Software Requirement Specification (SRS) CampusERP – Smart College Management System

---

Document Control

Item	Description
Document Title	Software Requirement Specification (SRS)
System Name	CampusERP – Smart College Management System
Module	Academic Module
Version	1.0
Prepared For	Institutional Use
Prepared By	CampusERP Development Team
Date	December 2025

---

1. Introduction

1.1 Purpose

This document defines the **Software Requirement Specification (SRS)** for the **Module** of **CampusERP – Smart College Management System**.

The objectives of this document are to:

- Establish **clear modular boundaries and responsibilities**
- Ensure **scalability, maintainability, and extensibility**
- Enable **seamless integration with dependent modules**
- Provide a **single source of truth** for developers, testers, and stakeholders
- Support **future academic regulations and compliance requirements**

1.2 Scope

CampusERP is a **centralized, web-based ERP platform** designed to automate and integrate **academic and administrative operations** of educational institutions.

The **Academic Module** acts as the **core configuration and reference module** supplying academic structure data to:

1.3 Intended Users

User Type	Description
College Management	Strategic oversight and analytics
Academic Administrators	Academic configuration and control
Faculty Members	Teaching, attendance, assessments
Examination Controller	Exams & results
Students	Academic access
Parents / Guardians	Academic monitoring
ERP Administrator	System management

Access is governed through **Role-Based Access Control (RBAC)**.

1.4 Definitions, Acronyms, and Abbreviations

Term	Description
ERP	Enterprise Resource Planning
RBAC	Role-Based Access Control
API	Application Programming Interface
UI	User Interface
SRS	Software Requirement Specification

---

2. Overall System Description

2.1 System Perspective

CampusERP follows a **modular, service-oriented architecture** with a centralized database.

Key Characteristics

- Loosely coupled modules
- Centralized academic master data
- Secure authentication & authorization
- API-based inter-module communication

The **Academic Module** serves as a **master data provider** for multiple downstream modules.

2.2 High-Level System Modules

No.	Module Name	Description
1	**Dashboard**	Displays system overview, analytics, and KPIs
2	**Alerts**	Generates alerts for important events
3	**Reports**	Produces academic and operational reports
4	**Notifications**	Sends SMS, email, and in-app notifications
5	**Admissions**	Manages student admission and enrolment
6	**Students**	Maintains student profiles and lifecycle
7	**Academic**	Manages academic structure and curriculum
8	**Staff**	Manages faculty and staff information
9	**Attendance**	Tracks student attendance
10	**Timetable**	Manages class and faculty schedules
11	**Exams**	Handles examination processes
12	**Library**	Manages library resources
13	**Hostel**	Manages hostel facilities
14	**Transport**	Manages transport operations
15	**Finance**	Manages fees and financial transactions

---

3. Academic Module

3.1 Role of Academic Module

The Academic Module defines the **institution's academic framework** and provides standardized reference data.

3.2 Academic Master Entities

- Regulation
- Academic Year
- Department
- Course
- Class
- Section
- Semester

- Subject

All academic transactions depend on these masters.

---

4. Person Management (Core Identity Model)

4.1 Concept

The **Person** entity is the **central identity master** for all humans associated with the institution.

4.2 Supported Roles

- Student
- Teaching Staff
- Non-Teaching Staff
- Admin / Office Staff
- Parent / Guardian

4.3 System Layers

- **Identity Layer** – Person
- **Role Layer** – Student, Staff, Parent
- **Relationship Layer** – Enrollment, Teaching Assignment

4.4 Master Entity Flow (High-Level)

Person

├── Contact

├── Address

├── PersonDocument

├── UserAccount

│ └── UserRole → Role

├── Student

│ └── ParentGuardian → Person (Parent)

└── Staff

├── Teaching

└── Non-Teaching

4.5 Real-Time Scenarios

👨 New Student Admission

1. Create Person
2. Add Contact & Address
3. Upload Documents
4. Create Student record
5. Map Parent / Guardian

👩‍💼 New Staff Joining

1. Create Person
2. Add Contact & Address
3. Upload Documents
4. Create Staff record
5. Create User Account
6. Assign Roles

🔄 Student Becomes Staff

- Same Person retained
- Student record preserved
- New Staff record created

---

5. Staff Module

5.1 Teaching Assignment

TeachingAssignment defines:

Which staff teaches which subject,
for which class & section,
in which semester & academic year,
under which regulation.

5.2 Used By

- Timetable
- Attendance
- Exams
- Faculty workload
- Reports

👩‍🏫 **Faculty Assignment Flow**

1. Create Person → Staff
2. Assign Subject via TeachingAssignment
3. Timetable & Attendance auto-consume mapping

---

6. Student Module

6.1 Student Enrollment

StudentEnrollment defines:

Which student studies
in which class & section,
for which semester & academic year,
under which regulation.

6.2 Importance

Without StudentEnrollment:

- ❌ Attendance impossible
- ❌ Exams impossible
- ❌ Promotions impossible
- ❌ Results impossible

6.3 Entity Flow

Person
└── Student
  └── StudentEnrollment
    ├── Class
    ├── Section
    ├── Semester
    ├── AcademicYear
    └── Regulation

---

7. Timetable Module

7.1 Purpose

Defines:

- When a class occurs
- Which subject is taught
- Which faculty teaches
- Which class & section attends

7.2 Dependency Model

Timetable is a **runtime operational layer** built on:

- TeachingAssignment
- StudentEnrollment

7.3 Timetable Flow

🗓️ **Timetable Creation**

1. Create TeachingAssignment
2. Create timetable slots
3. Assign time & room
4. Publish timetable

7.4 Views

👨‍🎓 **Student View**

Student

→ StudentEnrollment

→ Class & Section

→ Timetable

→ Subject + Faculty + Time

👩‍🏫 **Faculty View**

Staff

→ TeachingAssignment

→ Timetable

→ Class + Subject + Time

---

8. Attendance Module

8.1 Dependency

Attendance **strictly depends** on Timetable.

Attendance

└── Timetable

└── TeachingAssignment

└── Subject + Staff

❌ No Timetable → ❌ No Attendance

8.2 Attendance Flow

👩‍🏫 **Faculty Marks Attendance (QR / Manual)**

1. Login
2. View today's timetable
3. Select lecture
4. System loads students (via Student Enrolment)
5. Mark attendance
6. Save

👨‍🎓 **Student View**

- Subject-wise attendance %
- Monthly attendance report

➞ Exam / Internal Assessment

COMPLETE REAL-TIME FLOW (STEP BY STEP)

👩‍🏫 **Faculty Side (Teacher)**

1. Faculty logs in

2. Selects Subject → Data Structures

3. Selects Assessment → Unit Test 1

4. System fetches students using:

StudentEnrollment (Sem 3, Section A)

5. Faculty enters marks

6. Data saved in StudentAssessmentResult

🎓 **Student Side (Rahul)**

1. Rahul logs in

2. Clicks "My Results"

3. Selects Semester 3

4. System shows:

Data Structures

├── Unit Test 1 : 24 / 30

├── Mid Sem : 21 / 30

└── End Sem : 68 / 100

Student

→ StudentEnrollment

→ Semester

→ Assessment

→ StudentAssessmentResult

Module Structure

COMMON static academic MASTERS?

These are **static academic definitions** used by **many modules**:

Course

Class

Section

Semester

AcademicYear

Regulation

Subject

1️⃣ **StudentEnrollment**

student_id

course_id → Course (MASTER)

class_id → Class (MASTER)

section_id → Section (MASTER)

semester_id → Semester (MASTER)

academic_year_id → AcademicYear (MASTER)

regulation_id → Regulation (MASTER)

status

❓ **WHY ALL THESE FK ARE HERE?**

Because:

👉 A student is enrolled **IN ALL OF THESE TOGETHER**
👉 Enrollment is **time-bound**
👉 These can change over time

Student: Aman

Year 2024

Course : B.Tech

Class : CSE

Section : A

Semester : 1

Academic Year : 2024–25

Regulation : R2023

➞ Stored in **ONE StudentEnrollment ROW**

Year 2025 (Promotion)

Course : B.Tech

Class : CSE

Section : B (changed)

Semester : 3

Academic Year : 2025–26

Regulation : R2023

➞ **NEW StudentEnrollment ROW**

❗ Old record is NEVER updated
❗ History preserved

🔁 **WHY NOT PUT THESE IN Student TABLE?**

Because:

Problem	**If put in Student**
Promotion	Data overwrite
Section change	History lost
Regulation change	Impossible
Readmission	Broken
Backlog	Broken

✓ Student = identity
✓ Enrollment = academic state

🧩 **FLOW**

Person
└── Student
  └── StudentEnrollment
    ├── Course (MASTER)
    ├── Class (MASTER)
    ├── Section (MASTER)
    ├── Semester (MASTER)
    ├── AcademicYear (MASTER)
    └── Regulation (MASTER)

🔹 **CENTER = PERSON (ROOT)**

Every human is created **ONCE.**

Person

├── Student

├── Staff (Faculty / Admin)

└── Parent (optional)

(ALL MODULES TOGETHER)

Person

├── Contact / Address / Document

├── UserAccount → Role

│

├── Student

│ └── StudentEnrollment

│ ├── Course (MASTER)

│ ├── Class (MASTER)

│ ├── Section (MASTER)

│ ├── Semester (MASTER)

│ ├── AcademicYear (MASTER)

│ └── Regulation (MASTER)

│

│ ├── TeachingAssignment

│ │ ├── Subject (MASTER)

│ │ └── Faculty (Staff)

│

│ ├── Timetable

│ │ ├── Day / Period

│ │ └── TeachingAssignment

│

│ ├── Attendance

│ │ ├── Student

│ │ └── Timetable Slot

```
│
│       ├── Assessment
│       │     ├── Internal (UT, Mid)
│       │     └── External (End Sem)
│       │
│       │     └── StudentAssessmentResult
│       │
│       └── Result Processing
│             ├── SubjectResult
│             ├── SemesterResult (SGPA)
│             └── CGPAResult
│
└── Staff
    ├── Employment Details
    └── TeachingAssignment
```

1️⃣ StudentEnrollment

📌 Snapshot of student's academic state
➡ Course + Class + Section + Semester + Year + Regulation
✓ History preserved (promotion, section change)

2️⃣ TeachingAssignment

📌 Answers: **WHO teaches WHAT to WHICH class**

Faculty + Subject + Semester/Class

✓ Used by Timetable, Attendance, Exams

3️⃣ Timetable

📌 Answers: **WHEN a class happens**

Day + Period + TeachingAssignment

4️⃣ Attendance

📌 Answers: **WHO attended WHICH class**

Student + Timetable Slot

5️⃣ Assessment (Exam)

📌 Defines exams

Internal / External

Subject + Semester + Max Marks

6️⃣ StudentAssessmentResult

📌 Stores **marks of each student**

7️⃣ Result Processing

📌 Converts marks → grades

StudentAssessmentResult

→ SubjectResult

→ SemesterResult (SGPA)

→ CGPAResult

🎓 REAL-LIFE STORY (ONE STUDENT – FULL LIFE)

Rahul joins B.Tech CSE – 2024

1️⃣ Rahul → **Person**
2️⃣ Login created → **UserAccount (STUDENT)**
3️⃣ Admission → **Student**
4️⃣ Assigned → **StudentEnrollment (CSE, Sem-1, Sec-A)**
5️⃣ Faculty mapped → **TeachingAssignment**
6️⃣ Weekly plan → **Timetable**
7️⃣ Daily class → **Attendance**
8️⃣ UT + End Sem → **Assessment**
9️⃣ Marks saved → **StudentAssessmentResult**
🔟 Grades → **SubjectResult**

11 SGPA → **SemesterResult**

12 Final CGPA → **CGPAResult**