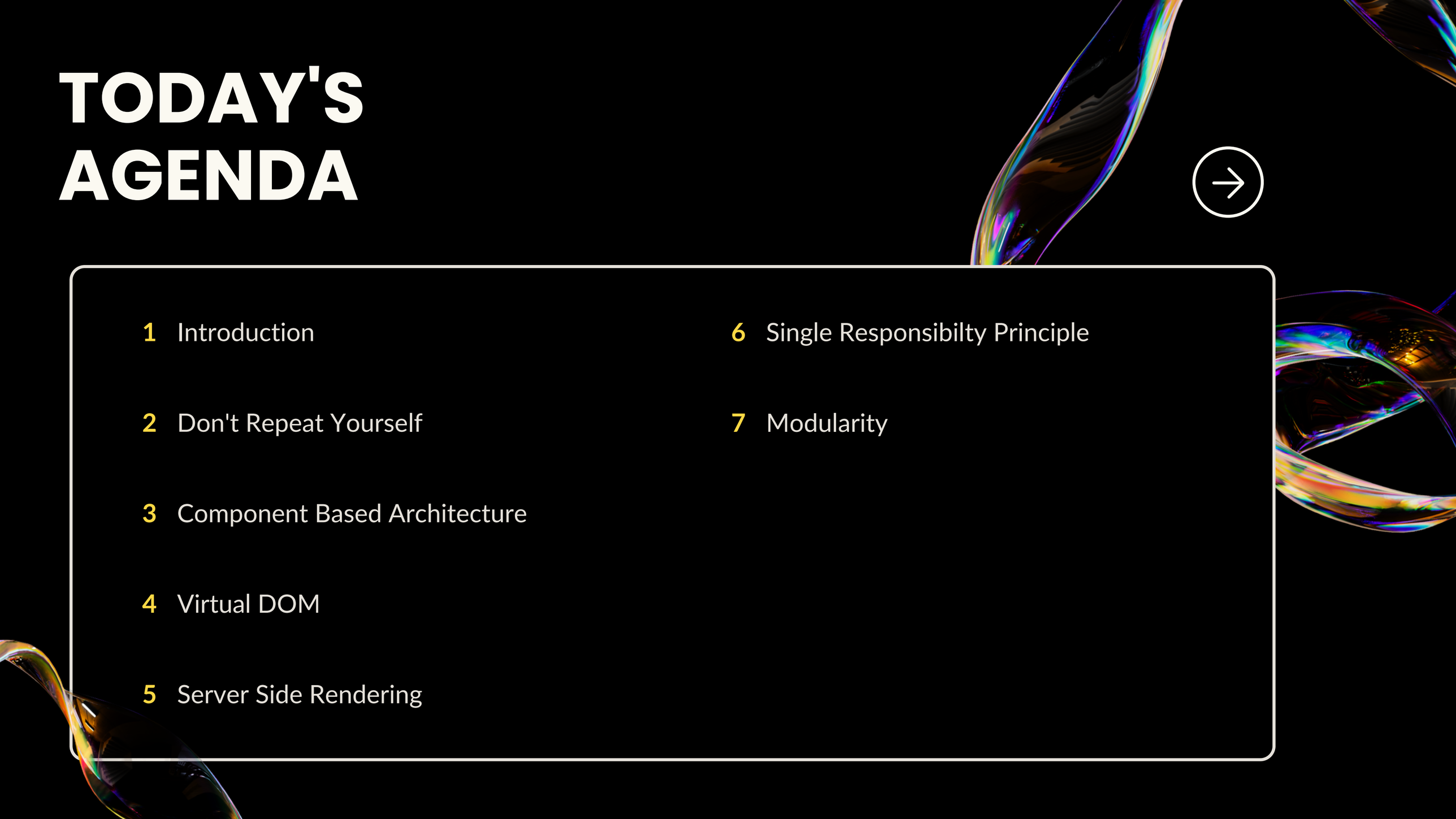
COURSE CRAFTER

Joyvin Mendonca - 9917
Vivian Ludrick - 9914
Mark Lopes - 9913

TODAY'S AGENDA



1 Introduction

2 Don't Repeat Yourself

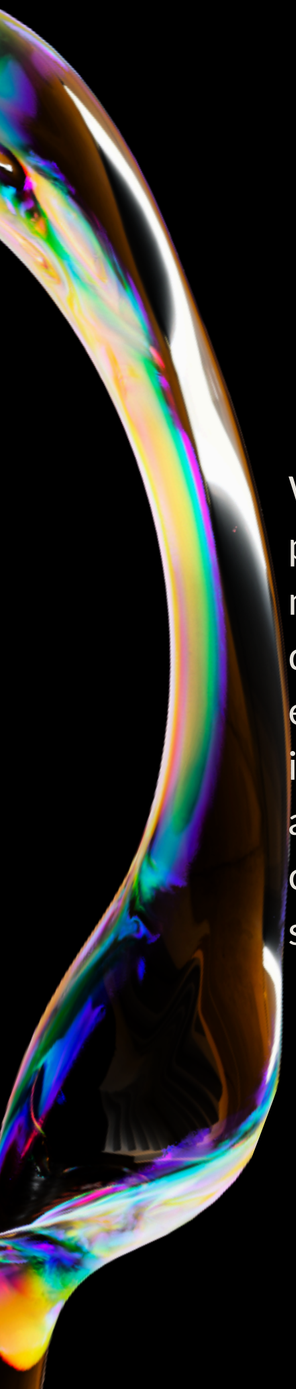
3 Component Based Architecture

4 Virtual DOM

5 Server Side Rendering

6 Single Responsibilty Principle

7 Modularity



INTRODUCTION

We are excited to present the design principles behind Course Crafter, a platform developed to streamline the process of course scheduling and management. In this presentation, we will explore the key software design principles that have shaped the development of Course Crafter, ensuring it is efficient, scalable, and easy to maintain. These principles, including modularity, reusability, and performance optimization, guide the architecture and user experience of the platform. Our goal is to provide a clear understanding of how these principles come together to create a seamless and effective solution for course scheduling.

DESIGN PRINCIPLES

DRY(Donot repeat yourself)

A programming principle that aims to minimize redundancy by avoiding code duplication, making code cleaner, easier to maintain, and less error-prone.

Component Architecture

A design approach that breaks software into reusable, self-contained components, each responsible for a specific function. This makes systems easier to develop, test, maintain, and scale.

DESIGN PRINCIPLES

Virtual DOM

Virtual DOM optimizes the process of updating the actual DOM. When a state change occurs, React only updates the parts of the DOM that have changed, making the app faster and more efficient.

Server Side Rendering

Server-Side Rendering (SSR) is better for SEO and initial load performance because it pre-renders content on the server before sending it to the client. This ensures faster content delivery, improving the user experience and making the site more discoverable by search engines.

DESIGN PRINCIPLES

Modularity

Definition: Modularity is the design principle of breaking a system into distinct, self-contained components or modules that can be easily maintained or modified.

- Flexible Updates: Each module can be developed, tested, and updated separately without impacting other parts of the system.
- Reusability: Modular components can be reused across different sections

Single Responsibility Principle

Definition: A class should have only one reason to change, meaning it should only have one job or responsibility.

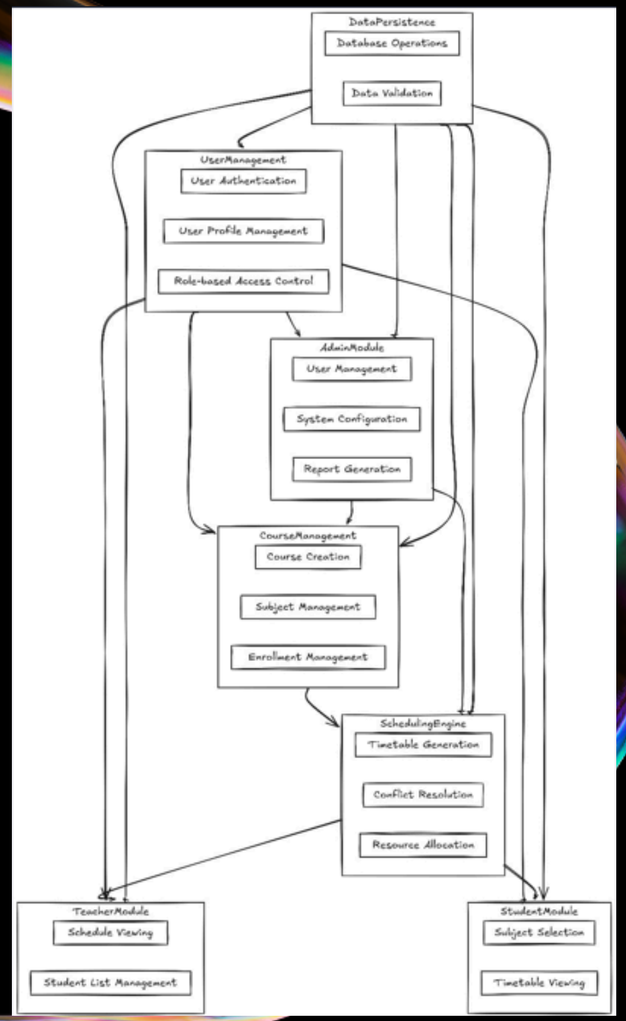
Application in Course Scheduler: Each component or class within the scheduler should focus on a single, specific function. For example ;-

- Course Algorithm
- Student Dashboard
- Teacher Dashboard

COHESION AND COUPLING

COHESION

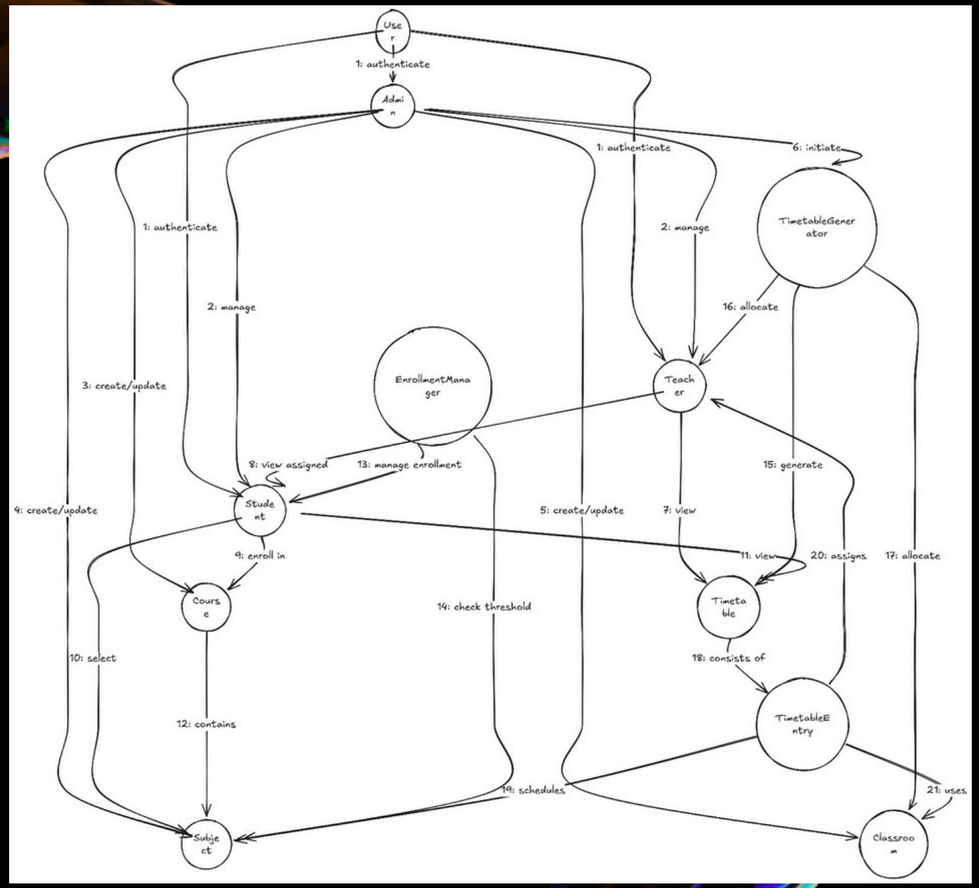
Cohesion is about how focused a module’s responsibilities are. High cohesion means a module does a specific task well, making it easier to understand, maintain, and reuse. For instance, a cohesive module might handle only user authentication.



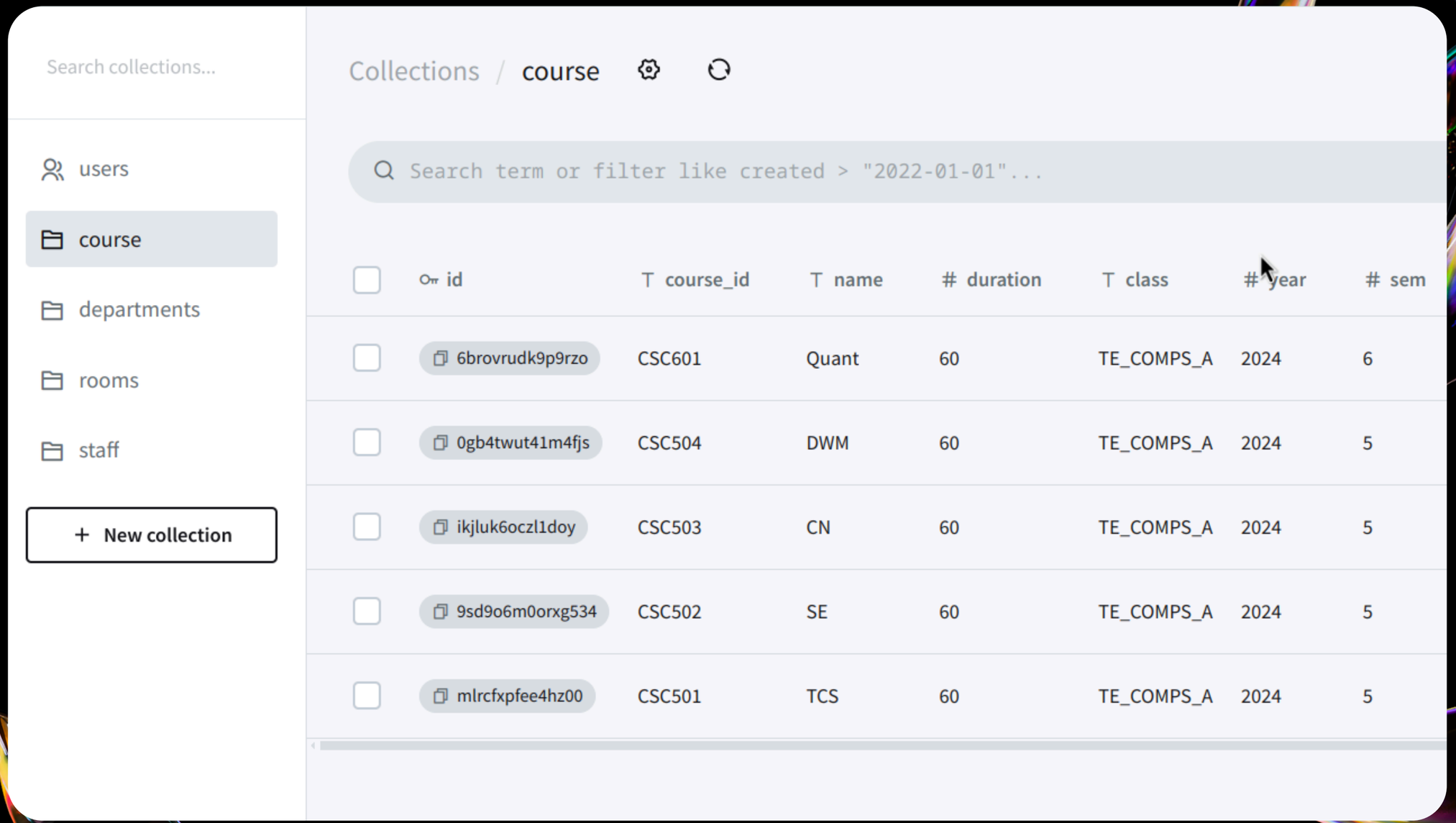
COHESION AND COUPLING

COUPLING

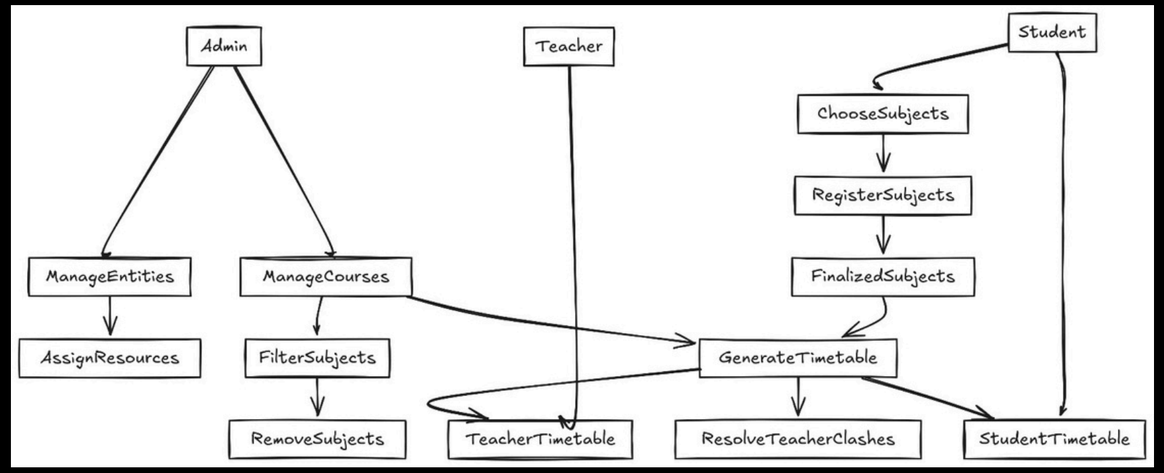
Coupling is about how much one module depends on others. Low coupling means changes in one module have little impact on others, which keeps the system flexible and easier to update. Systems aim for low coupling to avoid issues when modules heavily depend on each other.

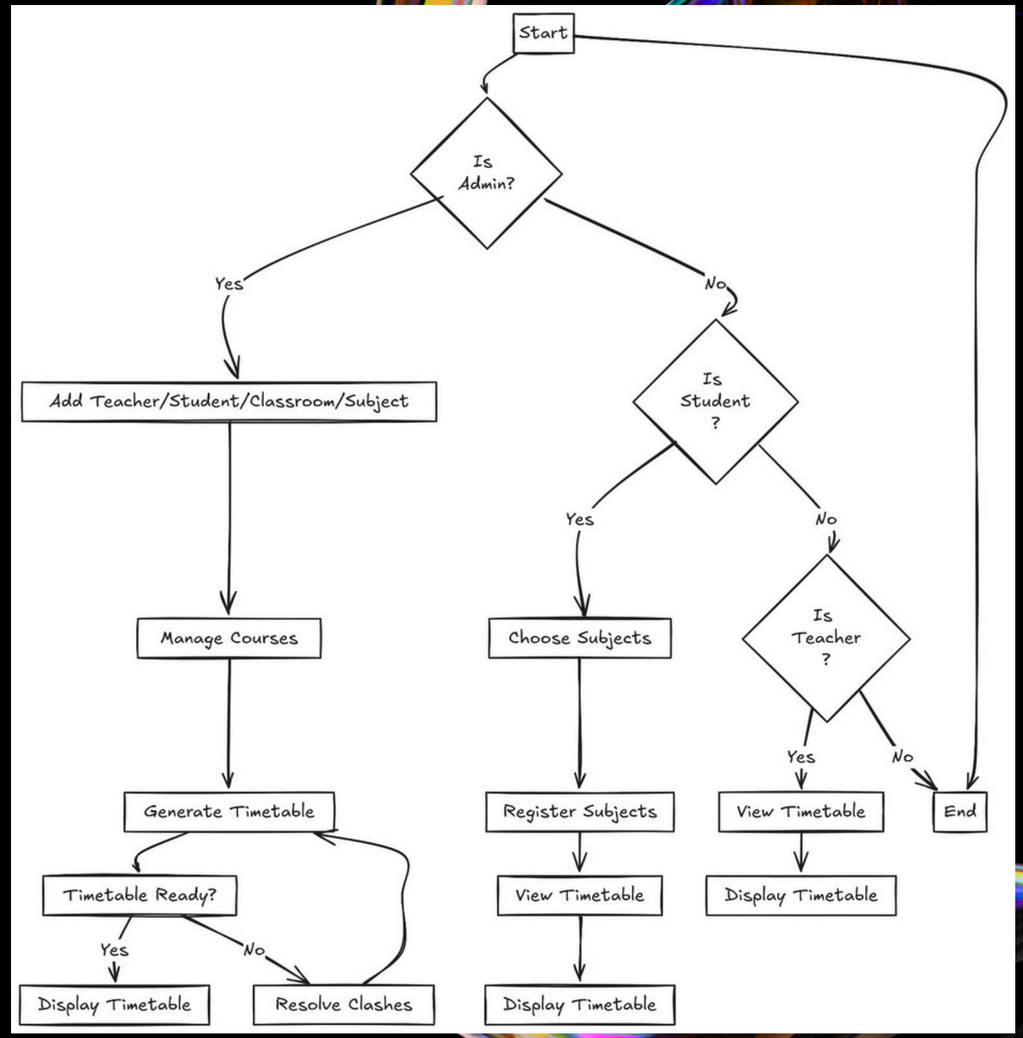


ENTITIES

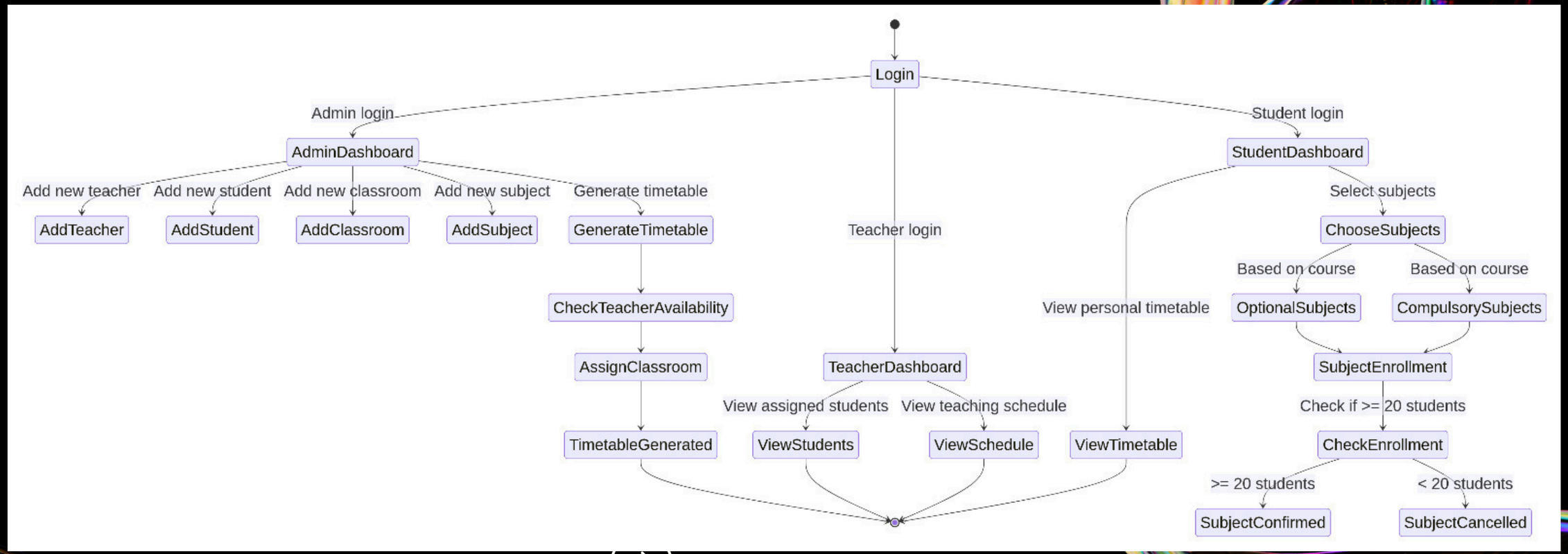

Search collections...
users
course
departments
rooms
staff
+ New collection
Collections / course
Search term or filter like created > "2022-01-01"...
id
course_id
name
duration
class
year
sem
6brovrudk9p9rzo CSC601 Quant 60 TE_COMPS_A 2024 6
0gb4twut41m4fjs CSC504 DWM 60 TE_COMPS_A 2024 5
ikjluk6oczl1doy CSC503 CN 60 TE_COMPS_A 2024 5
9sd9o6m0orxg534 CSC502 SE 60 TE_COMPS_A 2024 5
mlrcfxpfee4hz00 CSC501 TCS 60 TE_COMPS_A 2024 5

ARCHITECTURE DIAGRAMS





ARCHITECTURE DIAGRAMS



THANK YOU

for your time and attention