# Software Requirements Specification (SRS)

## Attendance Management System

### 1. Introduction

#### 1.1 Purpose

This Software Requirements Specification (SRS) document outlines the functional and non-functional requirements for an **Attendance Management System (AMS)** designed for **Law College , Chh. Sambhajinagar**. The system enables teachers to manage attendance for their assigned subjects, view their schedules in a calendar format, and calculate student attendance based only on conducted sessions (excluding holidays or cancellations). It aims to streamline attendance tracking, ensure teacher-specific access, and provide accurate reporting.

#### 1.2 Scope

The AMS will:

* Store and manage data for teachers, students, subjects, timetables, sessions, and attendance.
* Allow teachers to record, edit, and delete attendance for their enrolled students in their scheduled sessions.
* Restrict teachers to managing only their own subjects and students.
* Provide a calendar view of scheduled sessions based on a central timetable.
* Exclude cancelled sessions (e.g., holidays) from attendance calculations.
* Support administrative tasks like timetable setup and holiday management.
* Lay the groundwork for future enhancements (e.g., student portals, detailed reports).

#### 1.3 Definitions, Acronyms, and Abbreviations

* **AMS**: Attendance Management System.
* **Teacher**: Faculty member assigned to teach subjects and manage attendance.
* **Student**: Individual enrolled in subjects whose attendance is tracked.
* **Subject**: Academic course (e.g., "Mathematics").
* **Timetable**: Recurring weekly schedule for subjects.
* **Session**: Specific instance of a class on a given date.
* **Attendance**: Record of a student’s presence in a session.
* **Calendar View**: Visual representation of a teacher’s schedule (e.g., monthly or weekly).

#### 1.4 References

* Database schema provided in the final design (see Section 4.1).
* Discussions on requirements (e.g., teacher-specific access, holiday handling).

#### 1.5 Overview

Section 2 provides a general description of the system’s context and user needs. Section 3 details specific features and functionalities. Section 4 specifies data requirements, including the database schema. Section 5 covers non-functional requirements like performance and security. Sections 6 and 7 outline future scope and assumptions.

### 2. Overall Description

#### 2.1 Product Perspective

The AMS is a standalone system (web or desktop, TBD) with a relational database backend. It integrates teacher authentication, a centralized timetable, and session-based attendance tracking. It interfaces with users (teachers and admins) via a user-friendly application, ensuring data integrity and restricted access.

#### 2.2 Product Functions

* **Teacher Functions**: Authenticate, view calendar, mark attendance (Present/Absent/Late) for their students, edit/delete records.
* **Admin Functions**: Manage teachers, students, subjects, timetables, and holidays.
* **System Functions**: Generate sessions from timetables, calculate attendance based on conducted sessions, maintain audit trails.

#### 2.3 User Classes and Characteristics

* **Teachers**:
  + Need to log in, view their schedules, and mark attendance for their students.
  + Require simple, fast workflows (e.g., pre-populated student lists).
* **Administrators**:
  + Manage system data (e.g., add teachers, set timetables).
  + Require full access to configure the system.
* **Students** (Future Scope):
  + May view their attendance records.

#### 2.4 Operating Environment

* **Database**: Relational DBMS (e.g., MySQL, PostgreSQL).
* **Application**: Web-based (e.g., PHP, Python) or desktop (e.g., Java).
* **Hardware**: Standard server or local machine with sufficient storage and processing power.

#### 2.5 Design and Implementation Constraints

* Teachers can only access their own subjects and students (enforced via teacher\_id).
* Attendance calculations exclude cancelled sessions (Sessions.status = 'Cancelled').
* Database must support referential integrity and cascading deletes.

#### 2.6 Assumptions and Dependencies

* Teachers have unique email addresses for login.
* Timetables are set per semester and don’t change frequently.
* Session generation and holiday updates are admin-managed or scripted.
* A front-end application will enforce access control and UI logic.

### 3. System Features

#### 3.1 User Management

##### 3.1.1 Teacher Management

* **Description**: Store and manage teacher profiles for authentication and subject assignment.
* **Input**: First name, last name, email, phone (optional), password (hashed).
* **Output**: Unique teacher\_id assigned.
* **Constraints**: Email must be unique; password required.

##### 3.1.2 Student Management

* **Description**: Store and manage student details for enrollment and attendance tracking.
* **Input**: First name, last name, roll number, email, phone (optional).
* **Output**: Unique student\_id assigned.
* **Constraints**: Roll number and email must be unique.

#### 3.2 Subject Management

* **Description**: Define subjects and assign them to teachers.
* **Input**: Subject name, teacher ID.
* **Output**: Unique subject\_id assigned.
* **Constraints**: Each subject linked to one teacher; teacher ID nullable if unassigned.

#### 3.3 Timetable Management

* **Description**: Create and manage a recurring weekly schedule for subjects.
* **Input**: Subject ID, day of week, start time, end time, room (optional), semester start/end dates.
* **Output**: Unique timetable\_id assigned.
* **Constraints**:
  + No overlapping schedules for the same subject (unique constraint on subject\_id, day\_of\_week, start\_time).
  + Start time must be before end time.
  + Semester dates required.

#### 3.4 Session Management

##### 3.4.1 Session Generation

* **Description**: Generate specific session instances from the timetable for a semester.
* **Input**: Timetable ID, date range (derived from semester start/end).
* **Output**: Unique session\_id, date, and initial status (Scheduled).
* **Constraints**: No duplicate sessions for the same timetable entry on the same date.

##### 3.4.2 Session Status Update

* **Description**: Update session status based on occurrence or cancellation.
* **Input**: Session ID, new status (Scheduled, Completed, Cancelled).
* **Output**: Updated status in Sessions table.
* **Constraints**: Only admins or automated scripts (e.g., holiday checks) update status.

#### 3.5 Attendance Management

##### 3.5.1 Record Attendance

* **Description**: Allow teachers to mark attendance for their students in a session.
* **Input**: Session ID, student ID, status (Present, Absent, Late), teacher ID (recorded\_by).
* **Output**: Attendance record with attendance\_id, timestamp.
* **Constraints**:
  + Only the assigned teacher (via Subjects.teacher\_id) can mark attendance.
  + One record per student per session.

##### 3.5.2 Edit Attendance

* **Description**: Allow teachers to modify existing attendance records.
* **Input**: Attendance ID, updated status.
* **Output**: Updated status and timestamp.
* **Constraints**: Restricted to the teacher who recorded it or owns the subject.

##### 3.5.3 Delete Attendance

* **Description**: Allow teachers to remove attendance records (if allowed).
* **Input**: Attendance ID.
* **Output**: Record deleted from Attendance.
* **Constraints**: Restricted to the teacher who recorded it.

##### 3.5.4 Pre-Populate Attendance (Optional)

* **Description**: Automatically create attendance records for all enrolled students when a session is scheduled.
* **Input**: Session ID, enrolled students (via Student\_Subject).
* **Output**: Attendance records with status = NULL.
* **Constraints**: Only for non-cancelled sessions.

#### 3.6 Calendar View

* **Description**: Display a teacher’s schedule in a calendar format (e.g., monthly, weekly).
* **Input**: Teacher ID, date range (e.g., March 2025).
* **Output**: List of sessions with dates, times, subjects, and statuses.
* **Constraints**: Shows only the teacher’s subjects (filtered by Subjects.teacher\_id).

#### 3.7 Holiday/Exception Management

* **Description**: Define holidays or exceptions that cancel sessions.
* **Input**: Date, description (e.g., "Spring Break").
* **Output**: Unique exception\_id in Calendar\_Exceptions.
* **Constraints**: Unique date per exception.

#### 3.8 Attendance Calculation

* **Description**: Calculate student attendance percentage based on conducted sessions.
* **Input**: Student ID, subject ID, date range.
* **Output**: Percentage (e.g., "75%"), present/absent/late counts.
* **Constraints**: Only counts sessions with status = 'Completed'.

#### 3.9 Access Control

* **Description**: Restrict teachers to their own data.
* **Input**: Teacher login credentials (email, password).
* **Output**: Filtered view of subjects, sessions, and students.
* **Constraints**: Enforced via Subjects.teacher\_id in queries.

### 4. Data Requirements

#### 4.1 Database Schema

The AMS uses a relational database with the following tables:

##### 4.1.1 Teachers

sql

CollapseWrapCopy

CREATE TABLE Teachers (

teacher\_id INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

phone VARCHAR(15),

pass VARCHAR(255) NOT NULL

);

* Stores teacher profiles for authentication and assignment.

##### 4.1.2 Students

sql

CollapseWrapCopy

CREATE TABLE Students (

student\_id INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

roll\_number VARCHAR(20) UNIQUE NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

phone VARCHAR(15)

);

* Stores student details for enrollment and tracking.

##### 4.1.3 Subjects

sql

CollapseWrapCopy

CREATE TABLE Subjects (

subject\_id INT PRIMARY KEY AUTO\_INCREMENT,

subject\_name VARCHAR(100) NOT NULL,

teacher\_id INT,

FOREIGN KEY (teacher\_id) REFERENCES Teachers(teacher\_id) ON DELETE SET NULL

);

* Links subjects to teachers.

##### 4.1.4 Timetable

sql

CollapseWrapCopy

CREATE TABLE Timetable (

timetable\_id INT PRIMARY KEY AUTO\_INCREMENT,

subject\_id INT,

day\_of\_week ENUM('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday') NOT NULL,

start\_time TIME NOT NULL,

end\_time TIME NOT NULL,

room VARCHAR(50),

semester\_start\_date DATE NOT NULL,

semester\_end\_date DATE NOT NULL,

UNIQUE (subject\_id, day\_of\_week, start\_time),

CHECK (start\_time < end\_time),

FOREIGN KEY (subject\_id) REFERENCES Subjects(subject\_id) ON DELETE CASCADE

);

* Defines recurring schedules.

##### 4.1.5 Sessions

sql

CollapseWrapCopy

CREATE TABLE Sessions (

session\_id INT PRIMARY KEY AUTO\_INCREMENT,

timetable\_id INT,

date DATE NOT NULL,

status ENUM('Scheduled', 'Completed', 'Cancelled') DEFAULT 'Scheduled',

UNIQUE (timetable\_id, date),

FOREIGN KEY (timetable\_id) REFERENCES Timetable(timetable\_id) ON DELETE CASCADE

);

* Tracks specific class instances.

##### 4.1.6 Attendance

sql

CollapseWrapCopy

CREATE TABLE Attendance (

attendance\_id INT PRIMARY KEY AUTO\_INCREMENT,

student\_id INT,

session\_id INT,

status ENUM('Present', 'Absent', 'Late') DEFAULT NULL,

timestamp DATETIME DEFAULT CURRENT\_TIMESTAMP,

recorded\_by INT,

UNIQUE (student\_id, session\_id),

FOREIGN KEY (student\_id) REFERENCES Students(student\_id) ON DELETE CASCADE,

FOREIGN KEY (session\_id) REFERENCES Sessions(session\_id) ON DELETE CASCADE,

FOREIGN KEY (recorded\_by) REFERENCES Teachers(teacher\_id) ON DELETE SET NULL

);

* Records student attendance.

##### 4.1.7 Student\_Subject

sql

CollapseWrapCopy

CREATE TABLE Student\_Subject (

student\_id INT,

subject\_id INT,

PRIMARY KEY (student\_id, subject\_id),

FOREIGN KEY (student\_id) REFERENCES Students(student\_id) ON DELETE CASCADE,

FOREIGN KEY (subject\_id) REFERENCES Subjects(subject\_id) ON DELETE CASCADE

);

* Manages student enrollment.

##### 4.1.8 Calendar\_Exceptions

sql

CollapseWrapCopy

CREATE TABLE Calendar\_Exceptions (

exception\_id INT PRIMARY KEY AUTO\_INCREMENT,

date DATE NOT NULL,

description VARCHAR(100),

UNIQUE (date)

);

* Tracks holidays/exceptions.

#### 4.2 Relationships

* **Teachers → Subjects**: 1-to-many (one teacher teaches multiple subjects).
* **Subjects → Timetable**: 1-to-many (one subject has multiple schedule entries).
* **Timetable → Sessions**: 1-to-many (one timetable entry generates multiple sessions).
* **Students → Attendance**: 1-to-many (one student has multiple attendance records).
* **Sessions → Attendance**: 1-to-many (one session has multiple attendance records).
* **Students ↔ Subjects**: Many-to-many (via Student\_Subject).

#### 4.3 Data Integrity

* Foreign keys ensure referential integrity.
* Unique constraints prevent duplicates (e.g., roll\_number, email, timetable\_id + date).

### 5. Non-Functional Requirements

#### 5.1 Performance

* **Response Time**: Attendance marking for 100 students in a session should complete within 5 seconds.
* **Session Generation**: Generating sessions for a semester (e.g., 50 timetable entries over 4 months) should take less than 10 seconds.

#### 5.2 Security

* **Authentication**: Teachers log in with email and hashed passwords.
* **Authorization**: Teachers access only their subjects and students (via teacher\_id filtering).
* **Data Protection**: Prevent SQL injection and unauthorized access.

#### 5.3 Reliability

* **Uptime**: 99% availability for the application.
* **Data Consistency**: Cascading deletes maintain integrity (e.g., removing a subject deletes its timetable).

#### 5.4 Usability

* **Interface**: Teachers mark attendance in fewer than 5 clicks per session (e.g., pre-populated list with dropdowns).
* **Learning Curve**: Intuitive design for non-technical users.

#### 5.5 Scalability

* Supports up to 100 teachers, 1000 students, and 200 subjects without schema changes.
* Database handles 10,000 sessions per semester.

#### 5.6 Maintainability

* Modular design allows easy updates (e.g., adding new features like reports).

### 6. Future Scope

* **Student Portal**: Allow students to view their attendance.
* **Reports**: Generate detailed summaries (e.g., by subject, month).
* **Notifications**: Alert teachers of upcoming sessions or holidays.
* **Room Scheduling**: Add conflict checking for rooms.

### 7. Assumptions and Dependencies

* **Assumptions**:
  + Teachers manage one subject at a time per timetable slot.
  + Holidays are manually entered or scripted into Calendar\_Exceptions.
  + Attendance is marked post-session, updating Sessions.status to 'Completed'.
* **Dependencies**:
  + Relational DBMS (e.g., MySQL) for schema implementation.
  + Front-end application to enforce UI and access logic.
  + Hashing library (e.g., bcrypt) for password security.

### 8. Appendix

#### 8.1 Sample Queries

* **Teacher’s Calendar**:

sql

CollapseWrapCopy

SELECT sub.subject\_name, tt.day\_of\_week, tt.start\_time, tt.end\_time, ses.date, ses.status

FROM Teachers t

JOIN Subjects sub ON t.teacher\_id = sub.teacher\_id

JOIN Timetable tt ON sub.subject\_id = tt.subject\_id

LEFT JOIN Sessions ses ON tt.timetable\_id = ses.timetable\_id

WHERE t.teacher\_id = 1 AND ses.date BETWEEN '2025-03-01' AND '2025-03-31';

* **Attendance Marking List**:

sql

CollapseWrapCopy

SELECT s.student\_id, s.first\_name, s.last\_name, s.roll\_number, a.status

FROM Students s

JOIN Student\_Subject ss ON s.student\_id = ss.student\_id

JOIN Subjects sub ON ss.subject\_id = sub.subject\_id

JOIN Timetable tt ON sub.subject\_id = tt.subject\_id

JOIN Sessions ses ON tt.timetable\_id = ses.timetable\_id

LEFT JOIN Attendance a ON s.student\_id = a.student\_id AND a.session\_id = ses.session\_id

WHERE sub.teacher\_id = 1 AND ses.session\_id = 1;

#### 8.2 Assumptions on Implementation

* Session status updates may require a cron job or admin intervention for holidays.