29. Document Management System

Creating a Document Management System with version control using the MEAN stack (MongoDB, Express.js, Angular, Node.js) is a complex project, but it can be highly valuable for businesses and organizations. Below is a high-level overview of how to build such a system, along with some code snippets to guide you:

Project Setup and Structure

Set up a new project folder and structure for your Document Management System. Install the required Node.js packages and create a basic Angular application.

```
# Create a new Angular application

        ng new document-management-app
```

- Backend (Node.js & Express.js)

Create the backend of your Document Management System using Node.js and Express.js.

Installation of Packages

Install the necessary packages for Express.js, Mongoose (for MongoDB), and other dependencies.

```
npm install express mongoose cors
```

Setting up Express.js

Create your Express.js server, set up middleware, and handle routes.

- **javascript**

```
// server.js

const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');

const app = express();

// Middleware
app.use(express.json());
app.use(cors());

// Database connection
mongoose.connect('mongodb://localhost/document-management-app', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
  useCreateIndex: true,
});

// Define Mongoose models for User, Document, and DocumentVersion data
const User = mongoose.model('User', {
  username: String,
  password: String, // Use hashing for security
  // Add more user-related fields as needed
});

const Document = mongoose.model('Document', {
  name: String,
  // Add more document-related fields as needed
});

const DocumentVersion = mongoose.model('DocumentVersion', {
  documentId: mongoose.Schema.Types.ObjectId,
  version: Number,
  content: String,
  // Add more version-related fields as needed
});
```

```
// Routes for managing users, documents, versions, and version history
app.post('/api/register', async (req, res) => {
  // Register a new user
  // Store hashed password in the database
});

app.post('/api/login', async (req, res) => {
  // Authenticate user and generate a JWT token
});

app.post('/api/documents', async (req, res) => {
  // Create a new document
  // Save the document to the database
});

// Create a route for uploading new document versions
app.post('/api/documents/:documentId/versions', async (req, res) => {
  // Save a new version of the document to the database
});
```

// Create routes for retrieving documents and document version history

- Frontend (Angular)

Create the frontend of your Document Management System using Angular. Design the user interface for managing documents, viewing version history, and user accounts.

Design and UI

Design the user interface for your Document Management System using Angular components, templates, and styles.

Document Management

Create components and forms for users to create, update, and manage documents.

Version Control

Design components for viewing version history, comparing versions, and restoring previous versions.

User Authentication

Implement user registration and login functionality.

- **typescript**

// document-management.component.ts

```
import { Component } from '@angular/core';
import { DocumentService } from './document.service';

@Component({
  selector: 'app-document-management',
  templateUrl: './document-management.component.html',
})
export class DocumentManagementComponent {
  documentName: string;
  documentContent: string;

  constructor(private documentService: DocumentService) {}

  createDocument() {
    this.documentService.createDocument(this.documentName,
this.documentContent);
  }
}
```

MongoDB

Create a MongoDB database to store user profiles, documents, and document versions.

Putting It All Together

Integrate the frontend and backend by making API requests from Angular components to Node.js routes. Ensure that you handle document creation, version control, user authentication, and version history properly.

Building a Document Management System with version control is a sophisticated project that can be extended with features like access control, document collaboration, advanced search capabilities, notifications, and document archiving for a comprehensive document management solution.