****

**Name \_\_\_\_\_\_Muhammad Riaz**

**Roll No\_\_\_\_\_2K22-BSE238 (Sec (B))**

**Subject\_\_\_\_\_Software Construction & Development**

## Project \_\_\_\_School Fees Management System (SFMS)

**(SRS)**

**Submitted To \_\_\_Mr.M.Akhtar**

**Program \_\_\_\_\_ Software Engineering (BSSE)**

**Department\_\_\_\_\_\_Computer Science (CS)**

**Institute \_\_\_\_\_ NFC-IET-MULTAN**

# Table Of Content

### 1.Introduction…………………………………………………………………….

#### Purpose…………………………………………………………………………………….

#### Scope……………………………………………………………………………………….

#### Definitions, Acronyms, and Abbreviations………………………………………………

#### References…………………………………………………………………………............

#### Overview……………………………………………………………………………..........

### 2. Overall Description………………………………………………………………..

#### 2.1 Product Perspective…………………………………………………………………………

#### 2.2 Product Features………………………………………………………………………........

#### 2.3 User Classes and Characteristics…………………………………………………………..

#### 2.4 Operating Environment……………………………………………………………………..

#### 2.5 Design and Implementation Constraints……………………………………………………

#### 2.6 Assumptions and Dependencies……………………………………………………………

### 3. Specific Requirements…………………………………………………………….

#### 3.1 Functional Requirements…………………………………………………………………….

##### **3.1.1 User Authentication……………………………………………………………………………………………………………**

##### **3.1.2 Fees Management……………………………………………………………………………………………………………..**

##### **3.1.3 Payment Processing……………………………………………………………………………………………………………**

##### **3.1.4 Notifications……………………………………………………………………………………………………………………….**

##### **3.1.5 Reporting……………………………………………………………………………………………………………………………**

#### 3.2 Non-Functional Requirements…………………………………………………………......

**3.2.1 Performance…………………………………………………………………………………………………………………**

**3.2.2 Security………………………………………………………………………………………………………………………..**

**3.2.3 Scalability………………………………………………………………………………………………………………………**

**3.2.4 Usability………………………………………………………………………………………………………………………..**

**3.2.5 Maintainability………………………………………………………………………………………………………………**

#### 4.2 Data Flow Diagram (Level 1)………………………………………………..

### 5. Interface Requirements………………………………………………………..

#### 5.1 User Interfaces……………………………………………………………………………..

#### 5.2 Hardware Interfaces……………………………………………………………………….

#### 5.3 Software Interfaces…………………………………………………………………………

### 6. Other Non-Functional Requirements……………………………………….

#### 6.1 Performance Requirements………………………………………………………………..

#### 6.2 Security Requirements…………………………………………………………………….

#### 6.3 Backup Requirements……………………………………………………………………..

#### 6.4 Compliance Requirements………………………………………………………….

### 7. Appendices………………………………………………………………………………

#### 7.1 Glossary…………………………………………………………………………………...

#### 7.2 References…………………………………………………………………………………

# Software Requirements Specification (SRS)

## School Fees Management System (SFMS)

### 1. Introduction

#### 1.1 Purpose

The purpose of this document is to define the requirements for the School Fees Management System (SFMS), a web-based application developed using PHP. This system aims to automate the process of managing student fees, providing an efficient and user-friendly platform for schools.

#### 1.2 Scope

The SFMS is designed for educational institutions to manage student fee records, payments, and reports. It enables administrators to track fees, notify parents/students about pending dues, and generate financial reports. The system is intended for use in schools of various sizes.

#### 1.3 Definitions, Acronyms, and Abbreviations

* **SFMS**: School Fees Management System
* **SRS**: Software Requirements Specification
* **Admin**: System Administrator
* **User**: School staff, students, or parents interacting with the system

#### 1.4 References

* PHP Documentation: https://www.php.net/docs.php
* MySQL Documentation: https://dev.mysql.com/doc/
* ISO/IEC 25010:2011 for Software Quality Requirements

#### 1.5 Overview

This document outlines the functional and non-functional requirements for the SFMS. It also covers system architecture, user interface design, and constraints.

### 2. Overall Description

#### 2.1 Product Perspective

The SFMS is a standalone system that integrates with the school's existing databases. It is web-based and can be accessed through a browser. The primary stakeholders are administrators, students, and parents.

#### 2.2 Product Features

* **User Management**: Admin can add, update, or delete user accounts.
* **Fees Management**: Manage different fee types (e.g., tuition, library, transportation).
* **Payment Processing**: Track payments and generate receipts.
* **Reports**: Generate detailed reports on fee collection and outstanding dues.
* **Notifications**: Send reminders for pending fees.

#### 2.3 User Classes and Characteristics

* **Admin**: Manages the system and handles all administrative tasks.
* **Parents/Students**: View payment history, pending dues, and make online payments.
* **School Staff**: Operates the system to manage records and generate reports.

#### 2.4 Operating Environment

* **Server**: Apache or Nginx with PHP 8.x and MySQL 8.x.
* **Client**: Any modern web browser (Chrome, Firefox, Safari).
* **Platform**: Compatible with Linux, Windows, and macOS.

#### 2.5 Design and Implementation Constraints

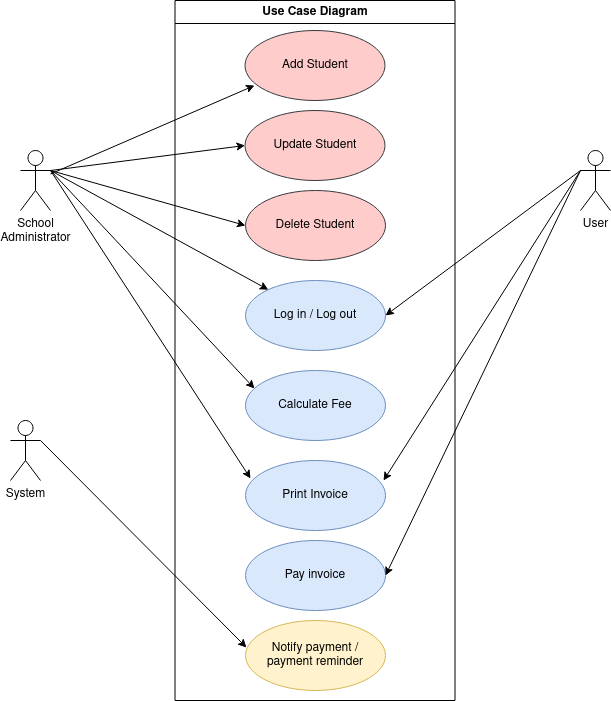
* The system must use PHP for backend development.
* MySQL will be the database engine.
* Payment gateway integration must comply with PCI DSS standards.

#### 2.6 Assumptions and Dependencies

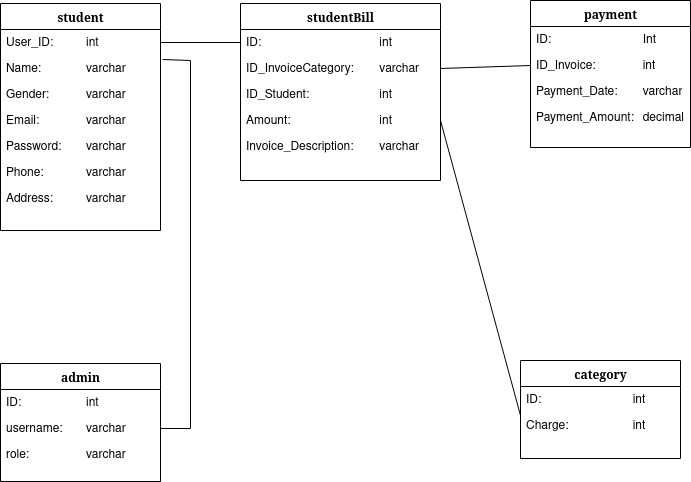
Internet connectivity is required for online payment features.

* Schools will provide necessary student data to populate the system.

Use Case Diagram



### Class Diagram



### 3. Specific Requirements

#### 3.1 Functional Requirements

##### **3.1.1 User Authentication**

* Users must log in with a valid username and password.
* Role-based access control for Admin, Staff, and Students/Parents.

##### **3.1.2 Fees Management**

* Admin can define fee categories and assign them to students.
* View all fee records and payment statuses.
* Update or delete fee records as needed.

##### **3.1.3 Payment Processing**

* Allow online payments through a secure payment gateway.
* Generate and email payment receipts.

##### **3.1.4 Notifications**

* Notify users about pending fees via email or SMS.
* Generate automatic reminders for overdue payments.

##### **3.1.5 Reporting**

* Generate reports for overall fee collection.
* Provide detailed breakdowns by student, class, and fee category.

#### 3.2 Non-Functional Requirements

* **Performance**: Handle up to 1,000 concurrent users.
* **Security**: Implement HTTPS and encrypt sensitive data.
* **Scalability**: Support future integration with other school systems.
* **Usability**: Ensure a user-friendly interface with clear navigation.
* **Maintainability**: Follow coding standards for easy updates.

### 4. System Architecture

#### 4.1 Overview

The SFMS is based on a client-server architecture with the following components:

* **Frontend**: HTML, CSS, JavaScript for user interface.
* **Backend**: PHP scripts to process logic.
* **Database**: MySQL to store user and payment data.

#### 4.2 Data Flow Diagram (Level 1)

1. User inputs credentials on the login page.
2. Backend validates credentials against the database.
3. Upon success, user accesses respective functionalities (fees, payments, reports).
4. Payments are processed via an integrated gateway.

### 5. Interface Requirements

#### 5.1 User Interfaces

* **Login Page**: Username, password fields, and login button.
* **Dashboard**: Summary of fees, notifications, and quick links.
* **Payment Page**: Payment gateway integration.
* **Report Page**: Downloadable reports in PDF and Excel formats.

#### 5.2 Hardware Interfaces

* Server: Minimum 4 GB RAM, 2 CPU cores, and 20 GB storage.
* Client: Any device with a modern web browser.

#### 5.3 Software Interfaces

* PHP 8.x for backend scripting.
* MySQL 8.x for database.
* Payment gateway API for transaction processing.

### 6. Other Non-Functional Requirements

#### 6.1 Performance Requirements

* Page load time must not exceed 3 seconds under normal load.
* The system must support 10,000 users with minimal latency.

#### 6.2 Security Requirements

* Use bcrypt for password hashing.
* Secure communication with HTTPS.

#### 6.3 Backup Requirements

* Automatic daily database backups.
* Retain backups for 30 days.

#### 6.4 Compliance Requirements

* Adhere to GDPR for data privacy.
* Ensure PCI DSS compliance for payment processing.

### 7. Appendices

#### 7.1 Glossary

* **HTTPS**: Hypertext Transfer Protocol Secure
* **PCI DSS**: Payment Card Industry Data Security Standard

#### 7.2 References

* PHP Official Website: https://www.php.net
* MySQL Documentation: https://dev.mysql.com
* Bootstrap for UI: https://getbootstrap.com