# Document Management System

To create a system that accounts for scalability, a microservices approach would be ideal so that each service can stand on its own and scale individually when needed.

I would use the following services to create the system:

1. Backend Services:
   1. Responsible for managing documents, including upload, versioning, deletion, and search functionalities.
   2. Technology: Flask (Python micro-framework) for simplicity and ease of development.
   3. Database: SQLite for simplicity, or PostgreSQL for more scalability if time allows.
   4. Additional Libraries: Flask-RESTful for building RESTful APIs, Flask-SQLAlchemy for ORM.
2. Frontend:
   1. A simple React frontend for interacting with the backend services.
   2. Use functional components and hooks to keep the codebase concise and manageable.
   3. Utilize React Router for client-side routing.
   4. Use Axios or Fetch API for making HTTP requests to backend services.
3. Containerization:
   1. Use Docker for containerization to simplify deployment and ensure consistency across environments.
   2. Docker Compose can be used for orchestrating multiple microservices during development.
4. Deployment:
   1. Deploy the application on a cloud platform such as Heroku or AWS Elastic Beanstalk for simplicity.
   2. Utilize container orchestration platforms like Kubernetes if scalability requirements demand it, though this might be overkill for a basic implementation within the time constraint.