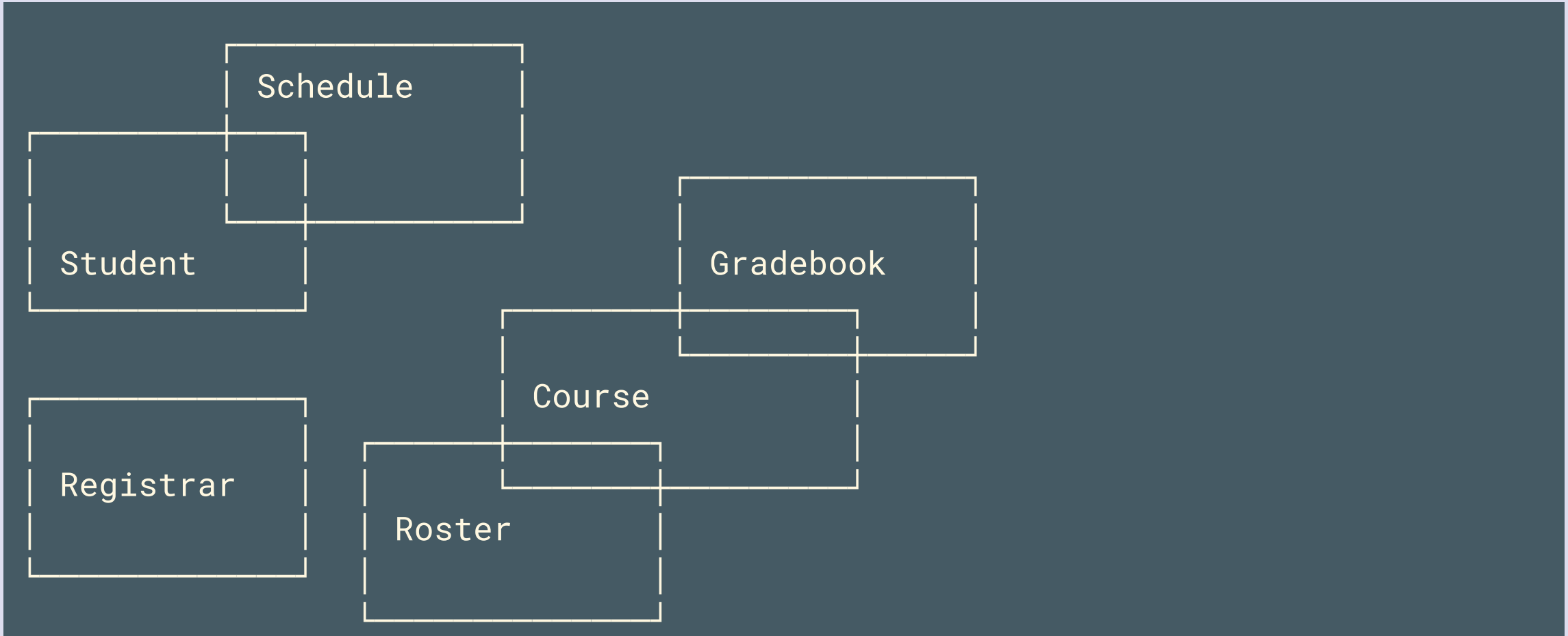
Practicing CRC: Learning Management

Step 3: **Re**define Collaborations

Class	Collaborators
Student	Gradebook, Roster, Schedule, Registrar
Course	Registrar, Roster, Gradebook
Registrar	Roster, Student, Schedule, Gradebook

Practicing CRC: Learning Management

Step 4: Asssessing our new creations



Last Thoughts

- Different people might come up with different designs!
- There is not always a single best answer, although some will be easier to program than others.
- Generally better if you can think of the CRC Model as a set of separate-but-related clusters
- Practice makes perfect!
- Remember that you can always iterate again if you're not satisfied.

Let's Practice

> You're writing software for an online book retailer like Bookshop. The site should allow customers to search an online inventory for a book they're interested in. The search should return a list of books from a database that match the search parameters. The user can then select whichever book they'd like to purchase. The book page should show local bookstores where the book is available first, and then if no copies are found locally, the book's page should show delivery options from a central warehouse. The user should be able to purchase the book and pick it up locally or get it shipped, depending on availability.

Google Doc Link

Here's where we'll put our work together.