Software Requirements Specification (SRS)

Student Time Regulation & Management System (STRMS)

Version: 1.0
Date: November 18, 2025

---

1. Introduction

1.1 Purpose

This document outlines the comprehensive requirements for the Student Time Regulation & Management System (STRMS), a web-based platform designed to help students effectively manage their academic schedules, track progress, and organize tasks through an intuitive interface.

1.2 Project Scope

STRMS provides students with a centralized platform featuring personalized dashboards, interactive calendar management, workspace organization, and cross-device compatibility. The system ensures data security through user authentication and offers responsive design for both mobile and desktop access.

1.3 Definitions and Acronyms

- **STRMS:** Student Time Regulation & Management System
- **UI:** User Interface
- **UX:** User Experience
- **API:** Application Programming Interface
- **CRUD:** Create, Read, Update, Delete operations
- **SPA:** Single Page Application

1.4 References

- UI Mockups: Dashboard (Screenshot 2025-11-18 013130.png), Navigation (image.png), Workspace (Screenshot 2025-11-18 014210.png)
- Approved project architecture and file structure

---

2. Overall Description

2.1 Product Perspective

STRMS operates as an independent web application that integrates frontend interfaces with backend services and database management, creating a cohesive ecosystem for student time management.

2.2 User Classes and Characteristics

- **Students:** Primary users who manage academic schedules, track progress, and organize tasks
- **Administrators:** System maintainers (future scope)

2.3 Operating Environment

- **Client-Side:** Modern browsers (Chrome 90+, Firefox 88+, Safari 14+, Edge 90+)
- **Mobile Support:** iOS Safari, Android Chrome, responsive design
- **Server-Side:** Apache/Nginx with PHP 8.0+, MySQL 8.0+
- **Processing:** C++17 compatible environment

2.4 Design and Implementation Constraints

- 9-day development timeline to final deployment
- Vanilla JavaScript frontend (no frameworks)
- PHP backend with MySQL database
- C++ integration for computational processing
- Mobile-first responsive design approach

2.5 Assumptions and Dependencies

- Users have reliable internet access
- Modern web browsers with JavaScript enabled
- PHP and MySQL available on hosting environment
- C++ compiler for backend processing modules

---

3. System Features

3.1 User Authentication System

3.1.1 Description

Secure user registration and login system protecting personal academic data and providing personalized experiences.

3.1.2 Functional Requirements

- **FR-AUTH-001:** User Registration
- **FR-AUTH-002:** User Login Authentication
- **FR-AUTH-003:** Session Management
- **FR-AUTH-004:** Logout Functionality
- **FR-AUTH-005:** Password Security

3.1.3 Function Flow Description

1. **Registration Process**
    - User accesses registration form
    - System validates input data (email, password strength)
    - System hashes password and creates user account
    - System redirects to login page upon success
2. **Login Process**
    - User enters credentials on login page
    - System verifies credentials against database
    - System creates secure session upon authentication
    - System redirects to dashboard upon success
3. **Session Management**
    - System maintains user session across page navigation
    - System validates session for protected routes
    - System automatically logs out after prolonged inactivity

3.2 Responsive Dashboard

3.2.1 Description

Central hub displaying timeline overview, upcoming activities, and calendar preview, optimized for both mobile and desktop viewing.

3.2.2 Functional Requirements

- **FR-DASH-001:** Timeline Display
- **FR-DASH-002:** Activity Status Overview
- **FR-DASH-003:** Calendar Integration Preview
- **FR-DASH-004:** Responsive Layout Adaptation
- **FR-DASH-005:** Real-time Data Updates

3.2.3 Function Flow Description

1. **Dashboard Initialization**
    - System authenticates user session
    - System fetches user-specific data from APIs
    - System renders timeline with upcoming deadlines
    - System displays calendar preview with current month
2. **Timeline Management**
    - System retrieves pending activities from database
    - System categorizes activities by priority and due date
    - System updates activity status in real-time
    - System displays "No activities" message when appropriate
3. **Responsive Behavior**
    - System detects device screen size on load and resize
    - System adapts layout from mobile to desktop breakpoints
    - System optimizes touch interactions for mobile devices

3.3 Interactive Calendar Management

3.3.1 Description

Comprehensive calendar system allowing students to schedule, view, and manage academic events and deadlines across multiple views.

3.3.2 Functional Requirements

- **FR-CAL-001:** Monthly Calendar View
- **FR-CAL-002:** Event Creation and Editing
- **FR-CAL-003:** Event Display and Management
- **FR-CAL-004:** Date Navigation
- **FR-CAL-005:** Event Persistence

3.3.3 Function Flow Description

1. **Calendar Rendering**
    - System generates monthly grid with proper date alignment
    - System highlights current date and user events
    - System displays event indicators on relevant dates
    - System handles month/year navigation
2. **Event Management**
    - User selects date to view/add events
    - System presents event creation form modal
    - User inputs event details (title, time, description)
    - System validates and saves event to database
    - System updates calendar display with new event
3. **Event Interactions**
    - User clicks event to view details
    - System provides edit/delete options for user-owned events
    - System confirms destructive actions (delete)
    - System synchronizes changes across all views

3.4 Workspace Management

3.4.1 Description

Task and project organization system replacing traditional course management with flexible workspace concept for academic work tracking.

3.4.2 Functional Requirements

- **FR-WRK-001:** Workspace Task Display
- **FR-WRK-002:** Task Creation and Modification
- **FR-WRK-003:** Progress Tracking
- **FR-WRK-004:** Task Organization
- **FR-WRK-005:** Search and Filter Capabilities

3.4.3 Function Flow Description

1. **Workspace Initialization**
    - System loads user's tasks and projects
    - System renders task cards with progress indicators
    - System applies default sorting and filtering
    - System displays empty state if no tasks exist
2. **Task Lifecycle Management**
    - User creates new task with title, description, progress
    - System generates task card with edit/delete controls
    - User updates progress via interactive controls
    - System persists all changes to backend
    - User can archive or delete completed tasks
3. **Workspace Organization**
    - System provides sorting options (name, progress, date)
    - User can search tasks by title or description
    - System filters tasks based on user criteria
    - Layout adapts between grid and list views based on screen size

3.5 Navigation System

3.5.1 Description

Consistent navigation framework providing seamless movement between application sections while maintaining user context and session state.

3.5.2 Functional Requirements

- **FR-NAV-001:** Main Navigation Menu
- **FR-NAV-002:** Responsive Navigation Adaptation
- **FR-NAV-003:** Active State Indication
- **FR-NAV-004:** Mobile Hamburger Menu
- **FR-NAV-005:** Breadcrumb Navigation

3.5.3 Function Flow Description

1. **Navigation Rendering**
    - System loads navigation component on all pages
    - System highlights current page in navigation
    - System adapts layout based on device type
    - Mobile devices show collapsed hamburger menu
2. **Navigation Interactions**
    - User clicks navigation items to switch sections
    - System loads requested page via AJAX or redirect
    - Mobile menu expands/collapses on hamburger click
    - System maintains session state during navigation
3. **Responsive Behavior**
    - System transitions between mobile and desktop layouts
    - Touch-friendly interfaces for mobile navigation
    - Smooth animations for menu transitions
    - Accessible keyboard navigation support

---

4. External Interface Requirements

4.1 User Interfaces

- Responsive design with mobile-first approach
- Consistent color scheme and typography
- Intuitive iconography and visual hierarchy
- Accessible contrast ratios and keyboard navigation
- Loading states and user feedback mechanisms

4.2 Hardware Interfaces

- Touch interface support for mobile devices
- Mouse and keyboard support for desktop
- Print-friendly styles for calendar and schedules

4.3 Software Interfaces

- **Frontend-Backend:** RESTful API communication using JSON
- **Database:** MySQL with PDO connections
- **Processing:** C++ binary execution via PHP system calls
- **Session Management:** PHP native sessions with security enhancements

4.4 Communication Interfaces

- HTTP/HTTPS for web communication
- AJAX/Fetch API for asynchronous data exchange
- JSON format for data serialization
- CORS configuration for cross-origin requests

---

5. Non-Functional Requirements

5.1 Performance Requirements

- Page load time under 3 seconds on 3G networks

- API response time under 500ms for 95% of requests
- Support for 100+ concurrent users
- Calendar rendering complete within 2 seconds

5.2 Security Requirements

- Password hashing using bcrypt algorithm
- SQL injection prevention through prepared statements
- XSS protection through output encoding
- CSRF protection for state-changing operations
- Session timeout after 30 minutes of inactivity

5.3 Reliability Requirements

- 99% uptime during academic periods
- Graceful error handling and user feedback
- Data backup procedures for user content
- Transaction rollback for failed operations

5.4 Usability Requirements

- Intuitive navigation learned within 5 minutes
- Mobile touch targets minimum 44x44 pixels
- WCAG 2.1 AA compliance for accessibility
- Consistent terminology across all interfaces

5.5 Compatibility Requirements

- Support for latest 2 versions of major browsers
- Responsive design from 320px mobile to 1920px desktop
- Progressive enhancement for JavaScript-disabled environments

---

6. System Architecture

6.1 High-Level Architecture

text

Client Layer (Frontend) → API Layer (PHP) → Data Layer (MySQL) → Processing Layer (C++)

6.2 Component Responsibilities

Frontend Components

- **HTML Pages:** Structure and semantic markup
- **CSS Modules:** Presentation and responsive behavior
- **JavaScript Modules:** Interactivity and business logic

Backend Components

- **PHP API Endpoints:** Request handling and data processing
- **C++ Processor:** Computational analysis and optimization
- **Database Schema:** Data persistence and relationships

6.3 Data Flow Description

1. **User Action Initiation**
    - Frontend captures user input/action
    - JavaScript validates input client-side
    - API request formulated with proper parameters
2. **Backend Processing**
    - PHP endpoint receives and sanitizes request
    - Business logic executed with database interaction
    - C++ processor invoked for complex computations
    - Response formatted and returned to frontend
3. **Frontend Update**
    - JavaScript processes API response
    - UI updated to reflect changes
    - User receives success/error feedback
    - Local state synchronized with server

---

7. Appendix

7.1 Database Schema Overview

- **users:** User accounts and authentication data
- **sessions:** Active user sessions and security tokens
- **tasks:** Workspace tasks and progress tracking
- **events:** Calendar events and scheduling data
- **user_preferences:** Interface customization settings

7.2 API Endpoint Summary

- **Authentication:** /api/auth/login, /api/auth/register
- **Tasks:** /api/tasks/get, /api/tasks/save, /api/tasks/delete
- **Events:** /api/events/get, /api/events/save, /api/events/delete

7.3 C++ Processing Modules

- **task_analyzer.cpp:** Academic workload analysis and optimization suggestions
- **Future modules:** Schedule conflict detection, time allocation optimization

---

Document Approval

Role	Name	Signature	Date
Project Sponsor			
Development Lead			
Quality Assurance			

Revision History

Version	Date	Author	Changes
1.0	2025-11-18	Initial Draft	Complete SRS document creation