# 📄 Document Management System – Requirements

## 🧩 1. Functional Requirements

### 1.1 File Upload via UI

- Users should be able to upload documents via a web-based UI.
- The UI must support file selection and preview.
- Uploads should invoke a backend API endpoint for processing and validation.

### 1.2 API-based Upload

- A secure REST API endpoint (e.g., `POST /api/documents/upload`) should accept file uploads along with metadata.
- API should support authentication and authorization (e.g., JWT or OAuth2). (For now assume that the auth bearer token is always present in the requests)

### 1.3 AWS Integration

- Files must be stored in AWS S3 using the AWS SDK (Java preferred).
- File URLs should be stored in the database along with metadata.
- Use a structured S3 path pattern: `companyId/employeeId/documentType/filename`.

### 1.4 Metadata Handling

For each uploaded document, the following metadata should be captured and stored:

- `documentName` (string)
- `documentType` (enum):

    - DL, Passport, Work Authorization, I-9, MSA, Purchase Order, Background Check Report, W-4

- `documentNumber` (string)
- `documentStatus` (enum): Active, Expired
- `validFrom` (date)
- `validTo` (date)
- `companyId` (reference)
- `vendorId` (reference)
- `employeeId` (reference)

**Additional metadata for Work Authorization:**

- `subType` (enum): H1-B, H4 EAD, L2 EAD, GC, OPT-EAD

### 1.5 File Constraints

- Allowed file types: `.pdf`, `.docx`
- Maximum file size: **4 MB**
- Duplicate documents (based on document type + number + employee/vendor/company) must be rejected with a clear error message.

---

## 🔐 2. Validation and Rules

- All metadata fields are required unless conditionally optional (e.g., `subType` only for Work Authorization).
- Check `validTo` and `validityEndDate` dates to determine document status (Active/Expired).

- Enforce unique constraint on: `(documentNumber + documentType + employeeId)`
- Reject uploads if:
    - File exceeds 4 MB
    - File type is not allowed
    - Duplicate detected

---

## 🧠 3. System Associations

Each document must be associated with one or more of the following:

- **Company** (e.g., Employer)
- **Vendor** (e.g., Contract agency)
- **Employee** (e.g., the individual whose document is being uploaded)

These associations should be stored as foreign key references in the database.

The document table should have the meta data along with the employerId\D, vendorID and employeeID

---

## 📚 4. Storage and Security

- Files are stored in **AWS S3** with private access.
- Generate **pre-signed URLs** for time-limited access to documents.
- Metadata is stored in a **relational database** (e.g., PostgreSQL, MySQL).
- Ensure encryption at rest and in transit for documents.

---

## 🧪 5. Optional Features (Future Scope)

- Versioning: Allow newer versions of the same document while archiving older ones.
- Notifications: Alert HR/admins when a document is about to expire.
- Search & filter: Allow querying documents by employee, type, date range, or status.

---

Tech Specs:

1. Build using Java/Spring Book, AWS SDK, MySQL or Postgresql
2. Use Post Man for your testing of the APIs. (Upload APi, get Document API etc)
3. No need to implement user auth. Just check if the bearer token is present in the API requests