**Software Requirements Specification (SRS)**

**Real-Time University Management System(UMS)**

**Table of Contents**

1. Introduction
   * 1.1 Purpose
   * 1.2 Scope
   * 1.3 Definitions, Acronyms, and Abbreviations
   * 1.4 References
   * 1.5 Overview
2. Overall Description
   * 2.1 Product Perspective
   * 2.2 Product Functions
   * 2.3 User Classes and Characteristics
   * 2.4 Operating Environment
   * 2.5 Design and Implementation Constraints
3. Specific Requirements
   * 3.1 Functional Requirements
   * 3.2 Non-Functional Requirements
   * 3.3 Use Cases
4. Appendices

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to provide a detailed description of the Real-Time University Management System (UMS) that operates in both console-based and graphical user interface (GUI) environments. This document outlines the system's functionalities, requirements, and constraints, serving as a guideline for developers and stakeholders.

**1.2 Scope**

The Real-Time University Management System is designed to manage university operations efficiently, including student records, course management, faculty management, attendance tracking, examination, and grading. This system will be developed in two versions: console-based and GUI-based.

**1.3 Definitions, Acronyms, and Abbreviations**

* **UMS:** University Management System
* **GUI:** Graphical User Interface
* **SRS:** Software Requirements Specification
* **DB:** Database

**1.4 References**

* IEEE Standard 830-1998: IEEE Recommended Practice for Software Requirements Specifications

**1.5 Overview**

This document is structured to provide a high-level overview of the system requirements followed by detailed specifications of functionalities and constraints.

**2. Overall Description**

**2.1 Product Perspective**

The UMS can function as a standalone application that uses a SQLite database for data persistence. Both versions share the same backend logic but differ in user interaction.

**2.2 Product Functions**

Key functionalities of the UMS include:

* Student Management
* Course Management
* Faculty Management
* Attendance Management
* Examination and Grading System
* Reporting and Administration

**2.3 User Classes and Characteristics**

* **Administrators:** Manage all aspects of the system, including user roles, courses, faculty, and reports.
* **Faculty:** Can add, view, and manage courses, as well as attendance.
* **Students:** Can view their records, grades, and attendance.

**2.4 Operating Environment**

* **Console-Based Version:** Runs in a terminal or command-line interface.
* **GUI-Based Version:** Runs on Windows/Linux with a graphical interface using Tkinter.

**2.5 Design and Implementation Constraints**

* Use of SQLite for data persistence.
* Python as the primary programming language.
* GUI must be developed using Tkinter.

**3. Specific Requirements**

**3.1 Functional Requirements**

**3.1.1 Student Management**

* Add, update, delete, and view student profiles.
* Track attendance and academic performance.

**3.1.2 Course Management**

* Create, modify, or delete course records.
* Assign faculty to courses.
* Enroll or remove students from courses.

**3.1.3 Faculty Management**

* Add or update faculty records.
* Track faculty performance and generate reports.

**3.1.4 Attendance Management**

* Add, view, and manage attendance records.

**3.1.5 Examination and Grading**

* Record and update exam results and grades.
* Calculate GPAs and generate performance reports.

**3.1.6 Reporting and Administration**

* Generate detailed reports on various metrics.
* Update tuition fees, scholarships, and financial records.

**3.2 Non-Functional Requirements**

* **Performance:** System should handle a minimum of 100 concurrent users.
* **Reliability:** System should maintain data integrity and consistency.
* **Usability:** The GUI version must be intuitive and easy to navigate.
* **Security:** Data should be securely stored with user role-based access.

**3.3 Use Cases**

**Use Case 1: Add Student**

* **Actors:** Administrator, Faculty
* **Preconditions:** User is logged in.
* **Postconditions:** A new student record is created in the database.

**Use Case 2: View Students**

* **Actors:** Administrator, Faculty, Students
* **Preconditions:** User is logged in.
* **Postconditions:** A list of students is displayed.

**Use Case 3: Add Course**

* **Actors:** Administrator, Faculty
* **Preconditions:** User is logged in.
* **Postconditions:** A new course record is created in the database.

**Use Case 4: View Courses**

* **Actors:** Administrator, Faculty
* **Preconditions:** User is logged in.
* **Postconditions:** A list of courses is displayed.

**Use Case 5: Add Attendance**

* **Actors:** Administrator, Faculty
* **Preconditions:** User is logged in.
* **Postconditions:** A new attendance record is created in the database.

**4. Appendices**

**Appendix A: Sample Data**

Sample data for initial database setup includes:

* **Students:** 8 entries with names, addresses, and grades.
* **Courses:** 8 entries with names and credits.
* **Faculty:** 8 entries with names and subjects.
* **Attendance:** Sample entries correlating students to courses.