Document Management System - Complete Installation Guide

Overview

This Document Management System is a comprehensive client-server application that provides:

- **Document Management**: Scan, upload, browse, drag & drop, index and store documents
- **User Management**: Admin console for creating user profiles with role-based access control
- **Project Management**: Custom projects with custom-defined fields
- **Advanced Features**: Search, print, email, annotation, and page reordering
- **Database**: SQLite for metadata with external image repository

System Requirements

Server Requirements

- **Operating System**: Windows 10+, macOS 10.14+, or Linux (Ubuntu 18.04+)
- **Node.js**: Version 16.0 or later
- **Memory**: Minimum 2GB RAM (4GB+ recommended)
- **Storage**: 10GB+ available disk space
- **Network**: TCP port 3000 (configurable)

Client Requirements

- **Web Browser**: Chrome 90+, Firefox 88+, Safari 14+, or Edge 90+
- **Network**: Access to server on port 3000
- **Scanner Integration**: TWAIN or WIA compatible scanners (optional)

Installation Steps

Step 1: Download and Extract Files

Create a new directory for the Document Management System and save the following files:

1. **server.js** - Main server application
2. **package.json** - Node.js dependencies
3. **setup.js** - Automated setup script
4. **index.html** - Frontend web interface

Step 2: Run Automated Setup

Open a terminal/command prompt in the project directory and run:

```bash
node setup.js
```

This will:

- Create necessary directories
- Generate configuration files
- Install Node.js dependencies
- Create startup scripts
- Generate documentation

Step 3: Configure Frontend

Copy the frontend HTML file to the public directory:

```bash
# Create public directory if it doesn't exist
mkdir public

# Copy the frontend file
cp index.html public/index.html
```

Step 4: Start the Server

Start the Document Management System:

```bash
npm start
```

Or use the platform-specific startup scripts:

- **Windows**: Double-click `start-dms.bat`
- **Linux/Mac**: Run `./start-dms.sh`

Step 5: Access the Application

1. Open your web browser
2. Navigate to: `http://localhost:3000`
3. Login with default credentials:
   - **Username**: `admin`
   - **Password**: `admin123`

Configuration

Environment Variables (.env file)

```env
# Server Configuration
NODE_ENV=production
PORT=3000
JWT_SECRET=change-this-secret-key-in-production

# Database
DATABASE_PATH=./dms.db

# File Storage
UPLOAD_PATH=./uploads
IMAGE_PATH=./images
MAX_FILE_SIZE=50MB

# Security
BCRYPT_ROUNDS=10
SESSION_TIMEOUT=24h

# File Types
ALLOWED_FILE_TYPES=jpg,jpeg,png,pdf,tiff,tif
```

Network Configuration

To allow remote access, modify the server.js file:

```javascript
// Change from localhost to 0.0.0.0 for remote access
app.listen(PORT, '0.0.0.0', () => {
    console.log(`Server running on http://0.0.0.0:${PORT}`);
});
```

User Guide

Initial Setup Tasks

1. **Change Default Password**
   - Login as admin
   - Go to Users tab
   - Edit admin user
   - Set a strong password
2. **Create User Roles**
   - Go to Roles tab
   - Create roles like "Manager", "Employee", "Read-Only"
   - Assign appropriate permissions
3. **Create Users**
   - Go to Users tab
   - Add new users with appropriate roles
   - Provide login credentials to users
4. **Create Projects**
   - Go to Projects tab
   - Create projects with custom fields
   - Examples: "Invoices", "Contracts", "HR Documents"

Daily Operations

For Administrators

1. **User Management**
   - Create/edit/disable user accounts
   - Assign roles and permissions
   - Monitor user activity
2. **Project Management**
   - Create new projects
   - Define custom fields for indexing
   - Manage project access permissions
3. **System Monitoring**

- Review dashboard statistics
- Monitor disk space usage
- Backup database regularly

For Users

1. **Document Upload**
   - Select a project
   - Create new document
   - Upload pages via drag & drop or file browser
   - Fill in index fields
2. **Document Search**
   - Use search bar in Documents tab
   - Filter by project
   - Search within custom fields
3. **Document Management**
   - View document pages
   - Add additional pages
   - Reorder pages (drag & drop)
   - Print or email documents
   - Add annotations

Features Guide

Document Scanning

- **Scanner Integration**: Connect TWAIN/WIA compatible scanners
- **Batch Scanning**: Scan multiple pages at once
- **Quality Settings**: Adjust resolution and color settings
- **Auto-cropping**: Automatic document edge detection

File Upload Methods

1. **Drag & Drop**: Drag files directly onto upload area
2. **File Browser**: Click to browse and select files
3. **Scanner Integration**: Direct scan-to-upload

4. **Email Integration**: Forward emails as documents

Search Capabilities

- **Full-text Search**: Search document titles and descriptions
- **Field-based Search**: Search within custom project fields
- **Date Range Filters**: Filter by creation/modification dates
- **Project Filtering**: Limit search to specific projects

Permission System

Role-based Access Control

- **Read Only**: View documents and search
- **Edit**: Create and modify documents
- **Delete**: Remove documents and pages
- **Admin**: Full system access

Project-level Permissions

- **Project Access**: Control which users can access specific projects
- **Granular Control**: Different permissions per project
- **Inheritance**: Role permissions + project-specific permissions

Custom Fields

- **Field Types**: Text, Number, Date, Select (dropdown)
- **Required Fields**: Mark fields as mandatory
- **Validation**: Automatic data type validation
- **Search Integration**: All custom fields are searchable

Database Schema

Core Tables

sql

```
-- Users and authentication
users (id, username, email, password, role_id, status, created_at)
roles (id, name, description, permissions, created_at)

-- Projects and structure
projects (id, name, description, created_by, created_at)
project_fields (id, project_id, field_name, field_type, required)

-- Documents and content
documents (id, project_id, title, description, total_pages, created_by)
document_pages (id, document_id, page_number, file_path, annotations)
document_field_values (id, document_id, field_id, field_value)

-- Access control
user_project_access (id, user_id, project_id, access_level)

-- Audit trail
audit_log (id, user_id, action, table_name, record_id, details)
```

Backup and Maintenance

Database Backup

```bash
# Create backup
cp dms.db dms_backup_$(date +%Y%m%d).db

# Automated daily backup (add to crontab)
0 2 * * * cp /path/to/dms.db /backups/dms_$(date +\%Y\%m\%d).db
```

File Repository Backup

```bash
# Backup image repository
tar -czf images_backup_$(date +%Y%m%d).tar.gz images/

# Sync to remote location
rsync -av images/ backup-server:/backups/dms-images/
```

Log Rotation

```bash
# Setup log rotation for application logs
# Add to /etc/logrotate.d/dms
/path/to/logs/*.log {
    daily
    rotate 30
    compress
    missingok
    notifempty
}
```

Troubleshooting

Common Issues

1. **Port Already in Use**

```bash
# Find process using port 3000
netstat -tulpn | grep :3000

# Kill process
kill -9 <process-id>

# Or change port in .env file
PORT=3001
```

2. **Permission Denied**

```bash
# Fix file permissions
chmod -R 755 .
chmod -R 777 uploads/ images/
```

3. **Database Locked**

```bash
# Check for competing processes
ps aux | grep node

# Restart server
npm restart
```

4. **Out of Disk Space**

```bash
# Check disk usage
df -h

# Clean old uploads
find uploads/ -mtime +7 -delete
```

Performance Optimization

1. **Image Compression**
   - Enable automatic image compression
   - Set appropriate quality settings
   - Use progressive JPEG format
2. **Database Optimization**
   - Regular VACUUM operations
   - Index optimization
   - Query performance monitoring
3. **Caching**
   - Enable browser caching for static files
   - Implement Redis for session storage
   - Use CDN for image delivery

Security Best Practices

Production Deployment

1. **Environment Security**

```env
NODE_ENV=production
JWT_SECRET=your-strong-random-secret-key
```

2. **HTTPS Configuration**
   - Use reverse proxy (nginx/Apache)
   - SSL certificate installation
   - Redirect HTTP to HTTPS

3. **Database Security**
    - Regular security updates
    - Access control lists
    - Encrypted backups
4. **File Upload Security**
    - Virus scanning integration
    - File type validation
    - Size limit enforcement

Access Control

1. **Strong Password Policy**
    - Minimum 8 characters
    - Mixed case, numbers, symbols
    - Regular password changes
2. **Session Management**
    - Automatic logout after inactivity
    - Secure session storage
    - Session invalidation on logout
3. **Audit Logging**
    - Log all user actions
    - Monitor failed login attempts
    - Regular log review

API Documentation

Authentication Endpoints

```javascript
POST /api/auth/login
{
  "username": "admin",
  "password": "admin123"
}
```

Project Management

```javascript
GET /api/projects          // List projects
POST /api/projects         // Create project
GET /api/projects/:id      // Get project details
PUT /api/projects/:id      // Update project
DELETE /api/projects/:id   // Delete project
```

Document Management

```javascript
GET /api/projects/:id/documents    // List documents
POST /api/projects/:id/documents   // Create document
POST /api/documents/:id/pages      // Upload pages
GET /api/documents/:id/pages       // List pages
```

Support and Updates

Getting Help

1. **Documentation**: Check README.md for detailed information
2. **Logs**: Review server logs in logs/ directory
3. **Community**: Check GitHub issues for common problems
4. **Professional Support**: Contact system administrator

System Updates

1. **Backup First**: Always backup before updates
2. **Test Environment**: Test updates in non-production environment
3. **Gradual Rollout**: Update in stages for large deployments
4. **Rollback Plan**: Prepare rollback procedures

Monitoring

Set up monitoring for:

- Server uptime and response time
- Disk space usage
- Database performance
- User activity levels

- Error rates and types

Conclusion

This Document Management System provides a comprehensive solution for document storage, indexing, and retrieval with robust user management and security features. The modular design allows for easy customization and scaling based on organizational needs.

For additional features or customization requirements, the system architecture supports extension through:

- Additional API endpoints
- Custom field types
- Integration with external systems
- Advanced workflow capabilities
- Reporting and analytics modules

Regular maintenance, security updates, and user training will ensure optimal performance and user adoption of the system.