Software Requirements Specification (SRS)

School Management System

Version: 1.0.0 | **Date:** December 2025 | **Author:** Satya Narayan Sahu **Technology Stack:** Python 3.8+, MySQL 8.0+, PyMySQL |

Table of Contents

1. Introduction

1.1 Purpose

The School Management System (SMS) is a comprehensive, enterprise-grade software solution designed to digitize and automate administrative, academic, and operational functions for Central Board of Secondary Education (CBSE) schools. The system replaces traditional paper-based processes with secure, real-time digital workflows to ensure data integrity, operational efficiency, and compliance with educational standards.

1.2 System Objectives

Business Objectives: - Reduce administrative workload by 70% through process automation - Ensure 100% data integrity and real-time information access - Achieve cost savings through paperless operations and streamlined workflows - Provide scalable solution supporting 100-5,000+ students with multi-role access - Maintain CBSE curriculum compliance and local education regulation adherence

Technical Objectives: - Implement ACID-compliant MySQL database with robust transaction management - Deploy SHA256 password hashing with comprehensive security framework - Ensure cross-platform compatibility (Windows, macOS, Linux) with Python 3.8+ - Maintain 99.9% system uptime with sub-second response times - Deliver modular, maintainable codebase following industry best practices

1.3 Scope of Work

System Scope: - Multi-role user authentication and authorization framework - Complete student lifecycle management (admission to graduation) - Teacher profile and assignment management system - Academic structure administration (classes, subjects, timetables) - Daily attendance tracking and analytics system - Comprehensive reporting and dashboard capabilities - System administration and maintenance tools

Exclusions: - Online examination and assessment platforms - Financial management (fees, payroll, procurement) - Library management and inventory systems - Transportation and logistics management - Mobile applications and SMS/email gateways - Parent portals and external communication systems

System Constraints: - Command-line interface only (no graphical user interface) - MySQL database server dependency (local or network) - Local Area Network connectivity requirement - English language interface and documentation only

1.4 System Overview

The SMS implements a three-tier client-server architecture with clear separation of concerns:

Presentation Tier: Command-line interface with role-based dashboards, input validation, and user guidance **Application Tier:** Business logic layer with 50+ methods, transaction management, and security controls **Data Tier:** MySQL relational database with 14 normalized tables and 25+ foreign key relationships

1.5 Definitions and Acronyms

Term	Definition
SMS	School Management System - The complete software application
CBSE	Central Board of Secondary Education - Indian educational board
CLI	Command Line Interface - Text-based user interaction method
ACID	Atomicity, Consistency, Isolation, Durability - Database transaction properties
RBAC	Role-Based Access Control - Security authorization methodology
SHA256	Secure Hash Algorithm 256-bit - Password hashing standard
CRUD	Create, Read, Update, Delete - Basic database operations

Term	Definition
FK/PK	Foreign Key/Primary Key - Database relationship constraints

1.6 References and Standards

- **IEEE 830-1998:** Software Requirements Specifications standard
- **CBSE Curriculum Guidelines 2024:** Educational standards compliance
- **ISO/IEC 9126:** Software quality characteristics framework
- **OWASP Security Guidelines:** Application security best practices
- **PEP 8:** Python coding standards and style guide
- **MySQL 8.0 Documentation:** Database implementation reference

2. Overall System Description

2.1 System Context

The School Management System operates as a client-server application within a Local Area Network (LAN) environment, connecting multiple user roles to a centralized MySQL database. The system serves CBSE school administrators, teaching staff, students, and support personnel through secure, role-based access to educational data and operational functions.

2.2 System Boundaries

Included Functionality: - User account management and authentication framework - Student profile management and academic tracking - Teacher profile and assignment administration - Academic structure management (classes, subjects, schedules) - Daily attendance recording and analytical reporting - Administrative reporting and system management tools

Excluded Functionality: - Financial transaction processing (fees, salaries, billing) - Online examination platforms and assessment tools - Library management and inventory tracking systems - Transportation management and logistics - Mobile applications and responsive web interfaces - External communication systems (SMS, email, notifications)

2.3 System Functions and Capabilities

Core Business Functions: 1. **Authentication & Authorization:** Secure user login with role-based access control 2. **Student Management:** Complete lifecycle from admission through graduation 3. **Teacher Management:** Profile creation, assignment management, privilege administration 4. **Academic Management:** Class, subject, and timetable administration 5. **Attendance System:** Daily marking, historical tracking, analytical reporting 6. **Reporting**

& Analytics: Comprehensive dashboards and export capabilities 7. **System Administration:** Database maintenance, backup, and health monitoring

2.4 User Characteristics

User Role	Expertise Level	Training Required	Frequency	Primary Responsibilities
System Administrator	High (Technical)	4-8 hours	Daily	System configuration, user management, maintenance
School Principal	Medium	2-4 hours	Daily	Oversight, reporting, policy implementation
Academic Coordinator	Medium-High	3-4 hours	Daily	Curriculum management, teacher coordination
Admission Department	Medium	2-3 hours	Daily	Student enrollment, documentation
Teachers	Low-Medium	1-2 hours	Daily	Attendance marking, student management
Students	Low	15-30 minutes	Weekly	Personal information access
Support Staff	Medium-High	4-6 hours	As needed	Technical support, troubleshooting

2.5 Operational Environment

Hardware Requirements: - **Minimum:** Intel Core i3 / AMD Ryzen 3, 2GB RAM, 500MB storage - **Recommended:** Intel Core i5 / AMD Ryzen 5, 4GB RAM, 1GB storage - **Server (Database):** Intel Xeon / AMD EPYC, 8GB+ RAM, 100GB+ SSD storage

Software Requirements: - **Operating Systems:** Windows 10+, macOS 10.14+, Ubuntu 18.04+ - **Runtime:** Python 3.8.0 or later with pip package manager - **Database:** MySQL 8.0+ Community/Enterprise edition - **Dependencies:** PyMySQL 1.0.2+, standard Python libraries

Network Requirements: - Local Area Network with 10-100 Mbps connectivity - TCP/IP protocol support with firewall configuration - Database server accessibility on standard MySQL port (3306)

Environmental Constraints: - Maximum 20 concurrent user sessions - Business hours operation (8 AM - 6 PM, Monday-Saturday) - LAN-only connectivity (no WAN/remote access)

3. System Architecture and Design

3.1 System Architecture

Three-Tier Pattern: - **Presentation Layer:** CLI with hierarchical menus, input validation - **Application Layer:** Business logic in Python classes, transaction management - **Data Layer:** MySQL with ACID compliance, referential integrity

Design Patterns: - Singleton: Database connections, configuration - Factory: User creation, report generation - Strategy: Authentication methods, reporting formats - Observer: Audit logging, error notifications

3.2 Database Design

14 Core Tables: - `users`: Authentication and roles - `teachers`: Teacher profiles and qualifications - `students`: Student profiles and enrollment - `classes`: Class-section definitions - `subjects`: Subject catalog with assignments - `timetable`: Class schedules - `teacher_assignments`: Teacher-class-subject relationships - `student_subjects`: Automatic subject enrollment - `student_attendance`: Daily attendance records - `teacher_attendance`: Staff attendance tracking - `student_status`: Student status management - `teacher_status`: Teacher status tracking - `teacher_privileges`: Granular permission settings - `teaching_records`: Employment history - `proxy_teachers`: Substitute teacher assignments

Database Relationships: - One-to-One: users teachers, users students - One-to-Many: classes students, teachers teaching_records - Many-to-Many: teachers subjects (via assignments)

3.3 User Interface Design

Menu Hierarchy:

```
Admin Dashboard
  User Management (Create Teacher/Student/Principal/etc.)
  Academic Management (Classes, Subjects, Timetable)
  Attendance Management (Mark/View Attendance)
  System Administration (Privileges, Maintenance)
  Reporting (Student/Teacher/Class Reports)

Teacher Dashboard
  Attendance (Mark Student Attendance)
  Class Management (View Students, Status)
  Academic (Timetable, Assignments)
  Reports (Class Attendance, Performance)

Student Dashboard
  Academic (Timetable, Subjects, Attendance)
```

```
   Personal (Profile, Contact Updates)

Principal/Dashboard (Read-Only)
   Student/Teacher/Class Overviews
   Attendance Analytics
   Academic Performance Reports
```

3.4 Security Architecture

Authentication: SHA256 hashing, session management, account lockout **Authorization:** 7-role RBAC with granular permissions **Data Protection:** Parameterized queries, input validation, audit logging

4. Functional Requirements

4.1 User Management System

SRS-USER-AUTH-001: User Authentication Process Description: Secure user authentication with credential verification and session management **Pre-conditions:** Valid user account exists in database with active status **Functional Requirements:** - Accept username and password via secure CLI input - Validate username format (3-50 alphanumeric characters + underscore) - Verify password hash using SHA256 algorithm with salt - Check account status (active/suspended/removed) for teachers - Establish session with role-based permissions and 30-minute timeout - Log successful authentication with timestamp and user details **Error Conditions:** Invalid credentials, account locked, database unavailable

SRS-USER-AUTH-002: Role-Based Access Control Description: Hierarchical permission system with granular access control **Functional Requirements:** - Implement 7 predefined user roles with specific permission sets - Validate permissions before executing sensitive operations - Provide administrative override capabilities for emergency access - Log all permission-based access decisions for audit purposes - Support dynamic permission assignment and revocation

SRS-USER-CREATE-001: Teacher Account Creation Description: Create complete teacher profile with professional qualifications **Input Requirements:** - Personal: Full name, age (18-70), date of birth, contact information - Professional: Highest qualifications, primary teaching subject - Employment: Teaching records and experience history **Processing Logic:** - Generate unique username (firstname.lastname format) - Validate username uniqueness system-wide - Set default password (teacher123) requiring immediate change - Create linked user and teacher table entries - Validate all required fields and business rules **Business Rules:** Qualifications mandatory, subject expertise required

SRS-USER-CREATE-002: Student Account Creation Description: Create comprehensive student profile with academic enrollment **Input Requirements:** - Academic: Admission number (unique), full name, age (5-25), date of birth - Personal: Contact information, emergency contacts - Family: Parent/guardian names, occupations, contact details - Academic: Class assignment and enrollment details **Processing Logic:** - Validate admission number uniqueness across all records - Generate username (firstname.lastname format) - Set default password (student123) for initial login - Create linked user and student table entries - Auto-assign subjects based on class curriculum - Validate class capacity and enrollment constraints **Business Rules:** Unique admission numbers, parent consent implied

4.2 Student Management System

SRS-STUDENT-PROFILE-001: Student Lifecycle Management Description: Complete student journey management from admission to graduation **Lifecycle Stages:** - **Admission:** Initial enrollment with comprehensive profile creation - **Active:** Regular academic participation and attendance tracking - **Suspended:** Temporary removal with reason and duration tracking - **Transferred:** Movement to another institution with record transfer - **Graduated:** Program completion with final academic records - **Removed:** Permanent removal from system (administrative action) **Functional Requirements:** - Status transition validation and business rule enforcement - Automatic access control adjustments based on status - Historical status tracking with complete audit trail - Cascading updates to related academic records - Status change notifications and documentation requirements

SRS-STUDENT-PROFILE-002: Student Information Updates Description: Modify student profile information with proper authorization controls **Modifiable Fields:** - Personal: Name, age, date of birth, contact information - Academic: Previous school, performance records, special needs - Family: Parent/guardian details, emergency contacts, relationships - Medical: Health conditions, allergies, emergency procedures **Authorization Matrix:** - Admin: Full access to all fields - Admission Department: Contact and enrollment information - Academic Coordinator: Academic and performance data - Class Teacher: Limited contact information updates - Student: Personal contact details only **Validation Requirements:** - Critical fields require enhanced verification (name changes, DOB) - Medical information requires healthcare provider validation - Contact changes require verification through multiple channels

SRS-STUDENT-CLASS-001: Class Assignment Management Description: Change student class and section assignments with cascading updates **Process Flow:** 1. Display current class-section assignment with enrollment history 2. Show available class-section combinations with capacity status 3. Validate selection against academic progression rules 4. Update student record

with new assignment and effective date 5. Automatically remove existing subject enrollments 6. Create new subject enrollments based on destination class curriculum 7. Update teacher assignment references and attendance records **Business Rules:** - Students can only be enrolled in one class-section simultaneously - Class capacity limits enforced with configurable thresholds - Academic year progression validation (age-appropriate placements) - Subject enrollment automatic recalculation and reconciliation

SRS-STUDENT-STATUS-001: Student Status Change Operations Description: Change student status with audit trail and business rule enforcement **Status Types:** - **Active:** Normal student status with full system access - **Suspended:** Temporary removal with specified reason and duration - **Removed:** Permanent removal requiring administrative approval - **Transferred:** Movement to another school with record portability - **Graduated:** Program completion with archival status **Authorization Requirements:** - Suspension: Admin or teachers with suspension privileges - Removal: Admin only (destructive operation requiring dual authorization) - Transfer/Graduation: Admin or admission department approval **Audit Requirements:** - All status changes logged with user, timestamp, and detailed reason - Status change history maintained indefinitely for compliance - Automatic notifications to relevant stakeholders - Documentary evidence requirements for significant status changes

4.3 Teacher Management System

SRS-TEACHER-PROFILE-001: Teacher Profile Creation and Management Description: Create and maintain comprehensive teacher profiles with professional qualifications **Required Information:** - **Personal:** Full name, age (18-70), date of birth, contact information, address - **Professional:** Highest qualifications, specialization areas, certifications - **Teaching:** Primary subject expertise, secondary subjects, teaching methodology - **Employment:** Current status, experience years, previous school affiliations - **Administrative:** System access level, reporting relationships **Functional Requirements:** - Comprehensive field validation with business rule enforcement - Employment history tracking with unlimited previous positions - Qualification verification workflow and certification validation - Automatic user account creation with role-based permissions - Profile completeness validation with required field indicators **Business Rules:** - Minimum qualification requirements for teaching positions - Subject expertise must align with assigned teaching subjects - Employment history required for experience validation - Profile updates require approval for critical information changes

SRS-TEACHER-ASSIGNMENT-001: Class-Subject Assignment Management Description: Assign teachers to specific class-subject combinations with conflict resolution **Assignment Process:** 1. Select teacher from active staff with current workload display 2. Choose target class name and section combination 3. Select specific subject from available class curriculum 4.

Validate assignment constraints and availability 5. Create teacher_assignments record with effective date and assignment type 6. Update subjects table with teacher assignment 7. Notify affected stakeholders of assignment changes **Validation Requirements:** - Teacher active status and employment verification - Class and section existence and active enrollment status - Subject availability within selected class curriculum - Schedule conflict detection with existing assignments - Qualification compatibility between teacher and subject - Workload balancing against configurable maximum assignments **Business Rules:** - Teachers may have multiple concurrent assignments - Unique constraint prevents duplicate teacher-class-subject assignments - Subject must exist in assigned class curriculum - Assignment changes require validation of impacted stakeholders

SRS-TEACHER-PRIVILEGE-001: Teacher Privilege Administration Description: Configure fine-grained permissions for teacher accounts beyond basic role access **Privilege Categories:** - **Student Management:** View, edit, create, delete, suspend student records - **Academic Management:** Modify subjects, assignments, timetables, curriculum - **Attendance Management:** Mark, edit, delete, analyze attendance records - **Reporting:** Generate reports, export data, access analytics - **Administrative:** System configuration, user management, maintenance access - **Communication:** Access to student/parent contact information **Configuration Framework:** - Individual privilege toggling with enable/disable states - Privilege templates for common teaching roles and responsibilities - Time-based privilege activation and expiration - Department-specific privilege sets and inheritance rules - Emergency privilege elevation with approval workflows **Security Controls:** - Secure by default (all privileges disabled initially) - Explicit permission granting required by authorized administrators - Privilege validation before executing privileged operations - Comprehensive audit logging of privilege changes and usage - Regular privilege review and certification processes **Business Rules:** - Privilege assignments must align with job responsibilities - Critical privileges require dual authorization for assignment - Privilege usage monitored for anomalous patterns - Automatic privilege revocation on role changes or terminations

4.4 Academic Management System

Class Management URS-ACADEMIC-001: Class Creation - Input: Class name (e.g., "12th"), section (A-Z single letter) - Validation: Unique class-section combination, naming conventions - Post-Creation: Ready for subject and student assignment

Subject Management URS-ACADEMIC-002: Subject Creation and Assignment - Class-specific subject creation with teacher assignment - Validation: Subject uniqueness within class, teacher qualification - Business Rules:

Automatic student subject enrollment - Reassignment: Teacher changes with validation and history

Timetable Management **URS-ACADEMIC-003: Schedule Creation** - Class-specific timetable with lecture slots - Subject assignment for each time slot - Validation: Time sequencing, teacher availability, subject compatibility - Conflict Detection: Overlapping schedules, resource constraints

4.5 Attendance Management System

Student Attendance **URS-ATTENDANCE-001: Daily Attendance Marking** - Class selection and active student display - Attendance marking (Present/Absent) for each student - Default absent status, duplicate date prevention - Recorder tracking and timestamp recording

URS-ATTENDANCE-002: Attendance Analytics - Individual student attendance history with percentages - Class attendance summaries and trend analysis - Date range filtering and custom reporting - Export capabilities for external analysis

Teacher Attendance **URS-ATTENDANCE-003: Staff Attendance Tracking** - Administrative attendance recording for teachers - Bulk operations for efficiency - Integration with payroll and administrative reporting

Proxy Teacher Management **URS-ATTENDANCE-004: Substitute Teacher Assignment** - Absent teacher replacement with specific timetable entries - Reason recording and approval workflow - Temporary privilege assignment to substitutes

4.6 Reporting and Analytics System

Student Reports **URS-REPORT-001: Student Profile Reports** - Personal and academic information - Attendance summary with statistics - Parent contact information - Status history and current enrollment

URS-REPORT-002: Performance Analytics - Attendance percentages and trends - Comparative analysis across classes - Predictive analytics for attendance patterns

Administrative Reports **URS-REPORT-003: System Overview Dashboards** - User statistics by role and status - Enrollment and capacity metrics - Attendance system-wide analytics - Academic performance summaries

URS-REPORT-004: Teacher Performance Reports - Teaching load analysis and distribution - Attendance marking statistics - Student performance correlations

4.7 System Administration

Database Maintenance URS-ADMIN-001: Database Operations - Automated backup procedures with integrity verification - Table optimization and index rebuilding - Data cleanup and archival operations - Health monitoring and diagnostic capabilities

URS-ADMIN-002: User Privilege Management - System-wide permission administration - Role assignment and modification tracking - Security audit and compliance reporting

5. Non-Functional Requirements

5.1 Performance Requirements

Response Times: - Authentication: <2 seconds - Menu Navigation: <1 second - Data Entry: <1 second - Simple Queries: <3 seconds - Complex Reports: <30 seconds - Bulk Operations: <60 seconds

Throughput: 100 transactions/minute, 10-20 concurrent users **Resource Usage:** <500MB memory, <20% CPU, optimized disk I/O

5.2 Security Requirements

Authentication Security: SHA256 hashing, account lockout, session management **Authorization:** RBAC with granular permissions, privilege validation **Data Protection:** Parameterized queries, input sanitization, audit logging **Compliance:** Data integrity, access logging, security monitoring

5.3 Reliability Requirements

Availability: 99% uptime during business hours **Fault Tolerance:** Graceful error handling, transaction rollback **Data Integrity:** ACID compliance, referential integrity **Recovery:** <4 hours MTTR, automated backup restoration

5.4 Usability Requirements

Learnability: <30 minutes training for basic operations **Efficiency:** Streamlined workflows, keyboard navigation **Error Prevention:** Input validation, confirmation prompts **Accessibility:** Keyboard-only operation, clear error messages

5.5 Maintainability Requirements

Code Quality: PEP 8 compliance, modular design, documentation **Configuration:** External settings, environment variables **Updates:** Easy deployment, rollback capabilities **Monitoring:** Health checks, performance tracking, logging

5.6 Portability Requirements

Platforms: Windows, macOS, Linux compatibility **Deployment:** Single-directory installation, portable execution **Configuration:** Environment-agnostic settings **Dependencies:** Standard library usage, minimal external requirements

6. System Interfaces

6.1 User Interfaces

CLI Interface: Hierarchical menus, input validation, tabular displays **Navigation:** Numbered menu selections, keyboard shortcuts **Feedback:** Real-time validation, progress indicators, error messages

6.2 Database Interfaces

PyMySQL Driver: Parameterized queries, connection pooling, transaction management **Query Execution:** Prepared statements, result set processing, error handling **Connection Management:** Automatic reconnection, timeout handling, resource cleanup

6.3 System Interfaces

File System: Configuration files, log files, export directories **Operating System:** Command execution for utilities, permission management **Network:** TCP/IP connectivity, firewall compatibility

7. System Implementation Details

7.1 Technical Implementation

Core Application: `studentmanage.py` (5,272 lines, 4.6MB) **Database Schema:** 14 tables with 25+ relationships **Security:** SHA256 hashing, RBAC implementation **Architecture:** Object-oriented design with clear separation of concerns

7.2 Installation and Deployment

Prerequisites: Python 3.8+, MySQL 8.0+, pip packages **Database Setup:** Execute setup_database.sql, create application user **Application Setup:** Update credentials, test connectivity **First Run:** Login with admin/admin123, change default password

7.3 System Maintenance

Backup Procedures: Automated daily backups with mysqldump **Optimization:** Table analysis, index rebuilding, statistics updates **Monitoring:** Health

checks, performance metrics, error tracking **Updates:** Version upgrades, configuration changes, security patches

8. Testing and Quality Assurance

8.1 Testing Strategy

Unit Testing: Individual functions and methods validation **Integration Testing:** Module interaction and data flow verification **System Testing:** End-to-end functionality and business process validation **User Acceptance Testing:** Real-world usage scenarios and stakeholder approval **Performance Testing:** Load testing, stress testing, benchmark validation **Security Testing:** Vulnerability assessment and penetration testing

8.2 Test Case Categories

Authentication Testing: Valid/invalid login, role permissions, session management **User Management:** Account creation, profile updates, role assignments **Student Operations:** Enrollment, class changes, status management **Attendance Systems:** Daily marking, bulk operations, analytics **Academic Management:** Class/subject/timetable operations **Reporting:** Data accuracy, export functionality, performance **Administration:** Backup/recovery, maintenance operations

8.3 Performance Benchmarks

Response Times: Authentication <2s, queries <3s, reports <30s **Load Testing:** 20 concurrent users, sustained throughput **Resource Usage:** Memory <500MB, CPU <20%, stable operation **Scalability:** Linear performance degradation with user growth

9. Appendices

9.1 Database Schema and Architecture

Core Database Tables Structure Authentication & User Management:

```
users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL COMMENT 'SHA256 hashed',
    role ENUM('admin', 'principal', 'teacher', 'student',
              'academic_coordinator', 'admission_department') NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    last_login TIMESTAMP NULL,
    login_attempts INT DEFAULT 0,
    account_locked BOOLEAN DEFAULT FALSE,
```

```
    INDEX idx_username (username),
    INDEX idx_role (role)
)

teachers (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT UNIQUE NOT NULL,
    name VARCHAR(100) NOT NULL,
    age INT CHECK (age >= 18 AND age <= 70),
    dob DATE NOT NULL,
    highest_qualifications TEXT NOT NULL,
    teaching_subject VARCHAR(100) NOT NULL,
    status ENUM('active', 'suspended', 'on_leave', 'retired', 'removed') DEFAULT 'active',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
    INDEX idx_user_id (user_id), INDEX idx_teaching_subject (teaching_subject)
)

students (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT UNIQUE NOT NULL,
    admission_number VARCHAR(50) UNIQUE NOT NULL,
    name VARCHAR(100) NOT NULL,
    age INT CHECK (age >= 5 AND age <= 25),
    dob DATE NOT NULL,
    class_id INT,
    contact_number VARCHAR(15) NOT NULL,
    emergency_contact VARCHAR(15),
    status ENUM('active', 'suspended', 'transferred', 'graduated', 'removed') DEFAULT 'activ
    enrollment_date DATE DEFAULT (CURRENT_DATE),
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (class_id) REFERENCES classes(id) ON DELETE SET NULL,
    INDEX idx_admission_number (admission_number), INDEX idx_class_id (class_id)
)
```

Academic Structure:

```
classes (
    id INT PRIMARY KEY AUTO_INCREMENT,
    class_name VARCHAR(50) NOT NULL,
    section VARCHAR(10) NOT NULL,
    capacity INT DEFAULT 50,
    academic_year VARCHAR(20) DEFAULT '2024-2025',
    status ENUM('active', 'inactive') DEFAULT 'active',
    UNIQUE KEY unique_class_section_year (class_name, section, academic_year)
)
```

```
subjects (
    id INT PRIMARY KEY AUTO_INCREMENT,
    subject_name VARCHAR(100) NOT NULL,
    class_id INT NOT NULL,
    teacher_id INT,
    subject_code VARCHAR(20) UNIQUE,
    credits INT DEFAULT 1,
    is_compulsory BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (class_id) REFERENCES classes(id) ON DELETE CASCADE,
    FOREIGN KEY (teacher_id) REFERENCES teachers(id) ON DELETE SET NULL,
    INDEX idx_class_id (class_id), INDEX idx_teacher_id (teacher_id)
)

timetable (
    id INT PRIMARY KEY AUTO_INCREMENT,
    class_id INT NOT NULL,
    day_of_week ENUM('Monday','Tuesday','Wednesday','Thursday','Friday','Saturday') NOT NULL
    lecture_number INT NOT NULL CHECK (lecture_number >= 1 AND lecture_number <= 10),
    start_time TIME NOT NULL,
    end_time TIME NOT NULL,
    subject_id INT NOT NULL,
    room_number VARCHAR(20),
    FOREIGN KEY (class_id) REFERENCES classes(id) ON DELETE CASCADE,
    FOREIGN KEY (subject_id) REFERENCES subjects(id) ON DELETE CASCADE,
    UNIQUE KEY unique_class_day_lecture (class_id, day_of_week, lecture_number)
)
```

Operational Tables:

```
student_attendance (
    id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT NOT NULL,
    date DATE NOT NULL,
    status ENUM('present', 'absent', 'late', 'excused') NOT NULL DEFAULT 'absent',
    recorded_by INT NOT NULL,
    recorded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    remarks TEXT,
    FOREIGN KEY (student_id) REFERENCES students(id) ON DELETE CASCADE,
    FOREIGN KEY (recorded_by) REFERENCES users(id) ON DELETE SET NULL,
    UNIQUE KEY unique_student_date (student_id, date),
    INDEX idx_student_id (student_id), INDEX idx_date (date)
)

teacher_assignments (
    id INT PRIMARY KEY AUTO_INCREMENT,
    teacher_id INT NOT NULL,
    class_id INT NOT NULL,
```

```
    subject_id INT NOT NULL,
    assigned_date DATE NOT NULL DEFAULT (CURRENT_DATE),
    workload_percentage INT DEFAULT 100,
    FOREIGN KEY (teacher_id) REFERENCES teachers(id) ON DELETE CASCADE,
    FOREIGN KEY (class_id) REFERENCES classes(id) ON DELETE CASCADE,
    FOREIGN KEY (subject_id) REFERENCES subjects(id) ON DELETE CASCADE,
    UNIQUE KEY unique_teacher_class_subject (teacher_id, class_id, subject_id)
)
```

Database Relationships Overview

- **One-to-One:** users-teachers, users-students (enforced by UNIQUE constraints)
- **One-to-Many:** classes-students, classes-subjects, teachers-teaching_records
- **Many-to-Many:** teachers-subjects-classes (via teacher_assignments table)
- **Self-Referential:** Hierarchical relationships in organizational structures

Key Design Principles

- **Normalization:** Third normal form (3NF) to eliminate data redundancy
- **Referential Integrity:** Cascading updates/deletes with business rule enforcement
- **Indexing Strategy:** Composite indexes on frequently queried column combinations
- **Constraint Management:** Check constraints for data validation at database level
- **Audit Capability:** Comprehensive tracking of data modifications and access

9.2 System Metrics Summary

- **Application:** 5,272 lines Python code, 4.6MB executable
- **Database:** 14 normalized tables, 25+ foreign keys, 50+ business methods
- **Users:** 7 role types, support for 100-5000 students
- **Performance:** Sub-second responses, <500MB memory usage
- **Scalability:** 50,000+ attendance records, 10-20 concurrent users

9.3 Error Codes Reference

- ERR_AUTH_001: Invalid credentials
- ERR_DB_001: Database connection failed
- ERR_VAL_001: Validation error
- ERR_PERM_001: Insufficient permissions
- ERR_SYS_001: System error

9.4 Current Limitations

- CLI-only interface (no GUI)
- LAN-dependent (no remote access)
- No mobile applications
- Limited concurrent users (20 max)
- Manual backup procedures
- Basic reporting capabilities

Document Version: 1.0.0 | **Date:** December 2025 | **Author:** Satya Narayan Sahu