**Software Requirements Specification (SRS) Document**

**1. Introduction**

**1.1 Purpose**

This document outlines the software requirements for the Multi-Community Admission & Enrollment System. The system allows a Super Admin to manage multiple communities, with each community having its own director. Schools (typically parents) can apply for membership in one or more communities, enroll students, and manage student details. The platform integrates a CMS for community customization and supports payments via Helcim or Plaid.

**1.2 Scope**

* **Multi-Community Management**: Super Admin can sell, resell, and manage all communities.
* **Community Customization**: Community Directors can modify CMS settings, including logos, themes, and website content / product / courses.
* **Enrollment System**: Schools (parents) apply for membership in one or more communities and enroll students.
* **Product Management**: Communities can create and manage various products (courses, materials, services, etc.).
* **Payments**: Transactions processed through Helcim and Plaid, with no automated recurring payments.
* **Authentication**: Implement email-based authentication featuring login links that remain valid for over one week, Additionally, include an option for seamless login using Google accounts.
* **Role-Based Access Control (RBAC)**: Defined roles for Super Admin, Community Directors, and Schools.
* **Reporting & Analytics**: Dashboards for tracking applications, enrollments, and payments.

**2. Functional Requirements**

**2.1 User Roles & Access Control**

1. **Super Admin**
   * Create, manage, and delete communities.
   * Assign and manage Community Directors.
   * Set up payment gateway configurations.
   * View all transactions, applications, and reports.
   * Manage user permissions.
2. **Community Director**
   * Customize community branding (logo, theme, content).
   * Manage applications from Schools.
   * Approve/reject student enrollments.
   * Create and manage products (courses, services etc.).
   * View and validate payments.
3. **School (Parent)**
   * Apply to join one or more communities.
   * Enroll and manage student details.
   * Select and purchase products.
   * View payment history.

**2.2 Community Management**

* Each community operates independently under the Super Admin’s oversight.
* Community Directors can modify CMS settings (logo, theme, pages, forms).
* Each community has its own set of products/courses.
* Mighty Arrows products are available across all communities.

**2.3 Enrollment & Applications**

* Schools submit applications to join communities.
* Applications are reviewed and approved/rejected by Community Directors.
* Enrollment involves structured sections:
  1. School Information
  2. Community-Specific Questions
  3. Student Enrollment Details

**2.4 Products & Course Management**

* Communities can create various types of products (courses, books, services, etc.).
* Each product can have:
  + Title, Description, Price
  + Prerequisites (if any)
  + Visibility settings (Community-specific or global)
* Mighty Arrows products are globally available.

**2.5 Payment System**

* **Payment Gateways**: Stripe.
* **One-time Payments Only**: No automated recurring payments.
* **Invoices & Receipts**: Schools receive invoices and receipts for payments.
* **Admin Validation**: Payments are manually validated where necessary.

**2.6 Authentication & Security**

* Password less google authentication.
* Login via email link (valid for more than 1 week).
* Role-based access control.
* Data encryption and secure transactions.

**3. Non-Functional Requirements**

**3.1 Performance**

* The system should handle high traffic with efficient database queries.
* Payment processing should be seamless and secure.

**3.2 Scalability**

* Support for multiple communities with independent settings.
* Future expansion to additional payment gateways if needed.

**3.3 Usability**

* Intuitive user interface for Super Admins, Community Directors, and Schools.
* Mobile-responsive design.

**3.4 Security**

* Secure authentication and authorization mechanisms.
* Data encryption for sensitive user information.

**3.5 Maintainability**

* Codebase structured for easy updates and enhancements.
* Logging and monitoring for debugging and performance tracking.

**4. Tech Stack (MERN)**

**4.1 Frontend**

* **React.js** for a responsive user interface.

**4.2 Backend**

* **Node.js & Express.js** for handling API requests.
* **MongoDB** for database storage.
* **Mongoose** for schema modeling.

**4.3 Authentication**

* **JWT (JSON Web Tokens)** for session management.

**4.4 Payment Gateway**

* **Stripe** for primary transactions.

**4.5 CMS Features**.

* **Custom Forms** for community-specific application questions.
* ChangeWebsitecontent , logo etc.

**5. Database Schema Overview**

**5.1 Collections**

1. **Users**: Super Admins, Community Directors, Schools.
2. **Communities**: Stores community details and settings.
3. **Applications**: Tracks school applications and status.
4. **Students**: Manages student enrollment details.
5. **Products**: Courses, materials, services.
6. **Payments**: Tracks transactions via Stripe.

**7. Conclusion**

This SRS document provides an in-depth overview of the Multi-Community Admission & Enrollment System. It ensures alignment with business requirements, outlining technical specifications.