**SRS FOR CMS**

1. **Objective:**Develop a flexible content management system designed to meet the diverse needs of various organizations. Built from scratch using Python and Django, it offers robust content management capabilities, user-friendly interfaces and extensibility. It aims to provide a flexible platform for creating, managing and publishing digital content for various organizations.   
   It will be a web-based application that allows users to create, edit, organize and publish content, manage media files and control user access.
2. **Introduction:**A web-based CMS application that can be integrated into various organizational workflows. It will provide both a user-friendly frontend for content consumers and a powerful backend for content managers and administrators.
   1. **Features**- User authentication  
      - Role based access control  
      - Content creation, editing and publishing workflow  
      - Media upload and management  
      - Templates and layouts  
      - Search capabilities  
      - Basic API for content access and integration
   2. **Users**- Administrators: Full system access and rights  
      - Managers: Create, edit and publish content  
      - Editors: Review content  
      - Viewers: Consume published content
3. **Tech stack and Environment:**- Backend: Python with Django  
   - Frontend: HTML, CSS, JavaScript, jQuery  
   - Database: Oracle Database or MySQL
4. **Functional Requirements:  
   4.1 User Management**- User registration and authentication  
   - Role-based access control (Admin, Editor, Viewer, Author)  
   - User profile management

**4.2 Content Management**- CRUD operations for various content types (pages, posts, articles)  
- Text editing for content creation  
- Content categorization   
- Content scheduling   
- Publishing workflow  
  
**4.3 Media Management**- Upload images, documents and other media files  
- Media library   
- Search and filter functionalities  
- Basic image editing (crop, rotate)  
  
**4.4 Search and Navigation**- Full text search across all content  
- Advanced search with filters (date posted, author, category)  
- Menu creation and management  
- Breadcrumb navigation

1. **Non-Functional Requirements:**- Performance   
   - Security  
   - Scalability   
   - Usability  
   - Reliability  
   - Maintainability
2. **Implementation Plan (Phase wise)  
   Phase 1: Project Initiation and setup**- Define project scope and objectives  
   - Setup development environment  
   - Initialize Django project and version control (git)  
   **Phase 2: Start Development**- Implement user authentication   
   - Develop basic content management features   
   - Create database models  
   **Phase 3: Content Management**- Text editing  
   - Content workflow  
   - Create media management system  
   **Phase 4: Design**- Develop frontend user interface  
   - Responsive design   
   **Phase 5: Navigation**- Implement full text search functionality   
   - Develop navigation system   
   **Phase 6: API Integrations   
   Phase 7: Testing   
   Phase 8: Deployment  
   Phase 9: Documentation  
   Phase 10: Maintenance**
3. **Problems:**- Integration challenges  
   - Performance issues (large content)  
   - Security vulnerabilities
4. **Future Scope:**- Multi language support  
   - AI/ML content recommendations