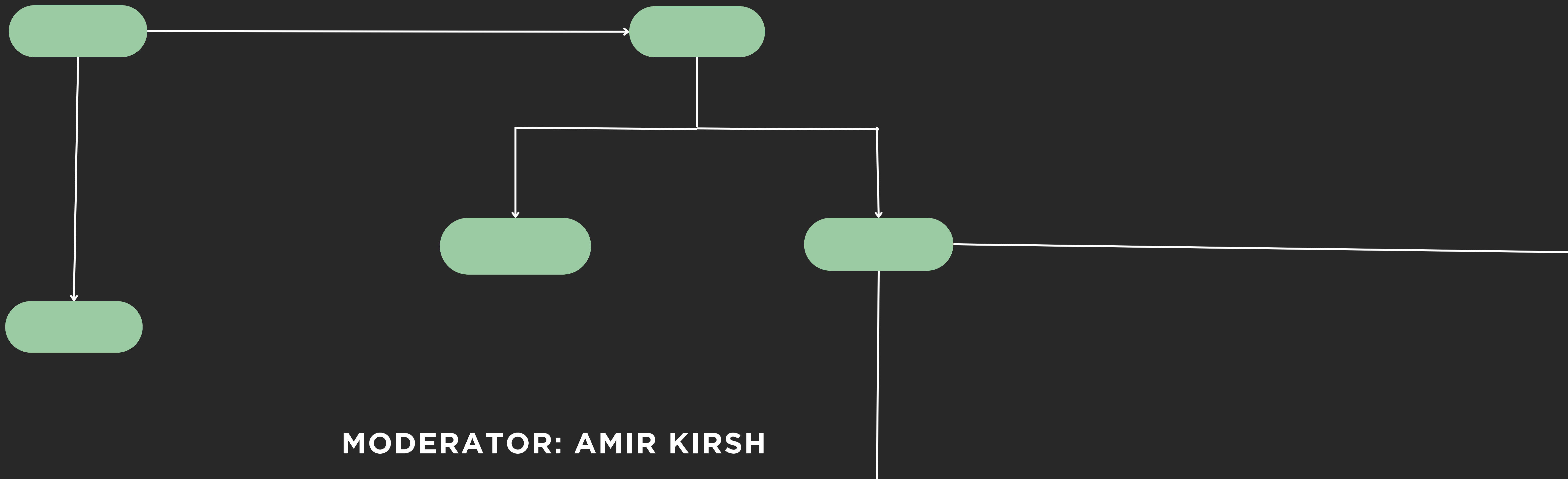
PROJECT N.15003414

CODE ARCHITECT

Yoav Sraya

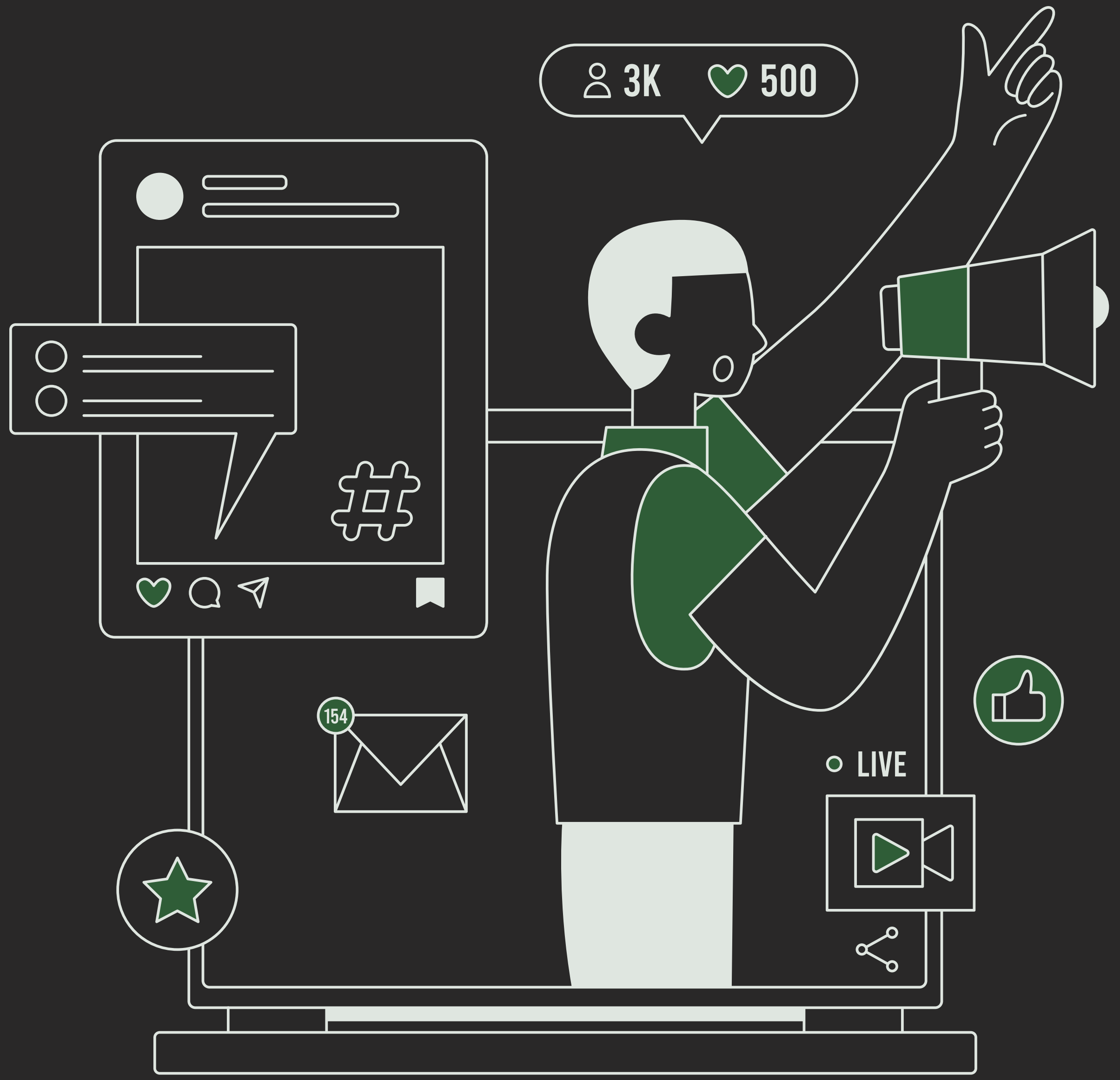
Design your OOP

Yonatan Brooker



MODERATOR: AMIR KIRSH

INTRODUCTION



Every OOP programmer knows how hard it is sometimes to keep track of your OOP world

To take into consideration: Design-Patterns, encapsulation, maintainability, efficiency, and a lot more...

NOT an easy task

WHAT ARE WE OFER

A live link to GitHub projects

No need to copy tens of files.
No need to forget to update changes
No need to create an account.
Just log in with GitHub.

An adjusted AI assistant

programmed to identify and look for OOP improvements, design pattern, and more...

A smart and interactive UML grpah

To give you a bright and visual structure for your OOP project.

HOW THIS IS WORKES

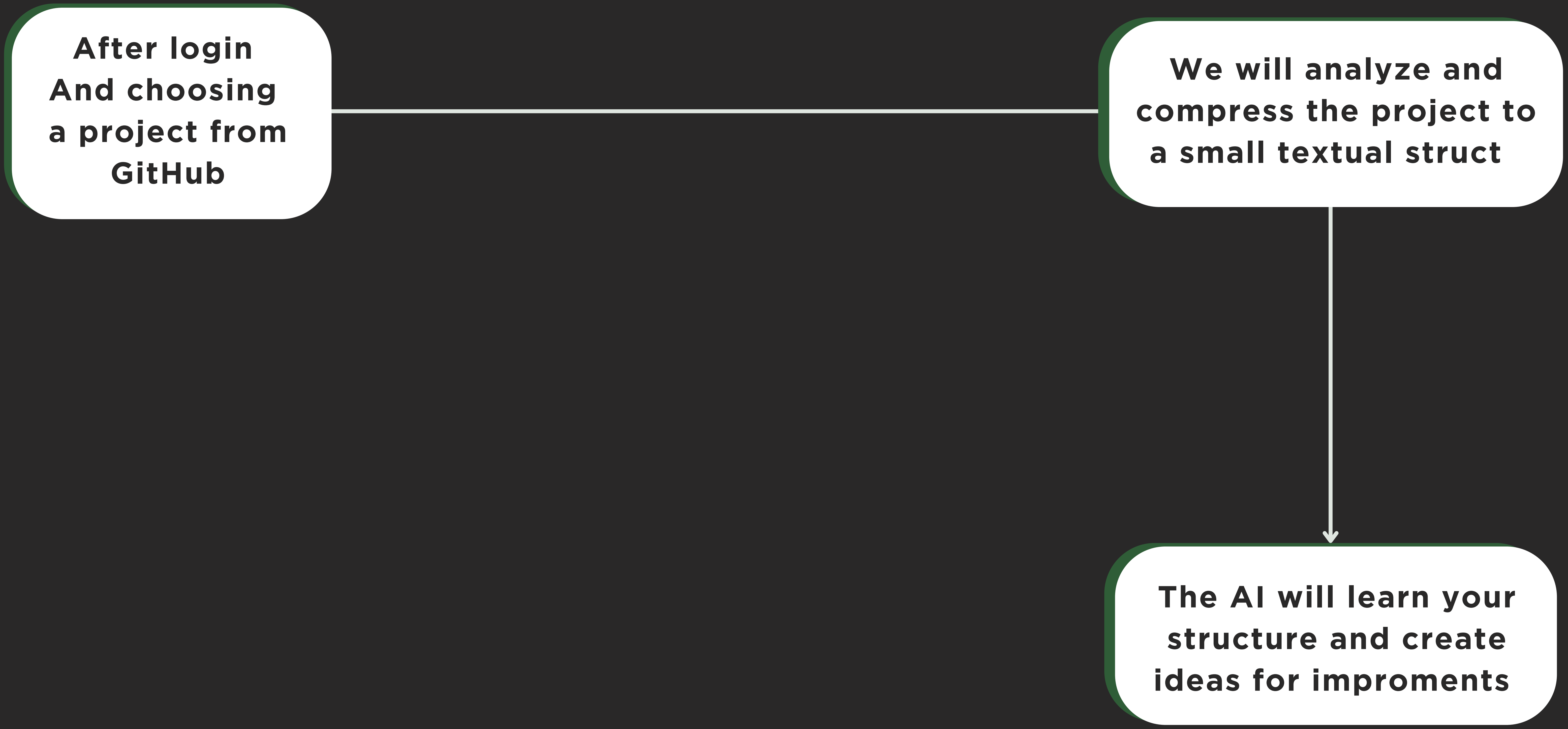
After login
And choosing
a project from
GitHub

HOW THIS IS WORKES

After login
And choosing
a project from
GitHub

We will analyze and
compress the project to
a small textual struct

HOW THIS IS WORKES



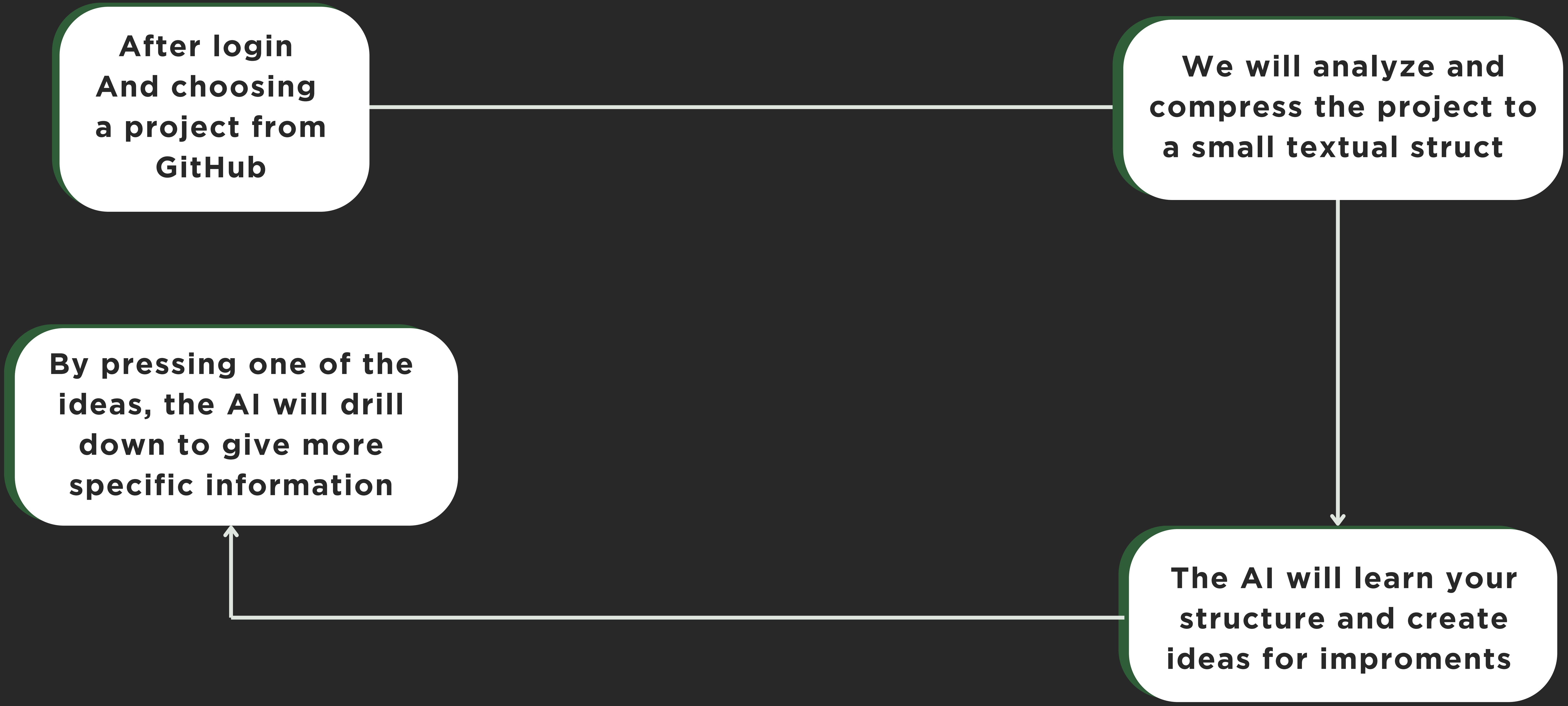
After login
And choosing
a project from
GitHub

We will analyze and
compress the project to
a small textual struct

The AI will learn your
structure and create
ideas for improments

CODE ARCHITECT

HOW THIS IS WORKES



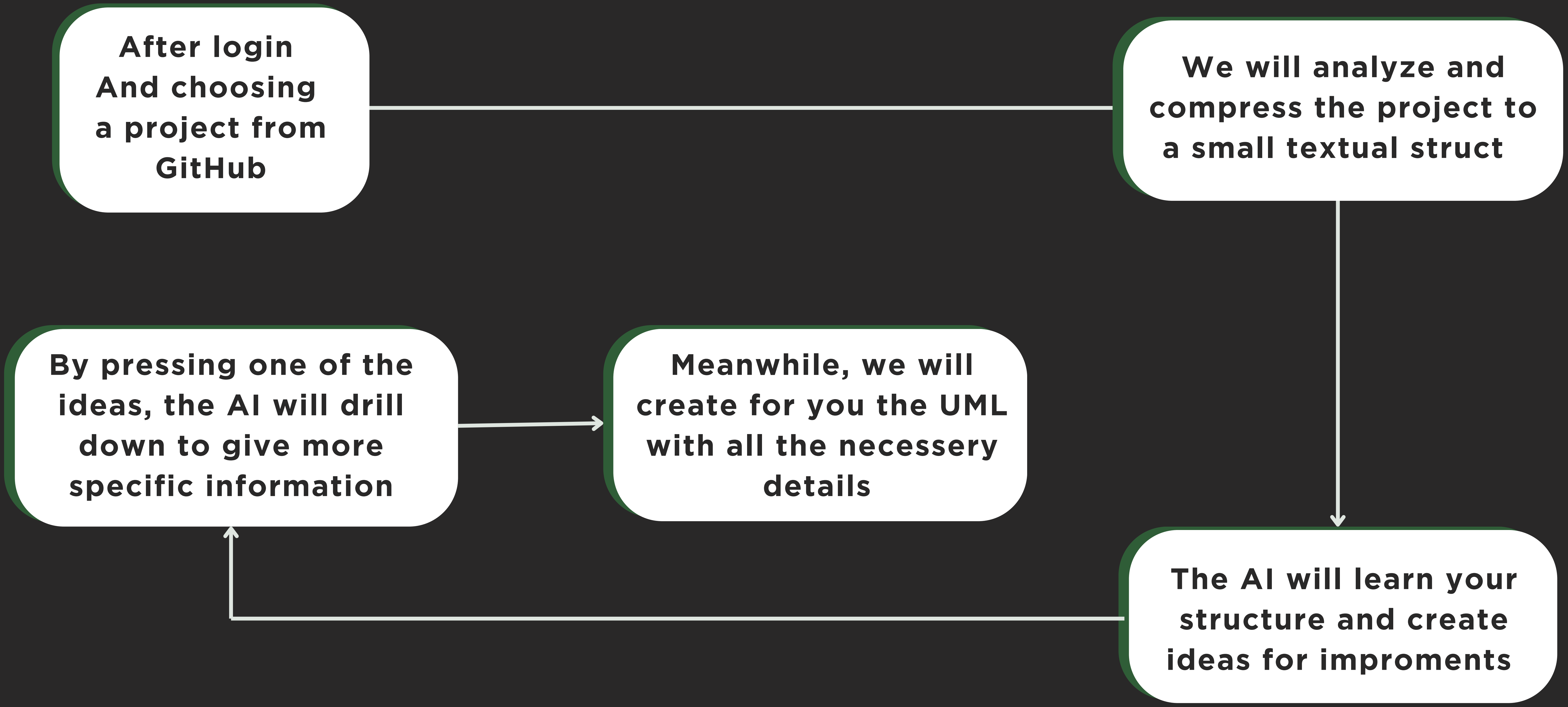
After login And choosing a project from GitHub

We will analyze and compress the project to a small textual struct

The AI will learn your structure and create ideas for improments

By pressing one of the ideas, the AI will drill down to give more specific information

HOW THIS IS WORKES



After login And choosing a project from GitHub

We will analyze and compress the project to a small textual struct

The AI will learn your structure and create ideas for improments

By pressing one of the ideas, the AI will drill down to give more specific information

Meanwhile, we will create for you the UML with all the necessery details

LET'S LOOK AT OUR ACADEMIC STRUCTURE

What are the classes we need:

STUDENT

COURSE

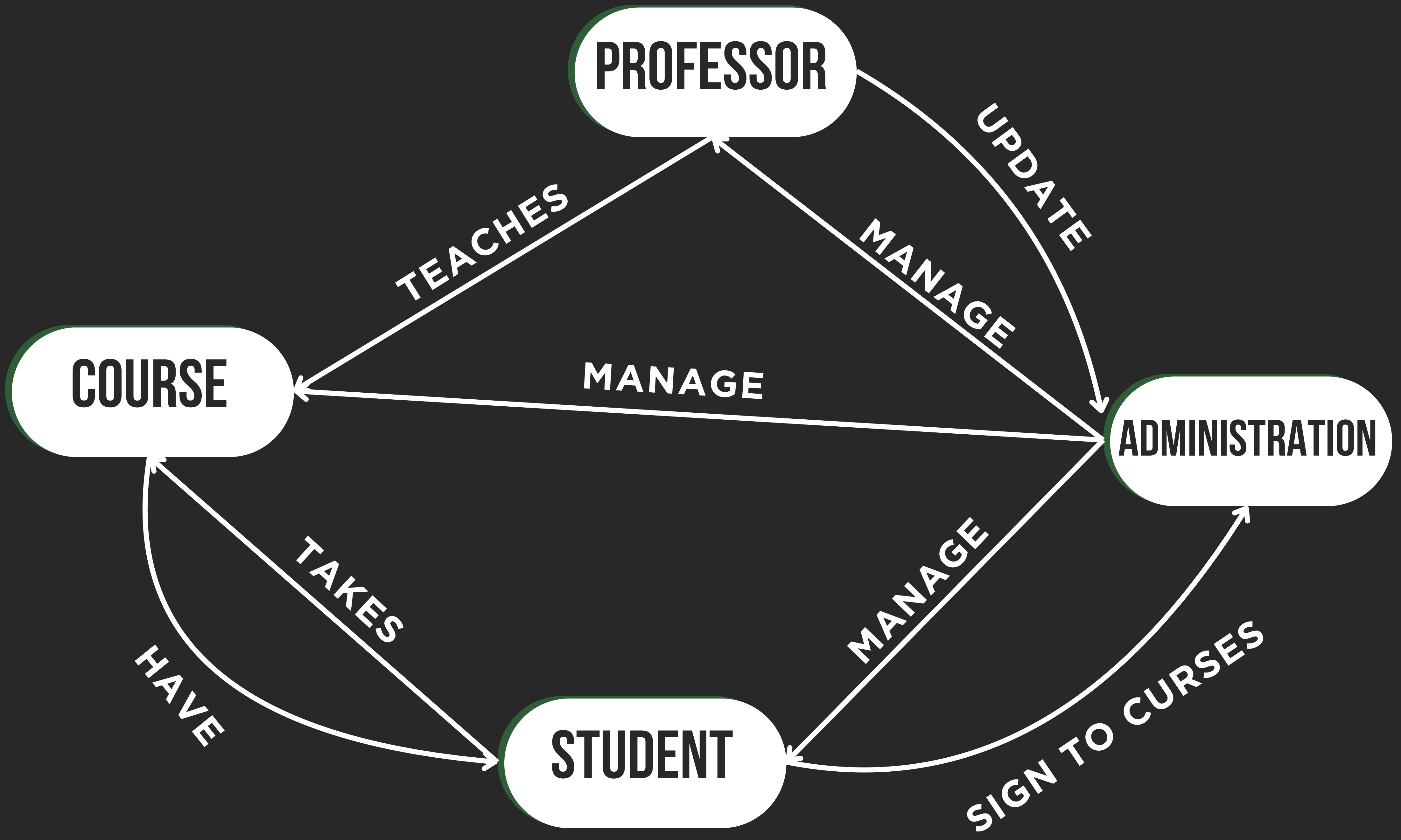
PROFESSOR

MANAMGENT

EXAMPLE

LET'S LOOK AT OUR ACADEMIC STRUCTURE

EXAMPLE



WHAT WE ARE SEE HERE:

As in UML:
vertex = class

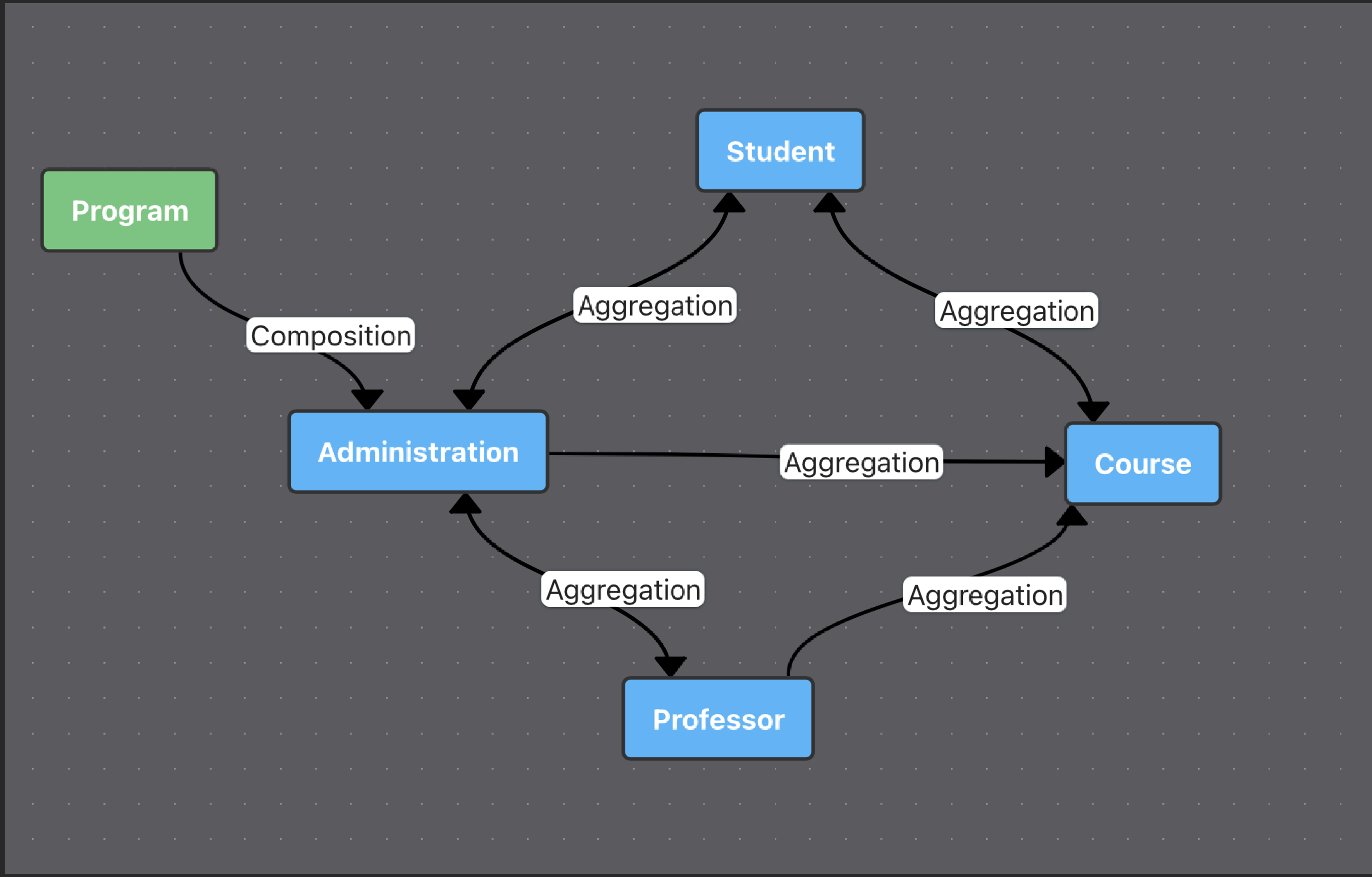
edge = connection between two class

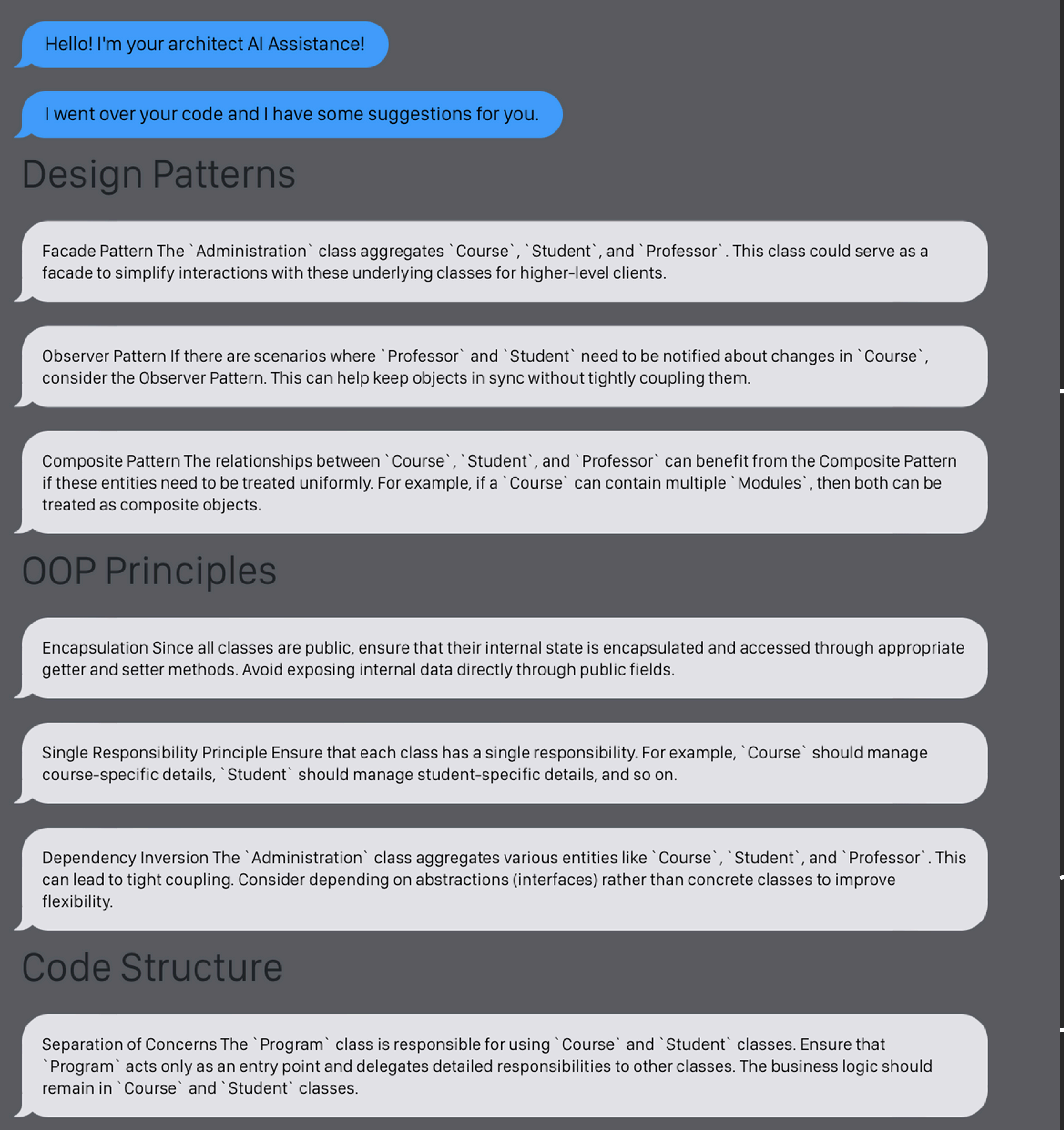
double line = circle

Each color = same File

There 4 kinds of connections:
Usage
Aggregation
Composition
Heritage

The vertices can move as the user wants to make his own unique order





Hello! I'm your architect AI Assistance!

I went over your code and I have some suggestions for you.

Design Patterns

Facade Pattern The `Administration` class aggregates `Course`, `Student`, and `Professor`. This class could serve as a facade to simplify interactions with these underlying classes for higher-level clients.

Observer Pattern If there are scenarios where `Professor` and `Student` need to be notified about changes in `Course`, consider the Observer Pattern. This can help keep objects in sync without tightly coupling them.

Composite Pattern The relationships between `Course`, `Student`, and `Professor` can benefit from the Composite Pattern if these entities need to be treated uniformly. For example, if a `Course` can contain multiple `Modules`, then both can be treated as composite objects.

OOP Principles

Encapsulation Since all classes are public, ensure that their internal state is encapsulated and accessed through appropriate getter and setter methods. Avoid exposing internal data directly through public fields.

Single Responsibility Principle Ensure that each class has a single responsibility. For example, `Course` should manage course-specific details, `Student` should manage student-specific details, and so on.

Dependency Inversion The `Administration` class aggregates various entities like `Course`, `Student`, and `Professor`. This can lead to tight coupling. Consider depending on abstractions (interfaces) rather than concrete classes to improve flexibility.

Code Structure

Separation of Concerns The `Program` class is responsible for using `Course` and `Student` classes. Ensure that `Program` acts only as an entry point and delegates detailed responsibilities to other classes. The business logic should remain in `Course` and `Student` classes.

The chat offers more the one principle of using DP correctly for example, professor and student should use observer pattern

The chat identified that there is a circle of dependency

The chat understands the simple rules of programing in c#

TECHNOLOGY

we use AWS as a continer for our web app

We use Open Ai API machine as AI chat

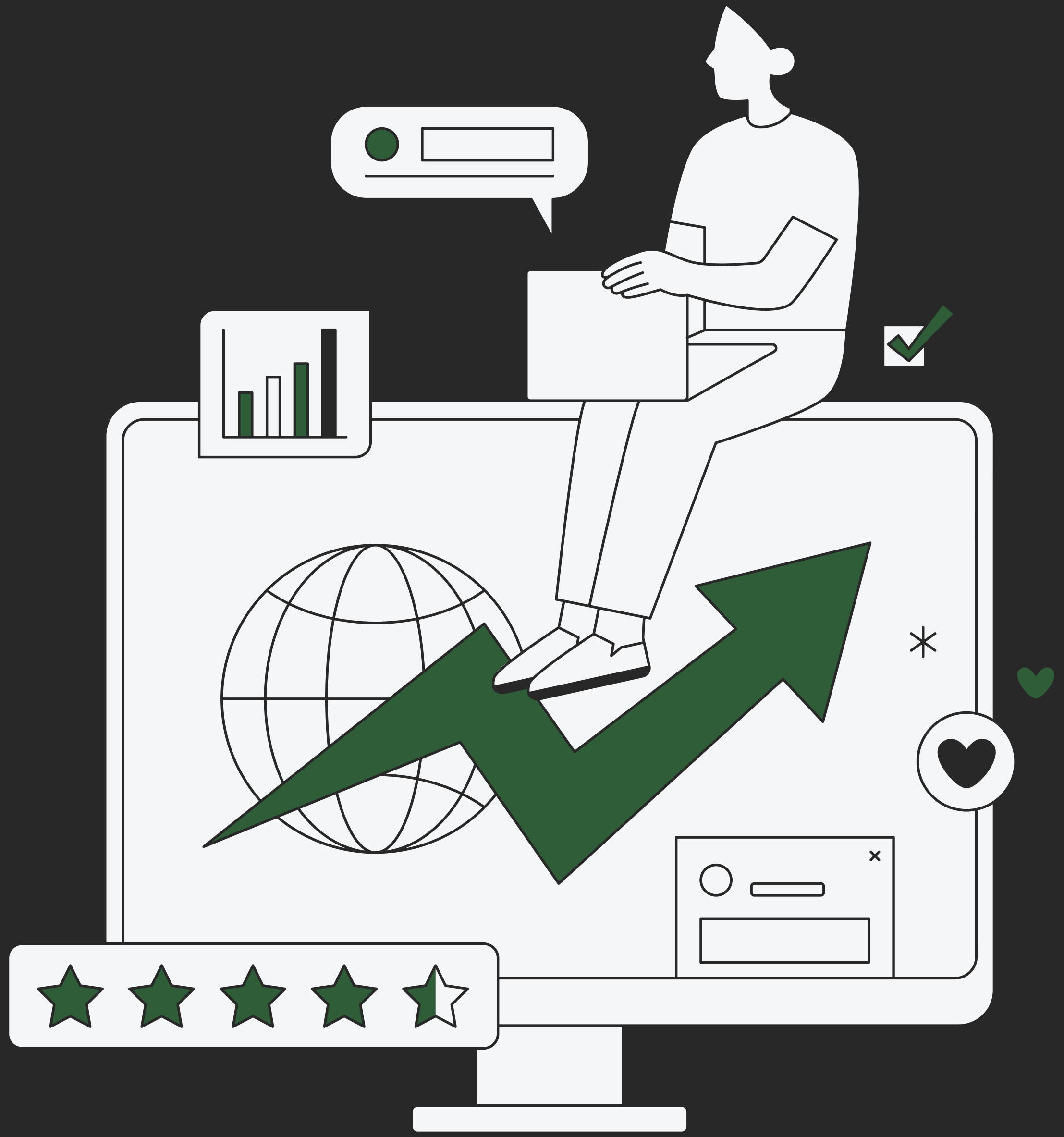
We use react as NodeJS as Server

We use GitHub API to get the user details and projects

We use React as our front-end framework

We use React-flow to create the UML and to run DFS to look for circle dependencies

THE MARKET



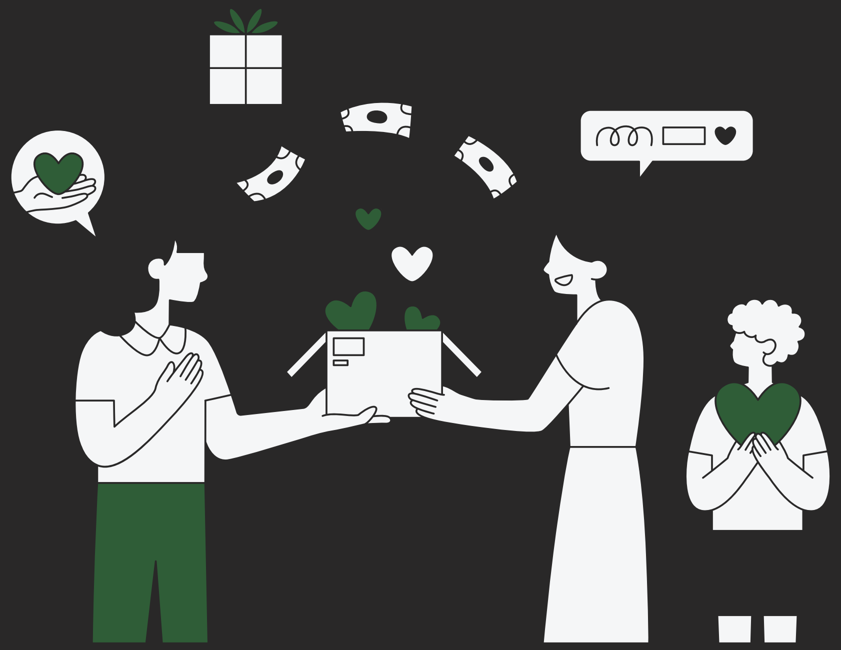
UML programs:

Our history with UML was one of the main reasons we chose this idea. looking “old”, unnecessary information, and a lot of work and time to make one

AI Chat

there are a lot of AI bots for which you can provide code and get corrections and suggestions. our AI is focused on the structure and architecture of the program.

SUMMERY



01 MAKE UML ACCESSIBLE

02 REDUCE ADAPTATION WITH EXIST PROJECT

03 INSPIRE THE PROGRAMMER WITH NEW IDEAS

04 MAKE COMPLICATED SIMPLER

OUR TEAM

YOAV SRAYA

YONATAN BROOKER

THANK YOU FOR LISTENING!