

# Sheet Service - Complete Folder Structure

## Directory Structure

```
services/sheet-service/  
├── src/  
│   ├── auth/  
│   │   └── jwt.strategy.ts  
│   ├── config/  
│   │   └── configuration.ts  
│   ├── database/  
│   │   ├── schemas/  
│   │   │   └── sheet.schema.ts  
│   │   └── database.module.ts  
│   ├── sheets/  
│   │   ├── dto/  
│   │   │   ├── create-sheet.dto.ts  
│   │   │   ├── update-sheet.dto.ts  
│   │   │   ├── sheet-response.dto.ts  
│   │   │   └── sheet-query.dto.ts  
│   │   ├── sheets.controller.ts  
│   │   ├── sheets.service.ts  
│   │   └── sheets.module.ts  
│   ├── app.module.ts  
│   └── main.ts  
├── test/  
├── package.json  
├── tsconfig.json  
├── tsconfig.build.json  
├── nest-cli.json  
└── README.md
```

## Key Features Implemented

### Complete CRUD Operations

- **GET /api/v1/sheets** - List sheets with pagination, filtering, and sorting
- **GET /api/v1/sheets/:id** - Get sheet by ID with populated references
- **POST /api/v1/sheets** - Create new sheet (Preparateur, Admin)
- **PUT /api/v1/sheets/:id** - Update sheet metadata or status

- **DELETE /api/v1/sheets/:id** - Deactivate sheet (soft delete)

## ✓ Additional Endpoints

- **GET /api/v1/sheets/statistics** - Get sheet statistics (Admin only)
- **GET /api/v1/sheets/reference/:reference** - Get sheet by reference number
- **GET /api/v1/sheets/department/:departmentId** - Get sheets by department
- **GET /api/v1/sheets/my-sheets** - Get current user's sheets
- **PATCH /api/v1/sheets/:id/validate** - Validate or reject sheet (IPDF, IQP, Admin)

## ✓ Sheet Schema Features

- **Core Fields:** title, reference, description, filePath, encryptionLv
- **Status Management:** PENDING, VALIDATED, REJECTED enum
- **User Associations:** uploadedBy, validatedBy (references to User)
- **Department Association:** department (reference to Department)
- **File Metadata:** originalFileName, mimeType, fileSize, checksum
- **Audit Trail:** createdAt, updatedAt, lastModifiedAt, lastModifiedBy
- **Validation Info:** validatedAt, validatedBy, validationComments
- **Additional Features:** tags array, version, isActive flag
- **Virtual Fields:** isValidated, isPending, isRejected
- **Security:** filePath and encryptionLv are not exposed in API responses

## ✓ Advanced Features

- **Text Search:** Full-text search on title, reference, and description
- **Comprehensive Filtering:** By status, department, uploader, tags, active status
- **Population Support:** Related entities (uploader, department, validator)
- **Pagination:** With metadata (total, pages, hasNext, hasPrev)
- **Sorting:** Multiple fields with asc/desc order
- **Statistics:** Aggregated data by department and overall metrics
- **Validation Workflow:** Multi-stage validation with comments and timestamps

## ✓ Validation & DTOs

- **CreateSheetDto** - Full validation for sheet creation with file metadata
- **UpdateSheetDto** - Partial validation for updates

- **SheetResponseDto** - Secure response format (excludes filePath/encryptionlv)
- **SheetQueryDto** - Advanced query parameters with validation
- **Class-validator** decorators for all input validation
- **Transform** decorators for data sanitization and type conversion

## ✓ Service Features

- **Comprehensive Error Handling:** Proper HTTP status codes and logging
- **Reference Uniqueness:** Prevents duplicate reference numbers
- **Role-Based Access:** Different permissions for different user roles
- **Validation Workflow:** Status transitions with proper validation
- **Department Integration:** Sheet-department associations
- **User Integration:** Sheet-user associations for uploaders and validators
- **Statistics Generation:** Advanced aggregation queries
- **Soft Delete:** isActive flag for data retention

## ✓ Security & Architecture

- **Global Guards:** JwtAuthGuard, RolesGuard, DepartmentGuard
- **Role-Based Authorization:** Different endpoints for different roles
- **JWT Authentication:** Integrated with shared JWT strategy pattern
- **Data Security:** Sensitive fields (filePath, encryptionlv) not exposed
- **Input Validation:** Comprehensive validation on all endpoints
- **Logging:** Structured logging with correlation IDs

## ✓ API Documentation

- **Complete Swagger Documentation:** All endpoints documented
- **Request/Response Examples:** Detailed examples for all operations
- **Error Response Documentation:** All possible error scenarios
- **Role Requirements:** Clear indication of required permissions
- **Parameter Validation:** Detailed validation rules and constraints

## ✓ Configuration & Infrastructure

- **Exact TypeScript Configuration:** Matching user-service patterns
- **Shared Package Imports:** Using @instruction-sheet/shared

- **MongoDB Integration:** Optimized connection settings
- **Rate Limiting:** Configurable throttling
- **Health Check:** Comprehensive health endpoint
- **Environment Configuration:** Flexible config management
- **Security Middleware:** Helmet, compression, CORS
- **Global Validation:** ValidationPipe with transform

## Installation & Setup

### 1. Navigate to the service directory:

```
bash
cd services/sheet-service
```

### 2. Install dependencies:

```
bash
npm install
```

### 3. Set environment variables (optional):

```
bash
# .env file
SHEET_SERVICE_PORT=3004
MONGODB_URI=mongodb://127.0.0.1:27017/instruction_sheet_db
JWT_SECRET=your-super-secret-jwt-key
NODE_ENV=development
MAX_FILE_SIZE=10485760 # 10MB
ALLOWED_MIME_TYPES=application/pdf
UPLOAD_PATH=./uploads
```

### 4. Build and start the service:

```
bash
# Development mode
npm run start:dev

# Production build
npm run build
npm run start:prod
```

## Service URLs

- **Main Service:** <http://localhost:3004>
- **API Documentation:** <http://localhost:3004/api/docs>
- **Health Check:** <http://localhost:3004/health>
- **API Base:** <http://localhost:3004/api/v1>

## Role-Based Access Control

### Admin

- Full access to all endpoints
- Can view statistics
- Can validate/reject sheets
- Can create, update, delete any sheet

### Preparateur

- Can create new sheets
- Can update own sheets
- Can view all sheets
- Cannot validate sheets

### IPDF/IQP

- Can view all sheets
- Can validate/reject sheets
- Cannot create new sheets
- Can update sheet status

### End User

- Can view sheets (read-only)
- Limited access to sheet operations

## Testing Endpoints

### Create Sheet

bash

```
curl -X POST http://localhost:3004/api/v1/sheets \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_JWT_TOKEN" \
-d '{
  "title": "Safety Procedures Manual",
  "reference": "REF-2024-001",
  "description": "Comprehensive safety procedures",
  "filePath": "/uploads/safety-manual.pdf",
  "encryptionIv": "abc123def456",
  "department": "60f7b1b3e1b3c4a5d8e9f0a1",
  "originalFileName": "safety-manual.pdf",
  "mimeType": "application/pdf",
  "fileSize": 2048576
}'
```

## Get All Sheets

bash

```
curl http://localhost:3004/api/v1/sheets?page=1&limit=10&status=PENDING \
-H "Authorization: Bearer YOUR_JWT_TOKEN"
```

## Get Sheet by ID

bash

```
curl http://localhost:3004/api/v1/sheets/[sheet_id] \
-H "Authorization: Bearer YOUR_JWT_TOKEN"
```

## Update Sheet Status

bash

```
curl -X PUT http://localhost:3004/api/v1/sheets/[sheet_id] \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_JWT_TOKEN" \
-d '{"status": "VALIDATED", "validationComments": "Approved for use"}'
```

## Validate Sheet

bash

```
curl -X PATCH http://localhost:3004/api/v1/sheets/[sheet_id]/validate \  
-H "Content-Type: application/json" \  
-H "Authorization: Bearer YOUR_JWT_TOKEN" \  
-d '{"status": "VALIDATED", "comments": "Sheet meets all requirements"}'
```

## Next Steps

1. **Test the basic CRUD operations** using the provided curl commands or Swagger UI
2. **Verify authentication** works with JWT tokens from auth-service
3. **Test role-based access control** with different user roles
4. **Validate sheet upload workflow** and status transitions
5. **Prepare for file upload integration** in the next iteration
6. **Add QR code generation** endpoints
7. **Implement file encryption/decryption** logic
8. **Add RabbitMQ messaging** for workflow notifications

## Future Enhancements Ready For

The service is architected to easily support:

- **File Upload Handling:** Multer integration for actual file uploads
- **Encryption/Decryption:** AES-256 encryption for file storage
- **QR Code Generation:** QR code linking to sheets
- **Workflow Notifications:** RabbitMQ messaging for status changes
- **File Streaming:** Secure file download/streaming endpoints
- **Audit Logging:** Comprehensive audit trail integration
- **Caching:** Redis caching for frequently accessed sheets

The service is now ready for testing and integration with your existing microservices!