SOFTWARE DEVELOPMENT PRACTICE

STUDENT MANAGEMENT SYSTEM

SUBMITTED BY,

SANTHOSH KANNA.M
SHAI SANKAR .A K
SANJAY.J.K
RAGUL.T

1. Introduction

- Purpose of the Document
- Scope of the System
- Document Conventions
- References

2. General Description

- Product Perspective
- Product Features
- User Classes and Characteristics
- Operating Environment
- Design and Implementation Constraints

3. Specific Requirements

3.1 External Interface Requirements

- User Interfaces
- Hardware Interfaces
- Software Interfaces
- Communication Interfaces

3.2 Functional Requirements

- Landing Page Features
- User Login
- Registration
- News and Announcements
- Quick Links
- Search Functionality
- Featured Events
- User Authentication
- User Registration
- News and Announcement Management
- Integration with Other Modules
- Security Requirements

3.3 Performance Requirements

- Response Times
- Scalability
- Availability
- Resource Utilization

3.4 Design Constraints

- Technology Stack
- Browser Compatibility
- Mobile Responsiveneses

4. index

- Applicable Glossary
- Use Case Diagrams (if applicable)
- Wireframes or Mockups (if available)
- Flow Diagrams (if applicable)

5.Appendices

- Glossary
- Reference

1. Introduction

1.1 Purpose of the document

The purpose of this document is to define the requirements for the Student Management System (SMS). This system will provide a comprehensive platform for educational institutions to manage student data, courses, attendance, and grading.

1.2 Scope of the system

.The SMS will cover user registration, authentication, student profile management, course management, attendance tracking, gradebook management, reporting, and administration

1.3 Document convention

This section outlines the conventions and notations used throughout this Software

Requirements Specification (SRS) document to ensure clarity and consistency in presenting the requirements

1.4 References

[List any relevant documents or external sources that were used in creating this SRS.]

2. General description

2.1product perspective

SCIS is a system that will provide detailed class information for both instructors and students. It is a system that can reduce large amounts of work for instructors and also let students check their individual work and give them an idea of how their performance compares to the other students.

2.2 product features

- **loud-enabled Platform.**

- **Student Data Maintenance.**
- **Student Details.**
- **Admission Management.**
- **Attendance Tracking.**
- **Grades Automation.**
- **Assessment Execution.**
- **Student Discipline Record Maintenance.**

2.3 User class and characteristics

```
class User:
  def __init__(self, user_id, username, password, full_name, email, phone_number):
      self.user_id = user_id
      self.username = username
      self.password = password
      self.full_name = full_name
      self.email = email
      self.phone_number = phone_number

  def display_profile(self):
      """Display user profile information."""
      print(f"User ID: {self.user_id}")
      print(f"Username: {self.username}")
      print(f"Full Name: {self.full_name}")
      print(f"Email: {self.email}")
      print(f"Phone Number: {self.phone_number}")

  def update_profile(self, username=None, password=None, full_name=None, email=None, phone_number=None):
      """Update user profile information."""
```

```
        if username is not None:
            self.username = username
        if password is not None:
            self.password = password
        if full_name is not None:
            self.full_name = full_name
        if email is not None:
            self.email = email
        if phone_number is not None:
            self.phone_number = phone_number

class Student(User):
    def __init__(self, student_id, username, password, full_name, email,
phone_number, roll_number, courses):
        super().__init__(user_id=student_id, username=username,
password=password, full_name=full_name, email=email,
phone_number=phone_number)
        self.roll_number = roll_number
        self.courses = courses

    def display_student_info(self):
        """Display student-specific information."""
        self.display_profile()
        print(f"Roll Number: {self.roll_number}")
        print("Courses Enrolled:")
        for course in self.courses:
            print(f"- {course}")

    def enroll_course(self, course):
        """Enroll the student in a course."""
        self.courses.append(course)
```

```
    def drop_course(self, course):
        """Drop a course from the st

udent's enrollment."""
        if course in self.courses:
            self.courses.remove(course)

# Example usage




# Create a student
student = Student()
    student_id=1,
    username="student1",
    password="password123",
    full_name="John Doe",
    email="john.doe@example.com",
    phone_number="123-456-7890
```

2.4 operating environment

The operating environment for a Student Management System (SMS) can vary depending on factors such as the specific requirements of the system, the technology stack chosen for development, and the scale of the institution or organization using the system.

2.5 design and implementation constraints

Designing and implementing a Student Management System (SMS) comes with various constraints that you need to consider to ensure the system's success, security, and usability. Here are some common design and implementation constraints for a Student Management System:

3. Special requirements

3.1 External interface requirements

External interface requirements for a Student Management System (SMS) are essential for ensuring the system can interact with various external entities and systems effectively. These interfaces allow the SMS to exchange data and information with other systems, users, and stakeholders. Here are some common external interface requirements and their types for a Student Management System:

User interface:

- **Web Interface: The SMS should have a user-friendly web interface accessible through standard web browsers. It should support multiple browsers and devices.**
- **Mobile Application: A mobile app interface for students, teachers, and administrators to access the system from smartphones and tablets.**

3.2 Functional requirements

The functional requirements of this system are:

- **Register new students.**
- **Record the attendance of students.**
- **Record the internal marks of students.**
- **Record the feed details of students.**

- **Register a new teacher/employee.**
- **Register a new user for the system.**
- **Record the salary details of employees.**
- **Record the course details and subject information.**
- **Record the scholarship details and information.**
- **Generate various reports for all transactions in the system3. Login: The existing customer can log in using the registered email**
- **id and password.**
- **a. Ids:**
- **Email**
- **Password**
- **LoginButton**
- **RegisterLink**
- **LoginBox**

3.3 performance requirements

Student performance requirements refer to the expectations and standards that students are expected to meet in their educational pursuits. These requirements can vary depending on the level of education (e.g., elementary school, high school, college, graduate school) and the specific course or program of study. Here are some common
elements of student performance requirements

3.4 Design constraints

Design constraints for students refer to limitations or restrictions that students may encounter when working on various projects or tasks. These constraints can

impact their ability to complete assignments, projects, or other educational activities effectively. Here are some common design constraints for students.

1. **Time constraints**
2. **Resource constraints**
3. **Financial constaints**

4. Index

Creating an index for students typically refers to a system or database used to organize and manage information about students in an educational institution. This index can include various details about each student, such as their personal information, academic records, attendance, and more. Here's a basic outline of what you might include in a student index:

Student Information:

Full Name

Date of Birth

Gender

Contact Information (Address, Phone Number, Email)

actors, use cases, and the relationships between them. Here's a simplified example of such a diagram:

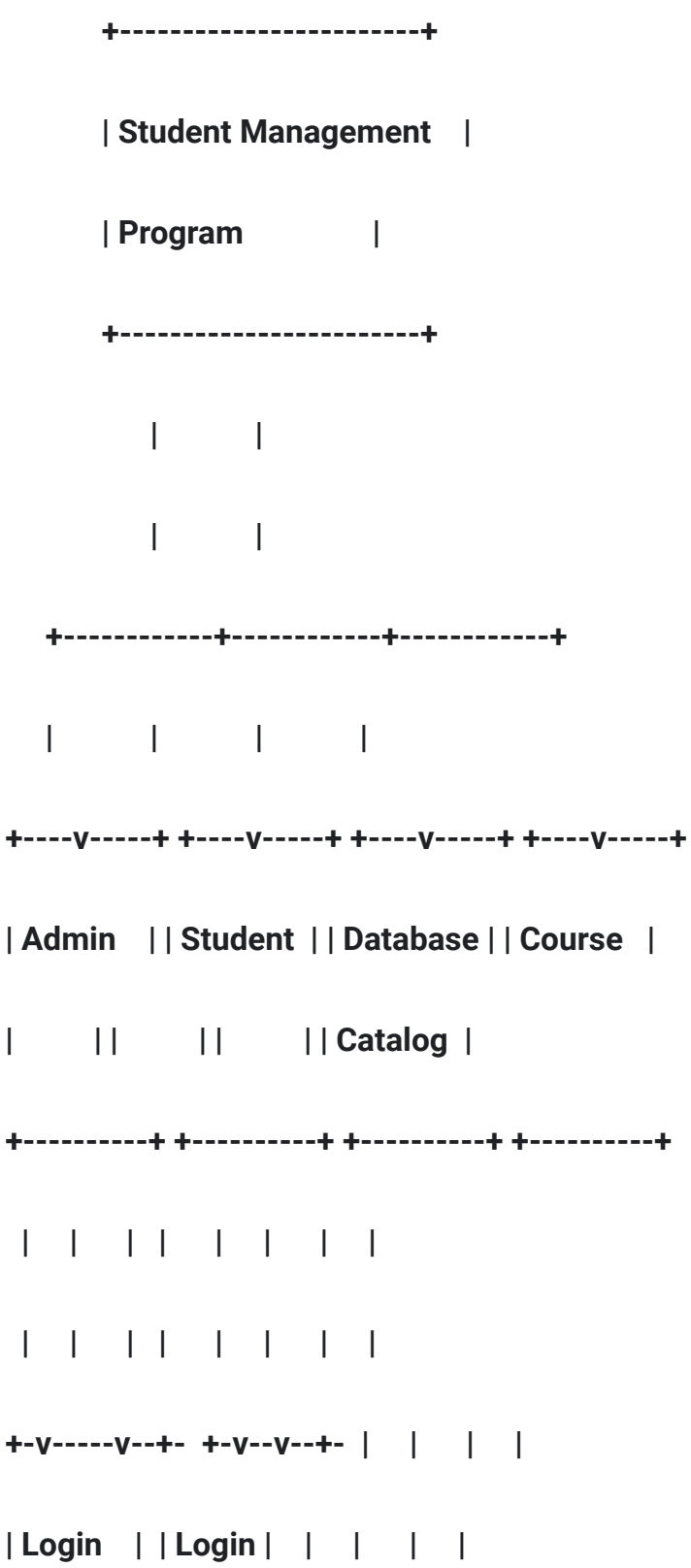
Actors:

- **Admin: The administrator or staff responsible for managing the student information system.**
- **Student: The primary user of the system who interacts with it to access their information.**

Use Cases:

- **Login: Both Admin and Student can log in to the system.**
- **View Student Information: Both Admin and Student can view student details.**
- **Add Student: Admin can add a new student to the system.**
- **Edit Student Information: Admin can edit student information (e.g., update contact details).**
- **Delete Student: Admin can remove a student record from the system.**
- **Generate Reports: Admin can generate various reports like attendance, grades, or student lists.**
- **Enroll in Courses: Students can enroll in courses.**
- **Drop Courses: Students can drop courses they have enrolled in.**
- **View Schedule: Students can view their class schedules.**

Search Courses: Students can search for available courses.



| View | | View | | | | | |

| Add | | Enroll| | | | | |

| Edit | | Drop | | | | | |

| Delete | | View | | | | | |

| Reports | | Schedule| | | | | |

| | | Search | | | | | |

+----------+ +--------+ +-----+ +-----+

5. Appendices

5.1 Glossary

1.Enrollment: The process of students registering for courses.

2.Schedule Conflict: A situation where a student's course schedule overlaps with another course.

5.2 Reference

1.This SRS provides a detailed overview of the functional and non-functional requirements for the Student Management System.

2. It should serve as a guide for development, testing, and validation of the system to ensure it meets the needs of the educational institution and its users.

STUDENT MANAGEMENT SYSTEM