1.Introduction

The **School Management System** is a software project made to help schools manage their daily tasks in a faster and easier way. Schools usually have a lot of work like keeping student records, taking attendance, scheduling classes, handling fees, sending reports, and more. Doing all of this manually takes a lot of time and can lead to mistakes.

This system will help by putting everything in one place. It allows school staff, teachers, students, and parents to access important information online anytime. For example, teachers can mark attendance, parents can check their child's grades, and students can see their class timetable.

The main goal of this project is to make school work more organized, save time, and improve communication between everyone involved.

### Purpose of this Document

### The purpose of this Software Requirements Specification (SRS) document is to define the requirements for the **School Management System (SMS)**. It is intended to serve as a reference for all stakeholders including the development team, testers, administrators, and school staff. This document provides a complete description of the system functionalities, constraints, and interactions with external systems, ensuring a shared understanding of the system’s behavior and goals.

### Scope of this document

· The system helps manage student records like admission, attendance, and performance.

· It allows teachers to mark attendance and enter exam grades.

· Administrators can schedule classes and manage timetables.

· The system handles fee collection and tracks payment status.

· It keeps records of staff and their assigned responsibilities.

· The system generates report cards and progress reports.

· Parents can view their child’s attendance and academic progress.

· It sends important notifications through email or SMS.

· The system manages the library, including books and borrowers.

· It ensures data security and access based on user roles.

1. Overview

This document provides a complete description of the School Management System (SMS), which is designed to automate and simplify daily school operations. It outlines the system’s features, technical requirements, user roles, and module functionalities. The document serves as a guide for developers, school administrators, and end-users to understand how the system works and how it can be used to manage students, teachers, classes, exams, fees, attendance, and communication efficiently. It also includes design details, security considerations, and plans for future improvements.

1. General description

The School Management System is a web-based application developed to manage all aspects of a school’s operations in a centralized and efficient way. It provides tools for handling student admissions, attendance, exams, results, fee payments, and communication between school staff, students, and parents. The system supports different user roles such as administrators, teachers, students, and parents, each with specific access and functionalities. It is designed to reduce manual work, improve data accuracy, and enhance decision-making through real-time information access. The system is scalable, secure, and customizable to meet the needs of various educational institutions.

1. Functional Requirements

**1.Student Management**

Add, update, view, and delete student records.

Manage student enrollment, classes, and academic history.

**2.Teacher/Staff Management**

Add, update, and assign teachers to classes and subjects.

View teacher profiles and manage staff roles.

**3.Attendance Management**

Record daily attendance for students and staff.

Generate attendance reports and notify parents of absences.

**4.Class & Timetable Management**

Create and assign class schedules.

Prevent timetable conflicts (e.g., teacher double-booking).

**5.Examination & Grading**

Create exam schedules and enter marks.

Generate report cards and performance analytics.

**6.Fee Management**

Generate and manage student fee invoices.

Track payments and send reminders for dues.

**7.Library Management** (Optional)

Add books, issue and return tracking.

Manage borrowers and due dates.

**8.User Management**

Define roles: Admin, Teacher, Student, Parent, Librarian.

Manage permissions and access rights.

**9.Communication & Notifications**

Send announcements via SMS or email.

Notify users of events, holidays, exam results, and fee dues.

**10.Reports & Analytics**

Generate reports on attendance, grades, fee status, etc.

Export data in printable formats (PDF, Excel).

1. Interface Requirement

#### ****1.User Interface (UI) Requirements****

* The system must have a **responsive web interface** accessible on desktops, tablets, and mobile devices.
* Each user role (Admin, Teacher, Student, Parent) must see a **customized dashboard** showing relevant features.
* The interface should be **user-friendly**, with intuitive navigation, clear buttons, and input forms.
* All forms must include **validation messages** (e.g., required fields, incorrect data format).
* The UI must provide **search and filter** options for student records, reports, and schedules.
* The system should support **multi-language options** (if required by the school).

#### ****Authentication Interface****

#### A **login page** must allow users to sign in securely with a username/email and password.

#### The system should support **password reset/recovery** via email.

#### **Two-factor authentication (2FA)** may be used for added security (optional/advanced).

#### 3.****Admin Interface****

* Admin users must have access to manage students, staff, fees, exams, and settings.
* A **role management interface** must allow assigning and editing user roles and permissions.

#### 4.****Teacher Interface****

* Teachers should be able to view class schedules, mark attendance, enter grades, and send messages.
* Interface must include a **gradebook** for managing student performance.

#### 5.****Student & Parent Interface****

* Students and parents must be able to view:

1. Timetables
2. Attendance records
3. Exam results
4. Fee payment history

* A **messaging or notification panel** must be included.

#### 6.****Database Interface****

* The system must interact with a **relational database** (e.g., MySQL/PostgreSQL) using secure queries.
* All CRUD (Create, Read, Update, Delete) operations must be supported through backend APIs.

### 5.****Performance Requirements****

**1.Fast Response Time**  
The system should respond to user actions (like login, submitting forms, or loading pages) within **2–3 seconds**.

**2.Support for Multiple Users**  
The system must handle at least **100 users at the same time** without slowing down.

**3.High Availability**  
The system should be available and working **99.9% of the time**, with minimal downtime.

**4.Data Handling Capacity**  
The system should smoothly manage records for **1,000 or more students and staff** without crashing.

**5.Daily Backup**  
The system must automatically **back up data every day** to prevent data loss in case of failure.

### 6.****Design Constraints****

**1.Role-Based Access Control (RBAC)**

The system must strictly control access based on user roles (Admin, Teacher, Student, Parent) to protect sensitive data and ensure proper functionality.

**2.Web-Based System Only**

The application must be fully web-based and accessible through standard browsers; no desktop or mobile app is required in the initial version.

**3.Secure Authentication**

All user passwords must be **encrypted**, and the system must use secure login protocols to protect user credentials.

**4.Technology Stack Limitation**

Developers must use only approved technologies (e.g., **PHP or Python for backend**, **MySQL or PostgreSQL for database**) as per school or organization policy.

**5.Scalability for Limited Users**

The initial system should be designed to handle up to **1,000 users** efficiently, with scalable architecture for future expansion.

### 7.****Non-Functional Attributes****

**1.Security**

The system must protect all user and student data through **secure authentication, encryption**, and **role-based access control**.

**2.Reliability**

The system must work **consistently without crashing or losing data**, especially during critical times like exam results or fee payments.

**3.Usability**

The interface must be **easy to use** for all types of users (admins, teachers, students, parents), requiring minimal training.

**4.Availability**

The system must be accessible **24/7 with at least 99.9% uptime**, so users can access information anytime.

**5.Scalability**

The system should be able to **grow with the school**, supporting more users and data without performance issues.

8.Preliminary Schedule and Budget

|  |  |  |
| --- | --- | --- |
| 1. **Requirement Gathering** | Meet with school admins, define features | 1 week |
| 2. **System Design** | UI mockups, database schema, tech stack setup | 1 week |
| 3. **Backend Development** | Core modules: Student, Teacher, Attendance, Fees | 2–3 weeks |
| 4. **Frontend Development** | Dashboards, responsive design | 2 weeks |
| 5. **Testing & Bug Fixing** | Functional testing, user feedback | 1–2 weeks |
| 6. **Deployment** | Hosting, domain setup, server configuration | 3–5 days |
| 7. **Training & Handover** | Admin training, guides, and support setup | 3–5 days |

Budget

| **Item** | **Estimated Cost (NPR)** |
| --- | --- |
| **Software Development** (local team) | NPR 80,000 – NPR 120,000 |
| **UI/UX Design** | NPR 10,000 – NPR 20,000 |
| **Web Hosting & Server (1 year)** | NPR 5,000 – NPR 12,000 |
| **Domain Registration (.com/.edu.np)** | NPR 0 – NPR 2,000 (edu.np is free) |
| **Testing & QA** | NPR 10,000 – NPR 15,000 |
| **SMS Gateway Integration** | NPR 5,000 – NPR 10,000 |
| **Training & Documentation** | NPR 5,000 – NPR 8,000 |
| **Contingency (10–15%)** | NPR 15,000 – NPR 25,000 |

**💵 Total Estimated Budget: NPR 130,000 – NPR 200,000**

## 9.****Appendices****

### ****Appendices A: Acronyms****

**SMS** – School Management System

**UI** – User Interface

**RBAC** – Role-Based Access Control

**DB** – Database

**API** – Application Programming Interface

### ****Appendices B: User Roles****

**Admin** – Full access to all modules

**Teacher** – Manage classes, attendance, grades

**Student** – View timetable, results, notices

**Parent** – Monitor child’s progress and fees

**Librarian** – Manage library inventory

### ****Appendices C: Technologies Used****

**Frontend**: HTML, CSS, JavaScript

**Backend**: PHP / Python

**Database**: MySQL / PostgreSQL

**Hosting**: Local server or cloud (e.g., DigitalOcean)

**SMS Gateway**: Sparrow SMS (Nepal)