University Management System – Requirements Specification

Extracted & Structured from Provided Java Code

Date: January 2026

System Type: Academic ERP-like application (in-memory prototype with Swing GUI)

1. Functional Requirements

ID	Category	Requirement Description	Main Implementation Location	Priority
F01	Student Management	Create new student record (ID, name, email, age, major) with default values (GPA=0, credits=0, level=Freshman, etc.)	`StudentController.createStudent()`	High
F02	Student Management	Enroll student in course (checks capacity, max credits by GPA, updates both student & course)	`EnrollmentService.enrollStudent()`	High
F03	Student Management	Drop enrolled course (updates counts and credit hours on both sides)	`EnrollmentService.dropCourse()`	High
F04	GPA Calculation	Calculate cumulative GPA from list of grades & corresponding credits	`Student.calculateGPA()`	High
F05	GPA Calculation	Calculate semester GPA (excluding PASS/FAIL courses)	`Student.calculateSemesterGPA()`	Medium
F06	Academic Standing	Determine standing: Dean's List, Good, Satisfactory, Probation, Suspension, Part-time	`Student.determineAcademicStanding()`	High
F07	Academic Level	Classify level: Freshman / Sophomore / Junior / Senior / Graduate based on completed credits	`Student.determineAcademicLevel()`	Medium
F08	Enrollment Rules	Validate regular course enrollment (max 18–21 credits depending on GPA)	`Student.canEnroll()`	High
F09	Honors Courses	Check eligibility for honors courses (GPA ≥ 3.5, ≥30 credits, prerequisites met)	`Student.canEnrollInHonorsCourse()`	Medium

ID	Category	Requirement Description	Main Implementation Location	Priority
F10	Graduation Eligibility	Check if student can graduate (≥120 credits, GPA ≥ 2.0, major ≥60, elective ≥30)	`Student.isEligibleForGraduation()`	High
F11	Graduation Planning	Calculate remaining credits needed to reach degree requirement	`Student.calculateRemainingCredits()`	Medium
F12	Graduation Planning	Estimate number of semesters left to graduate given credits/semester	`Student.calculateSemestersToGraduation()`	Medium
F13	Graduation Forecasting	Predict final GPA at graduation given current GPA and expected performance in remaining credits	`Student.predictGraduationGPA()`	Medium
F14	Tuition Calculation	Calculate tuition (in-state $350/cr, out-of-state $850/cr + $500 flat fee, scholarship discount)	`Student.calculateTuitionFees()`	High
F15	International Fees	Calculate international student fees ($1200/cr + fixed fees + visa fee, 5% discount ≥15 credits)	`Student.calculateInternationalFees()`	Medium
F16	Course Management	Create new course (ID, name, instructor, credits, capacity, department)	`CourseController.createCourse()`	High
F17	Course Availability	Calculate available seats & fill rate (%)	`Course.calculateAvailableSeats()`, `calculateFillRate()`	High
F18	Course Demand	Classify course popularity: High / Moderate / Low / Very Low demand based on fill rate	`Course.determineCoursePopularity()`	Medium
F19	Grade Conversion	Convert numeric score → letter grade → GPA points (standard 4.0 scale, A=93–100, etc.)	`Grade.calculateLetterGrade()`, `convertLetterToGPA()`	High
F20	Grade Statistics	Calculate course average, weighted average, curved grade, standard deviation	`GradeCalculationService.*` methods	High
F21	Grade Adjustment	Apply attendance/participation bonus, extra credit, drop lowest N scores	`applyBonusPoints()`, `calculateDroppedLowestScores()`	Medium
F22	Latin Honors	Determine graduation honors level (Cum Laude, Magna, Summa) based on final GPA	`GradeCalculationService.determineHonorsLevel()`	Medium

ID	Category	Requirement Description	Main Implementation Location	Priority
F23	Scholarship	Calculate merit scholarship amount (0–100% of base tuition based on GPA & credit load)	`FinancialService.calculateScholarshipAmount()`	High
F24	Loan Eligibility	Estimate federal student loan amount (dependent/independent, income bands)	`FinancialService.calculateLoanEligibility()`	Medium
F25	Payment Plans	Calculate monthly installment with interest	`FinancialService.calculatePaymentPlan()`	Medium
F26	Late Fees	Apply progressive late payment penalties	`FinancialService.applyLateFee()`	Medium
F27	Scheduling	Detect time conflict between two classes	`SchedulingService.hasTimeConflict()`	Medium
F28	Institutional Reporting	Calculate graduation rate, retention rate, department revenue, institutional health score	`ReportingService.*` methods	Medium

2. Non-Functional Requirements (Inferred from Code)

ID	Category	Characteristic / Constraint	Evidence / Observation	Strength
NF01	Correctness	Defensive programming – many methods return -1 / "INVALID" / false on bad input	Almost every public method	Very strong
NF02	Input Validation	Extensive checks for null, negative values, list size mismatch, invalid ranges	Repeated in nearly all calculation methods	Very strong
NF03	Numeric Precision	Money & GPA values rounded to exactly 2 decimal places	Consistent `Math.round(… * 100.0) / 100.0` pattern	Strong
NF04	Data Consistency	Atomic updates to student & course entities during enroll/drop	`enrollStudent()` & `dropCourse()` update both sides	Good
NF05	Persistence	In-memory only (HashMap-based repositories) – no database or file storage	All `*Repository` classes	Very weak
NF06	Scalability	Not designed for large data volumes or concurrency (no locking, single-threaded Swing)	In-memory maps + Swing EDT	Poor

ID	Category	Characteristic / Constraint	Evidence / Observation	Strength
NF07	Usability	Tabbed Swing GUI with real-time status bar and activity log	`MainFrame` , various `*View` classes	Good
NF08	Maintainability	Layered architecture (repository → service → controller → view)	Clear separation	Good
NF09	Code Duplication	High repetition of input validation logic	Validation blocks repeated many times	Medium–Poor
NF10	Testability	Many pure functions without side effects	Calculation methods generally testable	Good
NF11	Internationalization	Hard-coded English strings, US grading scale, US-centric tuition & visa logic	Letter grades A–F, $ currency, F1/J1/M1 visas	Poor
NF12	Security	No authentication, authorization, input sanitization	Open access, no login system	Very weak
NF13	Performance	O(n) filtering on collections via streams	`findByXxx()` methods	Acceptable (small data)
NF14	Error Handling	Magic return values instead of exceptions	-1.0, "INVALID", false, null	Consistent but primitive
NF15	Logging	Basic append-only JTextArea activity log – no structured logging framework	`displayMessage()` calls	Very basic
NF16	Documentation	Almost no Javadoc or inline comments (only method names)	Code relies on self-documenting names	Poor

3. Missing / Future Features (Obvious Gaps)

- Database / file persistence
- User authentication & roles (admin, student, faculty, advisor)
- Prerequisite checking during enrollment
- Waitlist management
- Course scheduling / timetable conflict detection (beyond simple pairwise check)
- Grade entry & transcript generation
- Search/filter students/courses by multiple criteria in GUI
- Export reports (PDF/CSV)
- Multi-semester/year support in grade history

End of Document