✅ 1) Functional Requirements (SMART)
FR1 – User Registration & Login

Specific: The system must allow students to create an account using name, email, password, and student ID.

Measurable: Registration and login should be completed within 3 steps.

Achievable: Implemented using C++ classes and file-based storage.

Relevant: Needed for secure access to the system.

Time-bound: Must be implemented before the course registration module.

FR2 – View Available Courses

Specific: The system should display all available courses with code, name, instructor, and capacity.

Measurable: Student should see 100% of courses in the file/database.

Achievable: Uses vector/list of Course objects.

Relevant: Students need to know what courses they can register for.

Time-bound: Implemented immediately after login functionality.

FR3 – Course Registration

Specific: Students must be able to register for any available course if capacity allows.

Measurable: A student may register for up to 6 courses per semester.

Achievable: Uses enrollment list linked to Student & Course classes.

Relevant: Core purpose of the system.

Time-bound: Should be completed within the main registration module.

FR4 – Course Dropping

Specific: Students must be able to drop any course they registered for.

Measurable: Dropping is allowed only if the course is in the student’s list.

Achievable: Remove course object from student's schedule.

Relevant: Needed for flexible registration.

Time-bound: Must be available in the same menu as registration.

FR5 – View Student Schedule

Specific: Students must be able to view all registered courses in a list format.

Measurable: The list must show course code, name, instructor, and credit hours.

Achievable: Data retrieved from student's course vector.

Relevant: Helps students track their registration.

Time-bound: Must be available after registration and drop modules.

FR6 – Admin Add/Edit/Delete Courses

Specific: Admin can add new courses, update existing ones, and delete courses.

Measurable: Admin actions must reflect immediately in the stored data.

Achievable: CRUD operations on the course list.

Relevant: Keeps system courses updated.

Time-bound: Implemented after student modules.

FR7 – Limit Enrollment Capacity

Specific: Each course has a maximum capacity (e.g., 40 students).

Measurable: System must prevent registration when capacity is full.

Achievable: Check size of enrolledStudents vector.

Relevant: Prevents over-enrollment.

Time-bound: Must be enforced every time a student registers.

✅ 2) Non-Functional Requirements (SMART)

NFR1 – Performance

Specific: System should load all course data within 2 seconds.

Measurable: Measured during startup.

Achievable: Data sets are small.

Relevant: Console apps should be fast.

Time-bound: Evaluated before final submission.

NFR2 – Usability

Specific: The console UI must be simple with menus and clear options.

Measurable: User should complete main tasks in fewer than 5 steps.

Achievable: Menus and text prompts.

Relevant: Students will use it easily.

Time-bound: Final UI polishing in last development phase.

NFR3 – Reliability

Specific: System should prevent invalid input (non-digit, empty, etc.).

Measurable: At least 90% of invalid inputs must be handled gracefully.

Achievable: Input validation.

Relevant: Avoid crashes during operation.

Time-bound: Tested weekly during development.

NFR4 – Security

Specific: Passwords must not be displayed on the screen.

Measurable: Login feature must hide or mask password input.

Achievable: Using simple masking (e.g., *).

Relevant: Protects student accounts.

Time-bound: Implemented with login module.

NFR5 – Portability

Specific: System must run on Windows and Linux.

Measurable: Test on both OS environments.

Achievable: Written in standard C++.

Relevant: Console apps must be portable.

Time-bound: Tested before final submission.