PRODUCTION-GRADE DOCUMENT PROCESSING SYSTEM - COMPLETE PLAN

STEP 1: Install Required Packages

```
# Navigate to backend workspace
cd apps/backend

# Install document processing dependencies
yarn add pdf-parse mammoth tesseract.js sharp cheerio mime-types file-type youtube-transc

# Install type definitions
yarn add -D @types/pdf-parse @types/mime-types @types/multer
```

STEP 2: Project Structure

```
apps/backend/src/v1/documents/
├── documents.module.ts
├── controllers/
│   ├── documents.controller.ts
│   ├── folders.controller.ts
│   └── embeddings.controller.ts
├── services/
│   ├── documents.service.ts
│   ├── folders.service.ts
│   ├── embeddings.service.ts
│   ├── vector-search.service.ts
│   └── cost-tracking.service.ts
├── processors/
│   ├── base-processor.abstract.ts
│   ├── pdf-processor.service.ts
│   ├── image-processor.service.ts
│   ├── youtube-processor.service.ts
│   ├── url-processor.service.ts
│   └── orchestrator.service.ts
├── queues/
│   ├── document-queue.producer.ts
│   └── document-queue.consumer.ts
├── guards/
│   ├── workspace-ownership.guard.ts
│   └── document-access.guard.ts
├── interceptors/
│   ├── file-size-limit.interceptor.ts
```

```
│   └── file-type-validator.interceptor.ts
├── dto/
│   ├── upload-document.dto.ts
│   ├── link-external.dto.ts
│   ├── update-document.dto.ts
│   ├── search-documents.dto.ts
│   ├── list-documents.dto.ts
│   ├── create-folder.dto.ts
│   ├── update-folder.dto.ts
│   └── move-items.dto.ts
├── utils/
│   ├── text-chunker.util.ts
│   ├── mime-detector.util.ts
│   ├── token-counter.util.ts
│   └── cost-calculator.util.ts
└── types/
    ├── document.types.ts
    ├── processor.types.ts
    └── embedding.types.ts
```

STEP 3: Complete API Endpoints

3.1 Document Management Endpoints

Method	Endpoint	Auth	Rate Limit	Purpose
POST	`/api/v1/workspaces/:workspaceId/documents/upload`	JWT	10/min	Upload file (PDF, image, etc.)
POST	`/api/v1/workspaces/:workspaceId/documents/external`	JWT	20/min	Link external source (YouTube, URL)
GET	`/api/v1/workspaces/:workspaceId/documents`	JWT	60/min	List all documents
GET	`/api/v1/documents/:documentId`	JWT	100/min	Get document details
PATCH	`/api/v1/documents/:documentId`	JWT	30/min	Update document metadata
DELETE	`/api/v1/documents/:documentId`	JWT	30/min	Delete document + embeddings
POST	`/api/v1/documents/:documentId/reprocess`	JWT	5/min	Regenerate embeddings

Method	Endpoint	Auth	Rate Limit	Purpose
GET	`/api/v1/documents/:documentId/status`	JWT	100/min	Get processing status
GET	`/api/v1/documents/:documentId/download`	JWT	30/min	Download original file

3.2 Vector Search Endpoints

Method	Endpoint	Auth	Rate Limit	Purpose
POST	`/api/v1/workspaces/:workspaceId/documents/search`	JWT	60/min	Semantic search across documents
GET	`/api/v1/documents/:documentId/embeddings`	JWT	30/min	List all embeddings for document
GET	`/api/v1/documents/:documentId/embeddings/stats`	JWT	30/min	Embedding statistics (count, cost, model)
POST	`/api/v1/documents/bulk-search`	JWT	30/min	Search across multiple workspaces

3.3 Folder Management Endpoints

Method	Endpoint	Auth	Rate Limit	Purpose
POST	`/api/v1/workspaces/:workspaceId/folders`	JWT	30/min	Create new folder
GET	`/api/v1/workspaces/:workspaceId/folders`	JWT	60/min	List folders (tree structure)
GET	`/api/v1/folders/:folderId`	JWT	60/min	Get folder details + documents
PATCH	`/api/v1/folders/:folderId`	JWT	30/min	Update folder (name, icon, color)
DELETE	`/api/v1/folders/:folderId`	JWT	20/min	Delete folder (cascade documents)
POST	`/api/v1/folders/move`	JWT	30/min	Move documents/folders

Method	Endpoint	Auth	Rate Limit	Purpose
POST	`/api/v1/folders/:folderId/duplicate`	JWT	10/min	Duplicate folder structure

STEP 4: Request/Response Specifications

4.1 Upload Document (Internal File)

Request:

```
POST /api/v1/workspaces/:workspaceId/documents/upload
Authorization: Bearer {jwt_token}
Content-Type: multipart/form-data

FormData:
  file: File (required, max 100MB)
  name: string (optional)
  folderId: string (optional)
```

Response (201):

```
{
  "statusCode": 201,
  "message": "Document uploaded successfully",
  "data": {
    "id": "doc_clx123abc",
    "name": "Report.pdf",
    "sourceType": "INTERNAL",
    "fileType": "application/pdf",
    "mimeType": "application/pdf",
    "sizeInBytes": 1048576,
    "status": "UPLOADING",
    "s3Bucket": "actopod-documents-prod",
    "storageKey": "documents/ws_abc123/doc_clx123abc/Report.pdf",
    "folderId": null,
    "uploadedBy": "user_xyz789",
    "createdAt": "2025-10-27T22:10:00.000Z",
    "estimatedProcessingTime": "30-45 seconds"
  }
}
```

Error Responses:

- `400 Bad Request`: Invalid file type or missing file
- `413 Payload Too Large`: File exceeds size limit
- `402 Payment Required`: Insufficient credits
- `403 Forbidden`: No workspace access

- `429 Too Many Requests`: Rate limit exceeded

4.2 Link External Source

Request:

```
POST /api/v1/workspaces/:workspaceId/documents/external
Authorization: Bearer {jwt_token}
Content-Type: application/json

{
  "sourceType": "YOUTUBE",
  "url": "https://www.youtube.com/watch?v=dQw4w9WgXcQ",
  "name": "AI Tutorial Video",
  "folderId": "folder_abc123"
}
```

Response (201):

```
{
  "statusCode": 201,
  "message": "External source linked successfully",
  "data": {
    "id": "doc_yt456def",
    "name": "AI Tutorial Video",
    "sourceType": "YOUTUBE",
    "externalUrl": "https://www.youtube.com/watch?v=dQw4w9WgXcQ",
    "externalProvider": "youtube",
    "externalFileId": "dQw4w9WgXcQ",
    "status": "PROCESSING",
    "metadata": {
      "title": "AI Tutorial Video",
      "duration": 600,
      "thumbnail": "https://img.youtube.com/vi/dQw4w9WgXcQ/maxresdefault.jpg",
      "channelName": "Tech Channel"
    },
    "folderId": "folder_abc123",
    "uploadedBy": "user_xyz789",
    "createdAt": "2025-10-27T22:11:00.000Z"
  }
}
```

4.3 List Documents

Request:

```
GET /api/v1/workspaces/:workspaceId/documents?page=1&limit=20&folderId=folder_123&status=
Authorization: Bearer {jwt_token}
```

Query Parameters:

- `page` (number, default: 1)
- `limit` (number, default: 20, max: 100)
- `folderId` (string, optional): Filter by folder
- `status` (enum, optional): UPLOADING | PROCESSING | READY | ERROR
- `sourceType` (enum, optional): INTERNAL | YOUTUBE | URL
- `search` (string, optional): Search by name
- `sortBy` (string, default: "createdAt"): createdAt | name | sizeInBytes
- `sortOrder` (string, default: "desc"): asc | desc

Response (200):


```
{
  "statusCode": 200,
  "message": "Documents retrieved successfully",
  "data": [
    {
      "id": "doc_123",
      "name": "Q4 Report.pdf",
      "sourceType": "INTERNAL",
      "fileType": "application/pdf",
      "sizeInBytes": 2048576,
      "status": "READY",
      "folderId": "folder_abc",
      "folder": {
        "id": "folder_abc",
        "name": "Reports",
        "icon": "",
        "color": "#3B82F6"
      },
      "embeddingCount": 45,
      "createdAt": "2025-10-27T22:00:00.000Z",
      "updatedAt": "2025-10-27T22:05:00.000Z"
    }
  ],
  "pagination": {
    "totalItems": 156,
    "totalPages": 8,
    "currentPage": 1,
    "pageSize": 20
  }
}
```

4.4 Semantic Search

Request:

```
POST /api/v1/workspaces/:workspaceId/documents/search
Authorization: Bearer {jwt_token}
Content-Type: application/json

{
  "query": "What are the key findings from the Q3 report?",
  "topK": 5,
  "threshold": 0.75,
  "documentIds": ["doc_123", "doc_456"],
  "folderId": "folder_reports"
}
```

Response (200):

```
{
  "statusCode": 200,
  "message": "Search completed successfully",
  "data": {
    "results": [
      {
        "documentId": "doc_123",
        "documentName": "Q3 Report.pdf",
        "chunkIndex": 12,
        "chunkText": "Key findings: Revenue increased by 23% compared to Q2. Customer acc
        "similarity": 0.89,
        "metadata": {
          "pageNumber": 5,
          "section": "Financial Summary"
        }
      },
      {
        "documentId": "doc_123",
        "documentName": "Q3 Report.pdf",
        "chunkIndex": 28,
        "chunkText": "The main highlights include: Product launches exceeded targets, mar
        "similarity": 0.84,
        "metadata": {
          "pageNumber": 12,
          "section": "Key Highlights"
        }
      }
    ],
    "processingTime": "245ms",
    "embeddingModel": "text-embedding-004",
    "totalDocumentsSearched": 2
  }
}
```

4.5 Create Folder

Request:

```
POST /api/v1/workspaces/:workspaceId/folders
Authorization: Bearer {jwt_token}
Content-Type: application/json

{
  "name": "Q4 Reports",
  "parentId": "folder_parent123",
  "icon": "",
  "color": "#10B981"
}
```

Response (201):

```
{
  "statusCode": 201,
  "message": "Folder created successfully",
  "data": {
    "id": "folder_new789",
    "workspaceId": "ws_abc123",
    "name": "Q4 Reports",
    "parentId": "folder_parent123",
    "icon": "",
    "color": "#10B981",
    "sortOrder": 0,
    "createdBy": "user_xyz789",
    "createdAt": "2025-10-27T22:15:00.000Z",
    "documentCount": 0,
    "subfolderCount": 0
  }
}
```

4.6 Move Documents/Folders

Request:

```
POST /api/v1/folders/move
Authorization: Bearer {jwt_token}
Content-Type: application/json

{
  "items": [
    {
      "type": "document",
      "id": "doc_123"
    },
    {
      "type": "folder",
```

```
      "id": "folder_456"
    }
  ],
  "targetFolderId": "folder_target789"
}
```

Response (200):

```
{
  "statusCode": 200,
  "message": "Items moved successfully",
  "data": {
    "movedCount": 2,
    "items": [
      {
        "id": "doc_123",
        "type": "document",
        "newFolderId": "folder_target789"
      },
      {
        "id": "folder_456",
        "type": "folder",
        "newParentId": "folder_target789"
      }
    ]
  }
}
```

⚙ STEP 5: Processing Pipeline Flow

```
1. Upload/Link Document
   ↓
2. Validate (Size, Type, Credits)
   ↓
3. Upload to S3 (if internal)
   ↓
4. Create DB Record (status: UPLOADING)
   ↓
5. Queue Background Job (BullMQ)
   ↓
6. Job Consumer Picks Up
   ↓
7. Route to Processor (PDF/Image/YouTube/URL)
   ↓
8. Extract Text Content
   ↓
9. Chunk Text (512 tokens, 50 overlap)
   ↓
10. Generate Embeddings (Gemini 768d)
    ↓
11. Store in pgvector + S3 Backup
    ↓
```

```
12. Track Costs & Deduct Credits
    ↓
13. Update Status → READY
    ↓
14. Emit WebSocket Event (document.ready)
```

🔄 STEP 6: Document Processing Types

6.1 PDF Processing

- **Library**: `pdf-parse`
- **Input**: PDF file from S3
- **Output**: Extracted text + metadata
- **Features**:
  - Text layer extraction
  - OCR fallback for scanned PDFs (Tesseract)
  - Handle encrypted PDFs (password support)
  - Extract metadata (pages, author, title)
- **Estimated Time**: 5-30 seconds

6.2 Image Processing

- **Library**: `tesseract.js`, `sharp`
- **Input**: Image file (JPG, PNG, WebP)
- **Output**: OCR text + description
- **Features**:
  - OCR text extraction
  - Image optimization (resize if >10MB)
  - Optional: Gemini Vision description
  - Support multiple languages
- **Estimated Time**: 10-60 seconds

6.3 YouTube Processing

- **Library**: `youtube-transcript`, `@distube/ytdl-core`
- **Input**: YouTube URL
- **Output**: Video transcript + metadata
- **Features**:
  - Fetch auto-generated captions

    - Manual captions support
    - Multi-language transcripts
    - Extract video metadata (title, duration, thumbnail)
- **Estimated Time**: 5-15 seconds

6.4 URL Processing

- **Library**: `cheerio`, `axios`
- **Input**: Web URL
- **Output**: Extracted main content
- **Features**:
    - Content extraction (Readability algorithm)
    - Remove ads, navigation, footer
    - Extract metadata (title, author, description)
    - Handle markdown/HTML
- **Estimated Time**: 3-10 seconds

✂️ STEP 7: Text Chunking Strategy

Algorithm:

```
Input: Full document text (e.g., 50,000 tokens)
Output: Array of chunks (e.g., 100 chunks × 512 tokens)

Process:
1. Count total tokens using tiktoken
2. Split text by paragraphs (\n\n)
3. Combine paragraphs into chunks ≤512 tokens
4. Add 50-token overlap between consecutive chunks
5. Preserve sentence boundaries (don't split mid-sentence)
6. Track chunk index and character position
```

Parameters:

- **Max tokens per chunk**: 512
- **Overlap**: 50 tokens
- **Separators** (priority): `\n\n` → `\n` → `.` → `!` → `?` →
- **Min chunk size**: 50 tokens (discard smaller)

🧠 STEP 8: Embedding Generation (Gemini)

Provider Selection Logic:

```
1. Check user's Gemini API key (BYOK)
   ✓ Found → Use user's key

2. Check user's OpenAI API key (BYOK)
   ✓ Found → Use OpenAI (1536d)

3. Check subscription tier
   ✓ PRO/TEAM → Use platform Gemini key
   ✓ HOBBYIST → Reject (require BYOK)

4. Check credit balance
   ✗ Insufficient → Reject with error
```

API Call Flow:

```
// Batch: 100 chunks at a time
for (let i = 0; i < chunks.length; i += 100) {
  const batch = chunks.slice(i, i + 100);

  // Parallel processing (5 concurrent)
  const embeddings = await Promise.all(
    batch.map(chunk => generateEmbedding(chunk.text))
  );

  // Store in pgvector + S3
  await storeEmbeddings(documentId, embeddings);
}
```

Cost Calculation:

- **Gemini**: $0.01 per 1M tokens
- **Example**: 50,000 tokens = $0.0005
- **Track in database**: `DocumentProcessingCost`

🧠 STEP 9: Dual Vector Storage

PostgreSQL (pgvector):

- **Purpose**: Fast similarity search
- **Storage**: `vector(768)` column
- **Index**: HNSW (m=16, ef_construction=64)
- **Query time**: <40ms for 10M vectors

S3 Backup:

- **Purpose**: Disaster recovery, cost optimization
- **Format**: Binary Float32Array
- **Path**: `embeddings/{documentId}/chunk-{index}.bin`
- **Cost**: $0.023/GB/month

🔟 STEP 10: Vector Search Query

SQL Query:

```
SELECT
  e."documentId",
  d.name as "documentName",
  e."chunkText",
  e."chunkIndex",
  1 - (e.vector <=> $1::vector(768)) as similarity,
  e.metadata
FROM "documents"."Embedding" e
JOIN "documents"."Document" d ON d.id = e."documentId"
WHERE
  d."workspaceId" = $2
  AND d.status = 'READY'
  AND e."vectorDimension" = 768
  AND (1 - (e.vector <=> $1::vector(768))) >= $3
  AND (
    CASE WHEN $4::text[] IS NOT NULL
    THEN d.id = ANY($4::text[])
    ELSE TRUE END
  )
  AND (
    CASE WHEN $5::text IS NOT NULL
    THEN d."folderId" = $5
    ELSE TRUE END
  )
ORDER BY e.vector <=> $1::vector(768)
LIMIT $6
```

Query Parameters:

- `$1`: Query embedding (vector(768))
- `$2`: Workspace ID
- `$3`: Similarity threshold (0.7)
- `$4`: Document IDs filter (optional)
- `$5`: Folder ID filter (optional)
- `$6`: Top K results (5)

💰 STEP 11: Cost Tracking & Credit Deduction

Processing Cost Breakdown:

```
{
  documentId: "doc_123",
  workspaceId: "ws_abc",
  subscriptionId: "sub_xyz",
  processingType: "DOCUMENT_EMBEDDING",

  extractionCost: 0.0000,  // YouTube API free
  embeddingCost: 0.0005,   // Gemini $0.01/1M × 50K tokens
  totalCostInUsd: 0.0005,

  chunkCount: 98,
  embeddingModel: "text-embedding-004",
  processingTimeMs: 45000,
  tokensProcessed: 50234
}
```

Credit Deduction:

```
UPDATE "billing"."Subscription"
SET credits = credits - 5  -- $0.0005 = 5 credits (if 1 credit = $0.0001)
WHERE id = 'sub_xyz'
AND credits >= 5  -- Ensure sufficient balance
```

📋 STEP 12: Background Job Queue (BullMQ)

Queue Configuration:

```
{
  name: 'document-processing',
  redis: {
    host: process.env.REDIS_HOST,
    port: 6379,
  },
  limiter: {
    max: 10,          // Max 10 jobs per second
    duration: 1000,
  },
  defaultJobOptions: {
    attempts: 3,
    backoff: {
      type: 'exponential',
      delay: 2000,    // 2s, 4s, 8s
    },
    removeOnComplete: {
      age: 86400,     // Keep for 24h
```

```
      count: 1000,
    },
    removeOnFail: {
      age: 604800,    // Keep failures for 7 days
    },
  },
}
```

Job Types:

1. **document.process** (Priority: High)
2. **document.reprocess** (Priority: Normal)
3. **document.cleanup** (Priority: Low)
4. **thumbnail.generate** (Priority: Low)

STEP 13: Error Handling

Error Types & Recovery:

Error	Retry Strategy	Recovery
S3 Upload Failed	3 retries (exponential backoff)	Mark as ERROR, notify user
Text Extraction Failed	Try OCR fallback → If fails, mark ERROR	Store error message
Embedding API Failed	3 retries with backoff	If quota exceeded, pause & notify
Insufficient Credits	No retry	Reject immediately, notify user
Invalid File	No retry	Mark ERROR immediately
Rate Limit Exceeded	Exponential backoff (max 5 min)	Resume when rate limit resets

Status Transitions:

```
UPLOADING → PROCESSING → READY
         ↘ ERROR (with errorMessage in DB)
```

STEP 14: Monitoring & Observability

Metrics to Track:

- Documents processed per hour
- Average processing time by file type
- Embedding generation cost per document
- Error rate by processor type

- Queue depth and processing lag
- Credit consumption rate
- pgvector query performance

Logging:

- Document upload events
- Processing start/complete/error
- Embedding generation batches
- Credit deductions
- Search queries (with latency)

🔟 STEP 15: Rate Limiting

By Endpoint:

- Upload: 10 requests/min
- External link: 20 requests/min
- List documents: 60 requests/min
- Search: 60 requests/min
- Reprocess: 5 requests/min

By Subscription:

- HOBBYIST: 50 documents/month
- PRO: 500 documents/month
- TEAM: 5,000 documents/month

✅ STEP 16: Implementation Checklist

- [ ] Install dependencies (`pdf-parse`, `tesseract.js`, etc.)
- [ ] Create module structure
- [ ] Implement documents controller & service
- [ ] Implement folders controller & service
- [ ] Create PDF processor
- [ ] Create Image processor
- [ ] Create YouTube processor
- [ ] Create URL processor
- [ ] Implement text chunking utility

- [ ] Implement embeddings service (Gemini)
- [ ] Implement vector search service
- [ ] Create BullMQ queue producer
- [ ] Create BullMQ queue consumer
- [ ] Implement cost tracking service
- [ ] Add guards (workspace ownership, document access)
- [ ] Add interceptors (file size, type validation)
- [ ] Create DTOs with validation
- [ ] Set up pgvector HNSW indexes
- [ ] Implement S3 dual storage
- [ ] Add WebSocket events (processing updates)
- [ ] Write unit tests
- [ ] Write integration tests
- [ ] Add Swagger documentation
- [ ] Set up monitoring & logging
- [ ] Configure rate limiting
- [ ] Test end-to-end flow

This plan provides a **complete, production-ready document processing system** with Gemini embeddings, multi-source support, and full RAG capabilities.