Week 2 – UML Design Modeling

Anthony Meza

University of Arizona Global Campus

CST 499: Capstone for Computer Software Technology

Instructor Charmelia Butler

November 4, 2024

UML Design Modeling

Testing plays a critical role in the software development life cycle, ensuring that systems function as expected and meet the required specifications. Different levels of testing are performed at various stages of development to detect defects and verify system behavior. These levels include component testing, integration testing, system testing, and acceptance testing. Each of these testing phases targets specific areas of the system, starting from individual components and progressing to the entire system and its interaction with real-world scenarios.

Component Testing

Component testing, also known as unit testing, is the foundation of the testing process. It focuses on the smallest units of the software, such as individual functions, methods, or classes. The purpose of component testing is to validate that each unit works as intended in isolation. Developers often write unit tests to verify that the code behaves as expected before it is combined with other components. For example, in a university course registration system, component testing would ensure that the logic behind functions like "enroll in course" or "view available courses" operates correctly. By isolating individual components, this type of testing allows developers to quickly identify and fix defects at an early stage (Ciampa, 2024). Catching errors early not only reduces the cost and complexity of fixing issues later in the development cycle but also improves the overall stability of the codebase. Since component tests are usually automated, they can be run frequently to ensure that any changes to the code do not introduce new issues.

Integration Testing

Integration testing comes after component testing and involves testing the interactions between multiple components to ensure they work together correctly. While component testing verifies that individual parts of the system function properly, integration testing focuses on identifying issues that arise when these components interact. In many cases, even if each component works as expected in isolation, integrating them can reveal problems such as incorrect data flow, incompatible interfaces, or unexpected behaviors. In the context of a university course registration system, integration testing would check how the user authentication module interacts with the course enrollment module and the database that stores course information. Integration testing is essential because it ensures that modules communicate effectively and that the data exchanged between them is accurate. This level of testing can also uncover issues related to system dependencies, such as database connections or third-party services. By verifying that the system's components work together seamlessly, integration testing helps ensure that the system behaves as a cohesive whole.

System Testing

System testing expands the scope further by testing the complete system in its entirety, as it would operate in a real-world environment. This level of testing aims to ensure that the system meets the functional and non-functional requirements specified in the Software Requirements Specification (SRS) (Tsui et al., 2018). System testing evaluates the entire workflow of the software, including performance, security, usability, and other critical aspects. In a university course registration system, system testing would simulate the entire user experience, from logging in and viewing courses to enrolling and managing waitlists. Additionally, system testing would check for performance bottlenecks, security vulnerabilities, and user accessibility issues, ensuring that the system operates efficiently under different scenarios. Unlike unit and integration testing, which are typically more focused, system testing provides a comprehensive view of the system’s readiness for deployment by assessing its overall behavior and robustness.

Acceptance Testing

Acceptance testing is the final stage before the software is released to the users, and it serves as a validation that the system meets business and user requirements. While system testing focuses on technical functionality, acceptance testing ensures that the system fulfills the needs of its intended users. It is typically carried out by the client or end users in an environment that mimics real-world usage. The goal is to verify that the system works as expected in practical situations and that it is ready for use. For instance, in the university course registration system, acceptance testing would involve real students registering for courses, administrators managing course capacities, and testing the system's performance during peak registration periods. Acceptance testing not only verifies that the system is functional but also that it meets the expectations of the stakeholders (Wills, 2022). If the system passes acceptance testing, it is considered ready for deployment.

In conclusion, each level of testing: component, integration, system, and acceptance testing, plays a distinct role in ensuring software quality. By starting with isolated component testing and progressing through integration, system, and acceptance testing, developers can systematically identify and resolve issues, resulting in a more stable, functional, and user-friendly system. These testing phases collectively contribute to the creation of reliable software that satisfies both technical and user requirements.

Figure 1

Use Case Diagram

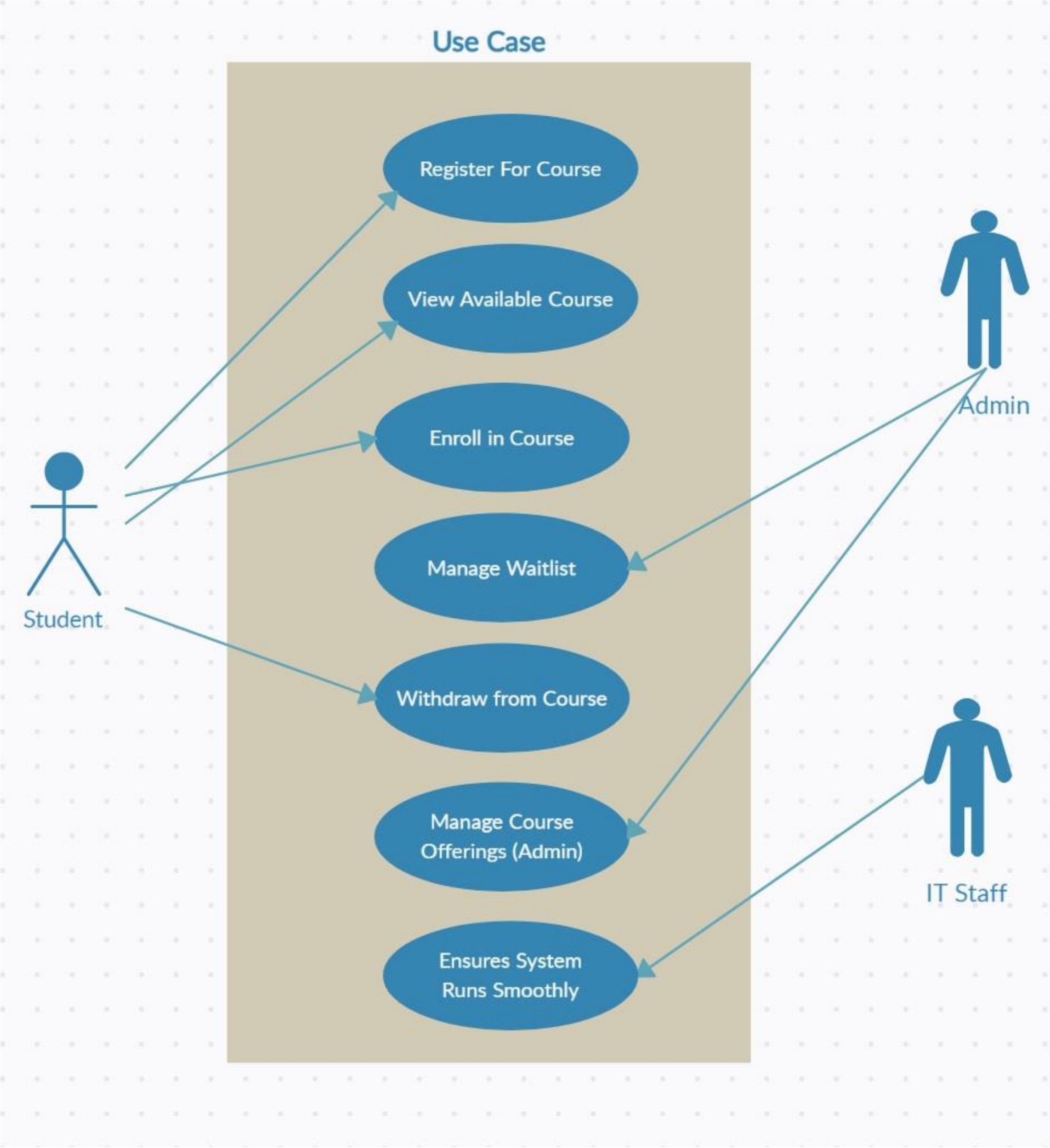


Figure 2

Class Diagram

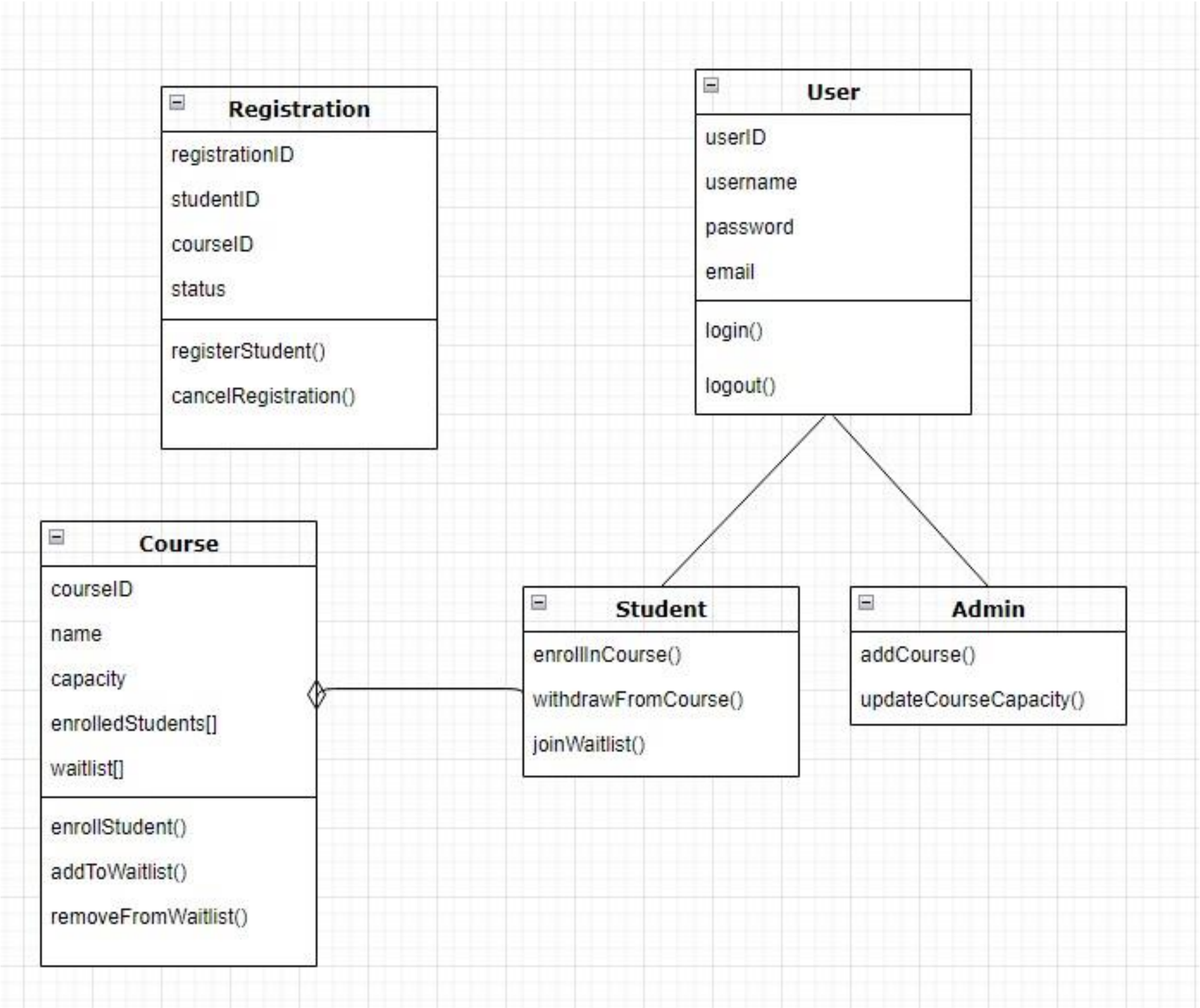


Figure 3

Sequence Diagram

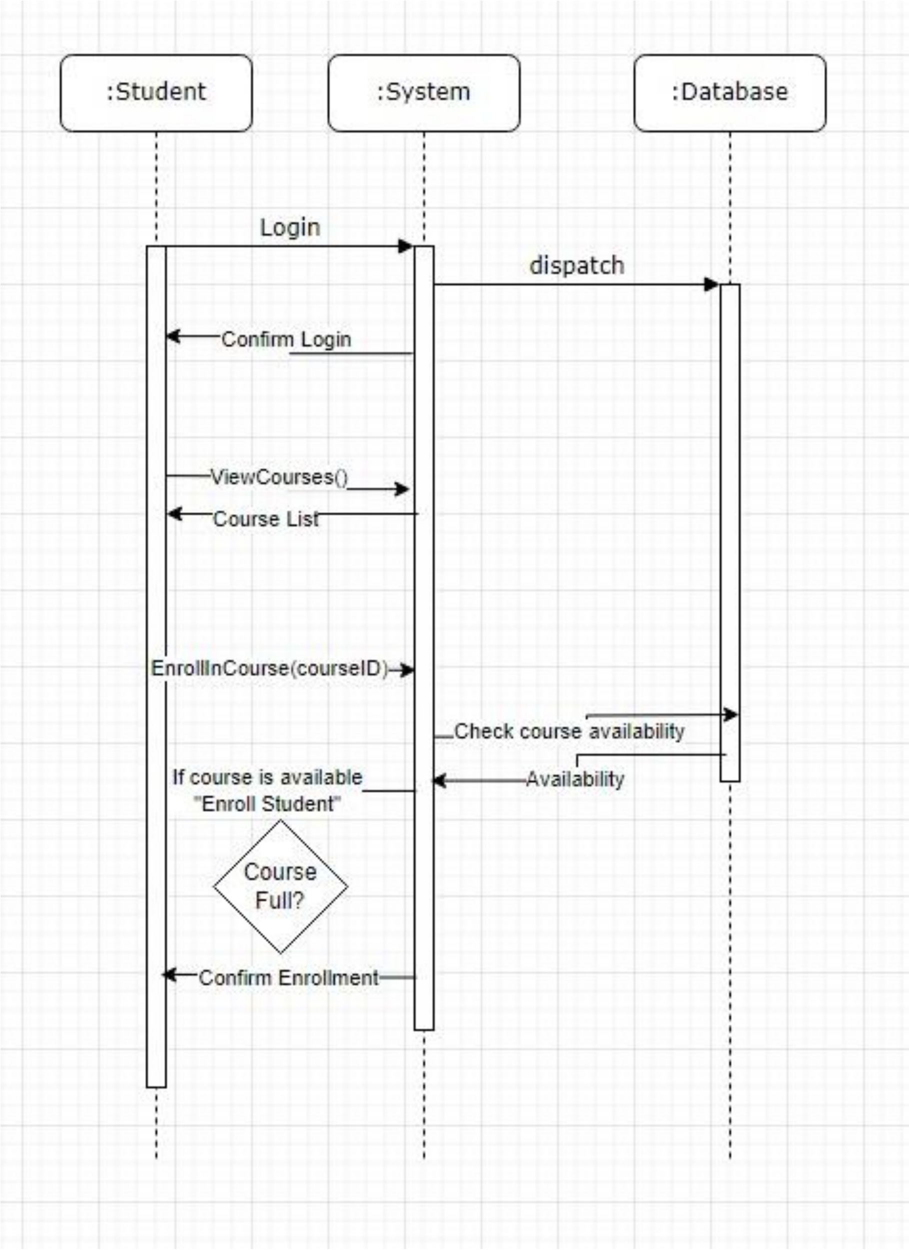
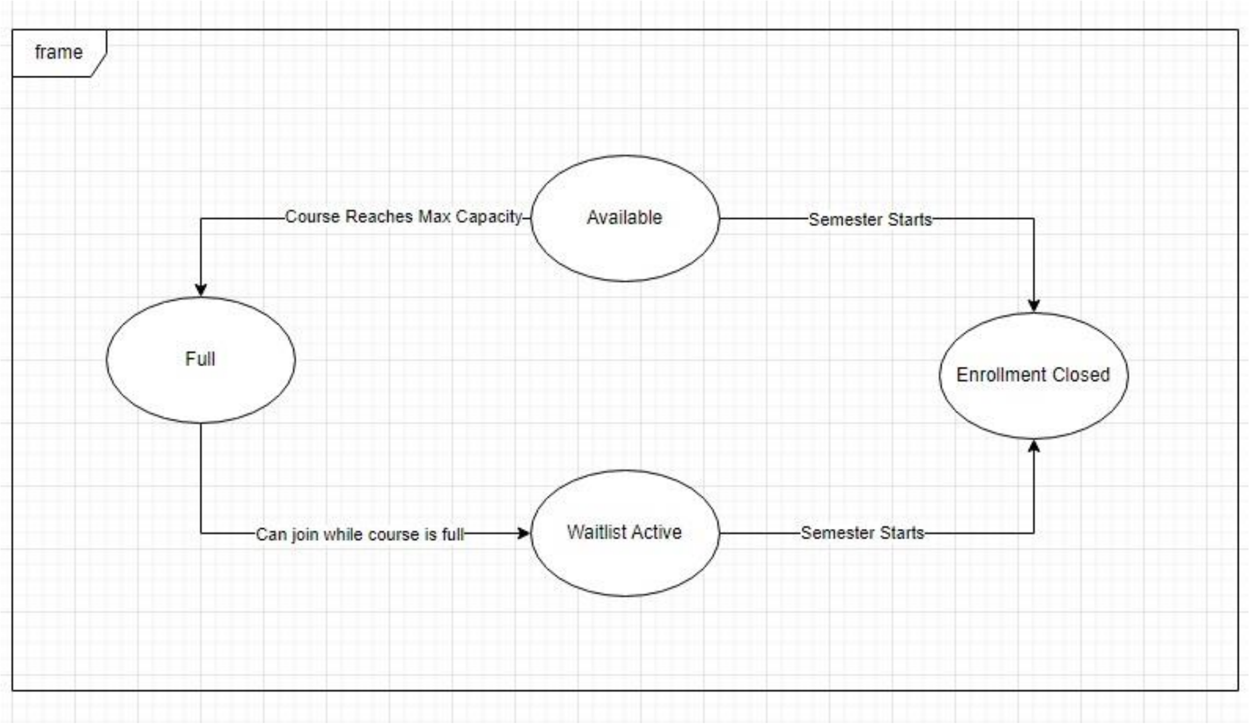


Figure 4

State Diagram


frame
Available
Course Reaches Max Capacity
Semester Starts
Full
Enrollment Closed
Can join while course is full
Waitlist Active
Semester Starts

References

Ciampa, M. (2024). *Security awareness: Applying practical security in your world* (6th ed.). Cengage Learning.

Tsui, F., Karam, O., & Bernal, B. (2018). *Essentials of software engineering* (4th ed.). Jones & Bartlett Learning.

Wills, M. (2022). *(ISC)2 SSCP systems security certified practitioner official study guide* (3rd ed.). Wiley Professional Development (P&T).