**Software Requirements Specification (SRS)**

**ERP System for Educational Institution**

**Date: August 2025**

**1. Introduction**

**1.1 Purpose**

This Software Requirements Specification (SRS) document provides a comprehensive description of the Educational ERP (Enterprise Resource Planning) system to be developed. The system integrates three core modules: Academic, Marketing & Finance, and Administration & Human Resources, designed to streamline operations for educational institutions.

**1.2 Scope**

The ERP system will serve educational institutions by providing:

* Centralized data management across all departments
* Role-based access control for different user types
* Real-time reporting and analytics
* Automated workflow processes
* Integration capabilities with third-party systems

**1.3 Definitions and Abbreviations**

* **ERP**: Enterprise Resource Planning
* **CRUD**: Create, Read, Update, Delete
* **API**: Application Programming Interface
* **SRS**: Software Requirements Specification
* **UI**: User Interface
* **HR**: Human Resources
* **ROI**: Return on Investment

**2. Overall Description**

**2.1 Product Perspective**

The ERP system is a web-based application that will replace manual processes and disparate systems currently used by educational institutions. It consists of three interconnected modules sharing a common database and user management system.

**2.2 Product Functions**

* User authentication and authorization
* Academic management (courses, grades, schedules, attendance)
* Financial management (fees, expenses, budgets)
* Human resource management (employees, payroll, performance)
* Marketing campaign management
* Reporting and analytics across all modules
* Communication and notification system

**2.3 User Classes**

1. **System Administrator**: Full system access and configuration
2. **Academic Staff**: Faculty members and academic administrators
3. **Students**: Access to academic records and fee information
4. **HR Personnel**: Employee and administrative management
5. **Finance Staff**: Financial operations and reporting
6. **Marketing Team**: Campaign management and analytics

**3. System Features and Functional Requirements**

**3.1 User Authentication System**

**Priority**: High **Description**: Secure user access control system

**Functional Requirements:**

* **FR-1.1**: System shall provide role-based access control
* **FR-1.2**: System shall implement secure password hashing
* **FR-1.3**: System shall support multi-factor authentication
* **FR-1.4**: System shall provide session management
* **FR-1.5**: System shall send email notifications for account activities
* **FR-1.6**: System shall provide password recovery functionality
* **FR-1.7**: System shall log all authentication attempts
* **FR-1.8**: System shall enforce password complexity policies

**3.2 Database Management**

**Priority**: High **Description**: Centralized data storage and management

**Functional Requirements:**

* **FR-2.1**: System shall use relational database (MySQL/PostgreSQL)
* **FR-2.2**: System shall ensure data normalization and integrity
* **FR-2.3**: System shall provide automated backup functionality
* **FR-2.4**: System shall support data recovery operations
* **FR-2.5**: System shall maintain audit trails for data changes
* **FR-2.6**: System shall provide data encryption at rest

**3.3 API Integration**

**Priority**: Medium **Description**: RESTful APIs for system integration

**Functional Requirements:**

* **FR-3.1**: System shall provide RESTful API endpoints
* **FR-3.2**: System shall support third-party integrations
* **FR-3.3**: System shall implement API authentication and authorization
* **FR-3.4**: System shall provide API documentation
* **FR-3.5**: System shall support webhook notifications
* **FR-3.6**: System shall implement rate limiting for API calls

**3.4 Communication System**

**Priority**: Medium **Description**: Internal communication and notifications

**Functional Requirements:**

* **FR-4.1**: System shall provide email notification service
* **FR-4.2**: System shall support in-app messaging
* **FR-4.3**: System shall implement community chat functionality
* **FR-4.4**: System shall provide AI chatbot assistance
* **FR-4.5**: System shall support push notifications
* **FR-4.6**: System shall maintain message history

**4. Non-Functional Requirements**

**4.1 Performance Requirements**

* **NFR-1**: System shall respond to user requests within 3 seconds under normal load
* **NFR-2**: System shall support up to 1000 concurrent users
* **NFR-3**: System shall achieve 99.5% uptime availability
* **NFR-4**: Database queries shall execute within 2 seconds

**4.2 Security Requirements**

* **NFR-5**: System shall implement HTTPS encryption for all communications
* **NFR-6**: System shall comply with data protection regulations (GDPR)
* **NFR-7**: System shall implement SQL injection protection
* **NFR-8**: System shall provide secure file upload functionality
* **NFR-9**: System shall implement session timeout after 30 minutes of inactivity

**4.3 Usability Requirements**

* **NFR-10**: System shall provide responsive design for mobile devices
* **NFR-11**: System shall support multiple languages
* **NFR-12**: System shall provide accessibility compliance (WCAG 2.1)
* **NFR-13**: System shall provide consistent UI/UX across all modules

**4.4 Reliability Requirements**

* **NFR-14**: System shall provide automated daily backups
* **NFR-15**: System shall implement error logging and monitoring
* **NFR-16**: System shall provide graceful error handling
* **NFR-17**: System shall support disaster recovery procedures

**5. System Architecture**

**5.1 High-Level Architecture**

The system follows a three-tier architecture:

* **Presentation Layer**: Web-based user interface
* **Business Logic Layer**: Application server handling business rules
* **Data Layer**: Relational database system

**5.2 Technology Stack**

* **Frontend**: React.js/Angular with responsive design
* **Backend**: Node.js/Python Django with RESTful APIs
* **Database**: MySQL/PostgreSQL with replication
* **Authentication**: JWT tokens with OAuth2 support
* **Deployment**: Docker containers with cloud hosting

**6. Use Case Diagram**

ERP System Use Cases

System Admin -----> Manage Users

-----> Configure System

-----> Generate Reports

-----> Backup/Restore Data

Academic Staff ---> Manage Courses

---> Track Attendance

---> Grade Students

---> Schedule Exams

Student ---------> View Grades

---> Check Attendance

---> Pay Fees

---> Access Course Materials

---> View Events

---> View Financial Reports

HR Personnel ----> Manage Employees

---> Process Payroll

---> Track Leave

---> Performance Review

Finance Staff ---> Manage Invoices

---> Track Payments

---> Generate Financial Reports

---> Budget Planning

Marketing Team --> Campaign Management

--> Lead Tracking

--> ROI Analysis

--> Event Management

**7. Class Diagram (High-Level)**

User

├── userId: String

├── username: String

├── email: String

├── password: String (hashed)

├── role: UserRole

├── createdAt: DateTime

├── lastLogin: DateTime

└── methods: login(), logout(), updateProfile()

UserRole (Enum)

├── ADMIN

├── ACADEMIC\_STAFF

├── STUDENT

├── HR\_PERSONNEL

├── FINANCE\_STAFF

└── MARKETING\_TEAM

DatabaseEntity (Abstract)

├── id: Long

├── createdAt: DateTime

├── updatedAt: DateTime

└── isActive: Boolean

AuditLog

├── logId: String

├── userId: String

├── action: String

├── entityType: String

├── entityId: String

├── oldValue: JSON

├── newValue: JSON

└── timestamp: DateTime

Notification

├── notificationId: String

├── userId: String

├── title: String

├── message: String

├── type: NotificationType

├── isRead: Boolean

└── sentAt: DateTime

**8. Sequence Diagram - User Authentication**

User -> LoginPage: Enter credentials

LoginPage -> AuthController: POST /api/auth/login

AuthController -> UserService: validateCredentials()

UserService -> Database: SELECT user WHERE username/email

Database -> UserService: User record

UserService -> AuthController: User validated

AuthController -> TokenService: generateJWT()

TokenService -> AuthController: JWT token

AuthController -> LoginPage: Login success + token

LoginPage -> Dashboard: Redirect with token

Dashboard -> AuthController: Verify token

AuthController -> Dashboard: Token valid

Dashboard -> User: Display dashboard

**9. Activity Diagram - System Backup Process**

Start

↓

Check Backup Schedule

↓

Initialize Backup Process

↓

Lock Database (Read-only)

↓

Export Database Schema

↓

Export Database Data

↓

Generate File Checksums

↓

Upload to Backup Storage

↓

Verify Upload Integrity

↓

Release Database Lock

↓

Log Backup Status

↓

Send Notification

↓

End

**10. Data Requirements**

**10.1 Database Schema**

* Normalized relational database structure
* Foreign key constraints for data integrity
* Indexes for performance optimization
* Views for complex queries
* Stored procedures for business logic

**10.2 Data Storage**

* Estimated 50GB initial storage requirement
* Growth rate: 10GB per year
* Backup retention: 30 days daily, 12 months monthly
* File storage for documents and media

**11. External Interface Requirements**

**11.1 User Interfaces**

* Responsive web interface supporting desktop and mobile
* Dashboard views customized by user role
* Form validation and error messaging
* Accessibility features for disabled users

**11.2 Hardware Interfaces**

* Standard web browsers (Chrome, Firefox, Safari, Edge)
* Mobile devices (iOS, Android)
* Printer support for reports and documents

**11.3 Software Interfaces**

* Email server integration (SMTP)
* Payment gateway APIs
* Single Sign-On (SSO) integration
* Learning Management System integration

**11.4 Communication Interfaces**

* HTTPS protocols for secure communication
* WebSocket for real-time notifications
* RESTful APIs for third-party integration
* SMTP for email communications

**12. Quality Assurance**

**12.1 Testing Requirements**

* Unit testing coverage: 85%
* Integration testing for all modules
* Performance testing under load
* Security vulnerability testing
* User acceptance testing

**12.2 Documentation Requirements**

* API documentation (**OpenAPI/Swagger**)
* User manuals for each role
* Administrator setup guide
* Developer documentation
* Deployment guide

**13. Constraints and Assumptions**

**13.1 Design Constraints**

* Must support modern web browsers
* Database must be SQL-compliant
* Must comply with educational data privacy laws
* Integration with existing university systems

**13.2 Assumptions**

* Stable internet connectivity
* Regular system maintenance windows available
* User training will be provided
* Third-party services remain available

**14. Acceptance Criteria**

**14.1 Functional Acceptance**

* All specified features implemented and tested
* User roles and permissions working correctly
* Data integrity maintained across operations
* Reports generated accurately

**14.2 Performance Acceptance**

* Response times meet specified requirements
* System stable under concurrent user load
* Backup and recovery procedures validated
* Security requirements verified

**14.3 Usability Acceptance**

* User interface intuitive and accessible
* Training materials comprehensive
* Error messages clear and helpful
* Mobile responsiveness verified