# **Day 2 – Design Document**

# **Project: Student Course Registration System**

# **INTRODUCTION**

The design phase focuses on defining the **object-oriented structure** for the Student Course Registration System. This includes identifying the core classes, their attributes, behaviors (methods), and the relationships between them. The design will adhere to **OOP principles** and will be scalable for future enhancements such as prerequisite checking, notifications, and reporting.

# **2. CORE CLASSES**

**2.1 Student**

* **Responsibilities:**
* Store student details (ID, Name, Email, Password)
* Manage enrolled and waitlisted courses
* **Attributes:**
* studentId: String
* name: String
* email: String
* password: String
* enrolledCourses: List<Enrollment>
* **Methods:**
  + viewDashboard()
  + addEnrollment(Enrollment)

**2.2 Course**

* **Responsibilities:**
* Store course details (ID, Name, Capacity, Dates)
* Manage enrolled students and waitlist
* **Attributes:**
* courseId: String
* courseName: String
* maxCapacity: int
* availableSeats: int
* startDate: LocalDate
* endDate: LocalDate
* enrolledStudents: List<Student>
* waitlist: Queue<Student>
* **Methods:**
  + - enrollStudent(Student)
    - dropStudent(Student)
    - addToWaitlist(Student)

**2.3 Enrollment**

* **Responsibilities:**
* Represent the relationship between Student and Course
* Store enrollment status (ENROLLED, WAITLISTED, DROPPED, COMPLETED)
* **Attributes:**
* student: Student
* course: Course
* status: EnrollmentStatus
* **Methods:**
* getStatus()
* getCourse()

**2.4 EnrollmentStatus (Enum)**

* ENROLLED
* WAITLISTED
* DROPPED
* COMPLETED

**UML CLASS DIAGRAM**

+-------------------------------------+

| Student |

+-------------------------------------+

| - studentId: String |

| - name: String |

| - email: String |

| - password: String |

| - enrolled Courses: List<Enrollment> |

+-------------------------------------------+

| + viewDashboard(): void |

| + addEnrollment(e: Enrollment) |

+-------------------------------------------+

1 \*

Student ------------------------- Enrollment

|

\* 1

|

Course

+-------------------------------------------------+

| Course |

+--------------------------------------------------+

| - courseId: String |

| - courseName: String |

| - maxCapacity: int |

| - availableSeats: int |

| - startDate: LocalDate |

| - endDate: LocalDate |

| - enrolledStudents: List<Student> |

| - waitlist: Queue<Student> |

+-----------------------------------------------------+

| + enrollStudent(s: Student): boolean |

| + dropStudent(s: Student): void |

+------------------------------------------------------+

+------------------------------------------------------+

| Enrollment |

+------------------------------------------------------+

| - student: Student |

| - course: Course |

| - status: EnrollmentStatus |

+------------------------------------------------------+

| + getStatus(): EnrollmentStatus |

| + getCourse(): Course |

+------------------------------------------------------+

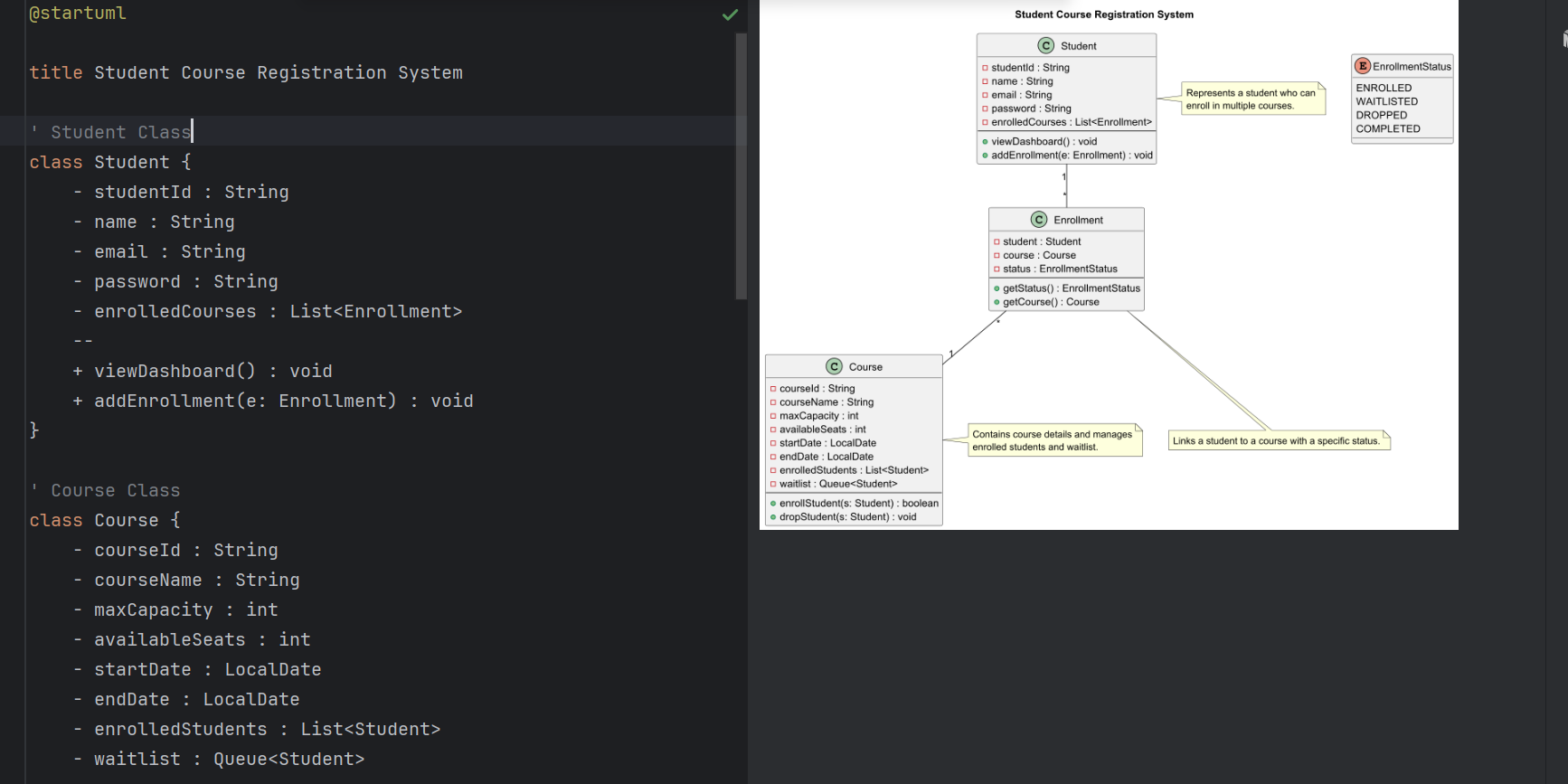
+------------------------------------------------------+

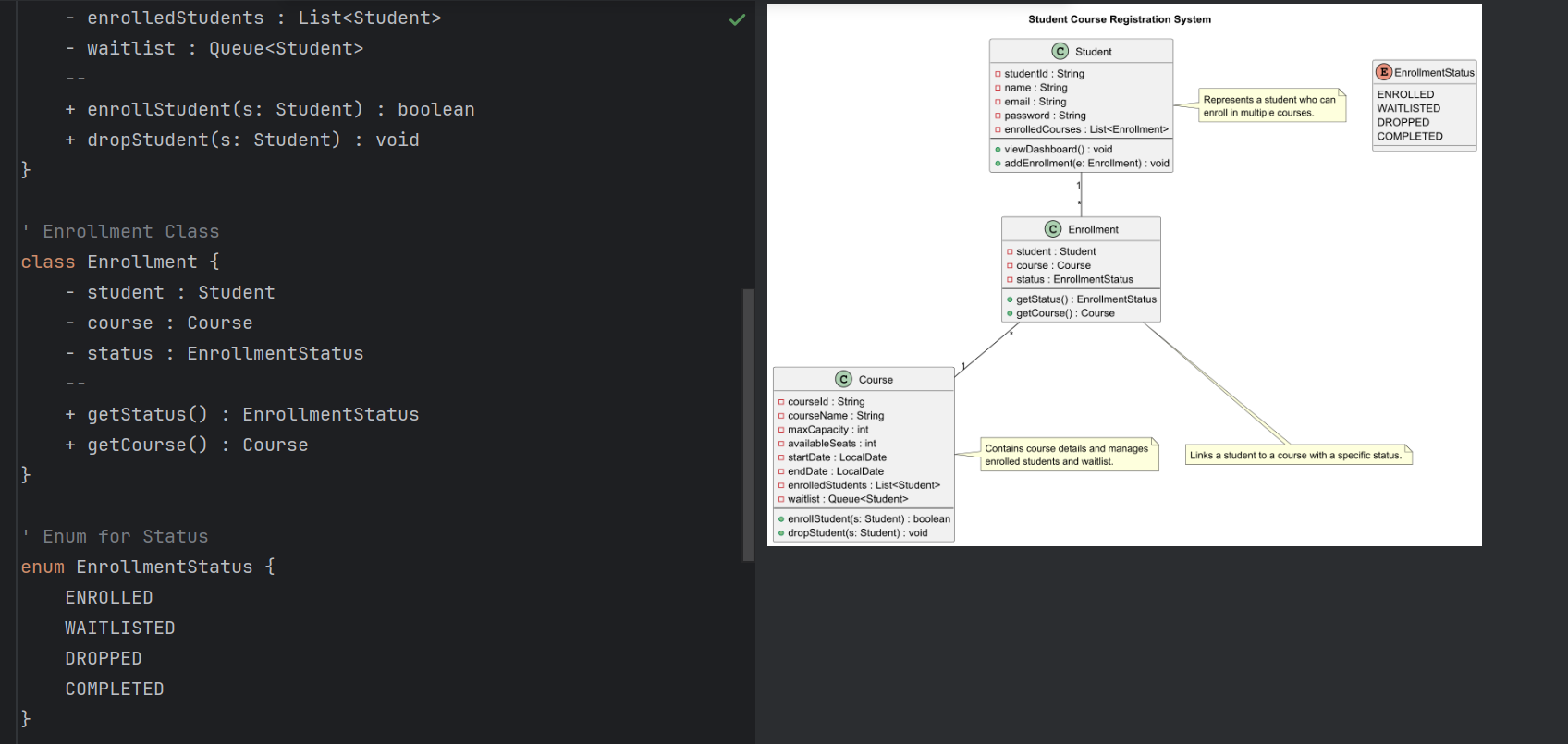
| EnrollmentStatus (enum) |

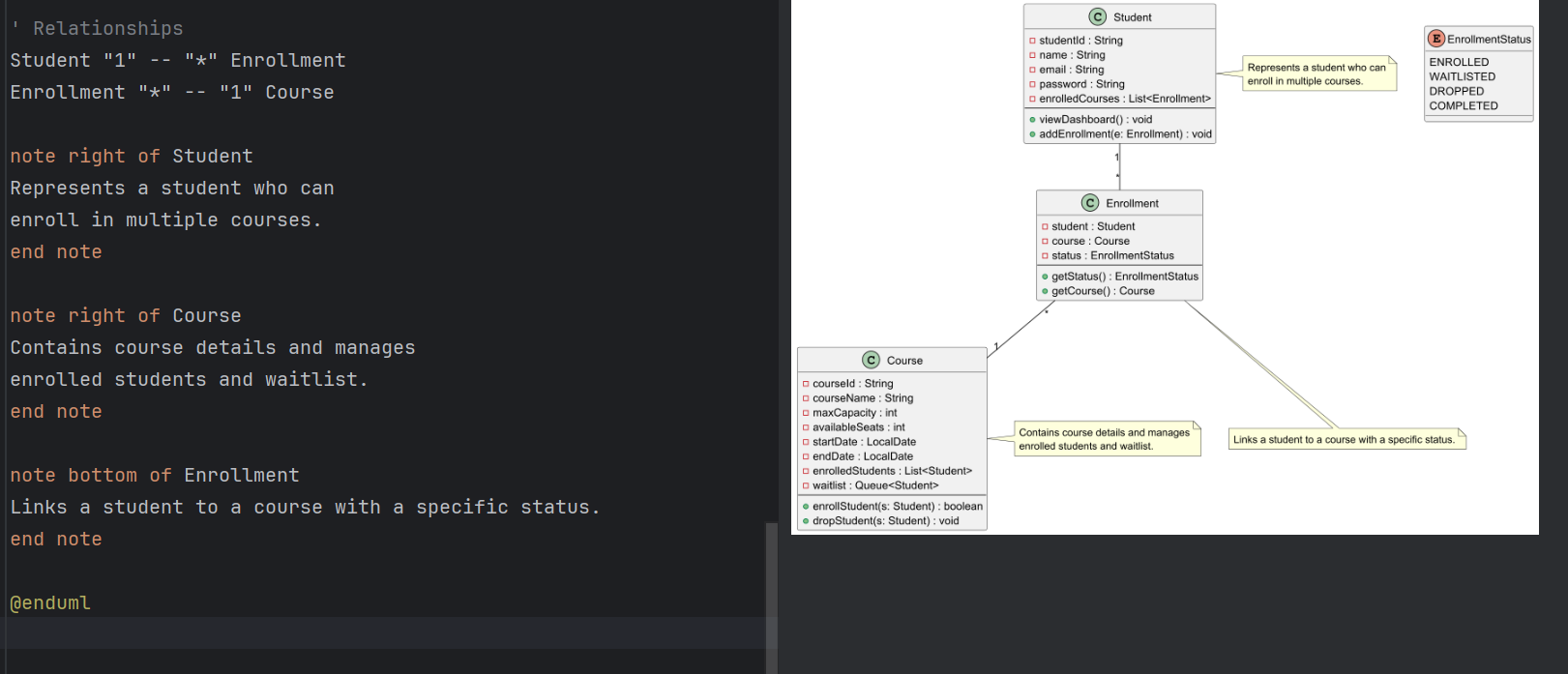
+------------------------------------------------------+

| ENROLLED, WAITLISTED, DROPPED, COMPLETED |

+-------------------------------------------------------+







**4. Relationships**

* **Student ↔ Enrollment**: One student can have multiple enrollments.
* **Course ↔ Enrollment**: One course can have multiple enrollments.
* **Student ↔ Course**: Many-to-many relationships managed through **Enrollment** class.

**5. Justification for design**

* **Scalability:** Easy to add new features like prerequisite checking or notifications.
* **Separation of Concerns:** Student, Course, and Enrollment have distinct responsibilities.
* **Extensibility:** Adding new enrollment statuses or course attributes will not affect other classes.