Financial Details Module - Setup Instructions

Directory Structure Created

```
Backend/apps/issuer/
├── __init__.py
├── apps.py
├── admin.py
├── urls.py
├── financials/
│   ├── __init__.py
│   ├── models/
│   │   ├── __init__.py
│   │   ├── base.py
│   │   └── audited_financials.py
│   ├── serializers/
│   │   ├── __init__.py
│   │   └── audited_financials.py
│   ├── views/
│   │   ├── __init__.py
│   │   └── audited_financials.py
│   ├── mixins/
│   │   ├── __init__.py
│   │   ├── pagination_mixin.py
│   │   ├── validation_mixin.py
│   │   └── upload_mixin.py
│   ├── utils/
│   │   ├── __init__.py
│   │   ├── constants.py
│   │   └── validators.py
│   ├── signals/
│   │   ├── __init__.py
│   │   └── handlers.py
│   ├── services/
│   │   ├── __init__.py
│   │   ├── file_service.py
│   │   ├── audit_service.py
│   │   └── validation_service.py
│   └── urls.py
└── migrations/
```

Installation Steps

1. Add App to Settings

Add to `Backend/config/settings/base.py`:

```python
INSTALLED_APPS = [
    # ... existing apps ...
    'apps.issuer',
    # ... other apps ...
]
```

2. Configure Media Files

Add to `Backend/config/settings/base.py`:

```python
# Media files configuration
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

# File upload settings
FILE_UPLOAD_MAX_MEMORY_SIZE = 10485760  # 10MB
DATA_UPLOAD_MAX_MEMORY_SIZE = 10485760  # 10MB
```

3. Configure URLs

Update `Backend/config/urls.py`:

```python
```

```
from django.conf import settings
from django.conf.urls.static import static
from django.urls import path, include

urlpatterns = [
    # ... existing patterns ...
    path('', include('apps.issuer.urls', namespace='issuer')),
    # ... other patterns ...
]

# Serve media files in development
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

4. Run Migrations

```bash
cd Backend

# Create migrations
python manage.py makemigrations issuer

# Apply migrations
python manage.py migrate issuer

# Verify migrations
python manage.py showmigrations issuer
```

5. Create Superuser (if not exists)

```bash
python manage.py createsuperuser
```

6. Collect Static Files (Production)

```bash
python manage.py collectstatic --noinput
```

Database Indexes Created

The following indexes are automatically created:

1. **AuditedFinancial**:
    - `idx_company_fy`: company_id + financial_year
    - `idx_company_status`: company_id + status
    - `idx_status_del`: status + is_del
    - `idx_created_del`: created_at + is_del
    - `idx_audit_date`: audit_report_date
    - Unique constraint: company_id + financial_year + is_del
2. **FinancialDocument**:
    - `idx_fin_doctype`: financial + document_type
    - `idx_doctype_del`: document_type + is_del
    - `idx_uploaded_at`: uploaded_at
    - Unique constraint: financial + document_type + is_del
3. **FinancialAuditLog**:
    - `idx_fin_action`: financial + action
    - `idx_user_time`: user_id + timestamp
    - `idx_log_timestamp`: timestamp
    - `idx_action_time`: action + timestamp
4. **DocumentValidation**:
    - `idx_fin_valstatus`: financial + validation_status
    - `idx_validated_at`: validated_at
5. **UploadSession**:
    - `idx_comp_status`: company_id + status
    - `idx_status_expires`: status + expires_at
    - `idx_started_at`: started_at

API Endpoints Available

Audited Financials APIs

Method	Endpoint	Description
POST	`/api/v1/company/{company_id}/financials/audited/upload`	Upload audited financials
GET	`/api/v1/company/{company_id}/financials/audited`	List all audited financials
GET	`/api/v1/financials/audited/{financial_id}`	Get financial details
PUT	`/api/v1/financials/audited/{financial_id}`	Update financial details
PUT	`/api/v1/financials/audited/{financial_id}/status`	Update status
POST	`/api/v1/financials/audited/{financial_id}/validate`	Validate documents
GET	`/api/v1/financials/audited/{financial_id}/documents/{document_type}`	Download document
PUT	`/api/v1/financials/audited/{financial_id}/documents/{document_type}`	Replace document
GET	`/api/v1/financials/audited/{financial_id}/audit-log`	Get audit trail
DELETE	`/api/v1/financials/audited/{financial_id}`	Soft delete financial
DELETE	`/api/v1/company/{company_id}/financials/audited/bulk-delete`	Bulk delete

Testing the APIs

1. Upload Audited Financial

```bash
curl -X POST \
  http://localhost:8000/api/v1/company/101/financials/audited/upload \
  -H 'Content-Type: multipart/form-data' \
  -F 'financial_year=FY 2022-23' \
  -F 'audited_financial_statement=@/path/to/file.pdf' \
  -F 'itr_filing=@/path/to/itr.pdf' \
  -F 'auditor_name=M/s XYZ & Associates' \
  -F 'audit_report_date=2023-03-31' \
  -F 'user_id_updated_by=1'
```

2. List Audited Financials

```bash
curl -X GET \
  'http://localhost:8000/api/v1/company/101/financials/audited?page=1&limit=10&financial_year=FY 2022-23'
```

3. Get Financial Details

```bash
curl -X GET \
  http://localhost:8000/api/v1/financials/audited/1
```

4. Update Financial

```bash
curl -X PUT \
  http://localhost:8000/api/v1/financials/audited/1 \
  -H 'Content-Type: application/json' \
  -d '{
    "auditor_name": "M/s XYZ & Associates LLP",
    "remarks": "Updated auditor name",
    "user_id_updated_by": 1
  }'
```

5. Validate Documents

```bash
curl -X POST \
  http://localhost:8000/api/v1/financials/audited/1/validate \
  -H 'Content-Type: application/json' \
  -d '{
    "validation_type": "AUTO",
    "user_id": 1
  }'
```

Performance Optimization Features

1. Query Optimization

- Uses `select_related()` and `prefetch_related()` to avoid N+1 queries
- Proper database indexing on frequently queried fields
- Efficient pagination with page size limits

2. File Handling

- Chunked file uploads for large files

- File hash calculation for integrity checks
- Efficient file storage with organized directory structure

3. Audit Trail

- Asynchronous audit log creation (can be moved to Celery tasks)
- Indexed fields for fast audit log queries
- JSON fields for flexible change tracking

4. Soft Delete

- All records use soft delete for data recovery
- Deleted records excluded from queries by default
- Separate indexes for active/deleted records

Security Features

1. File Upload Security

- File size validation (10MB limit)
- File type validation (PDF only)
- File integrity checks (SHA-256 hash)
- Virus scanning hooks (in signals)

2. Access Control

- User ID tracking for all operations
- IP address logging for audit trail
- TODO: Add authentication middleware
- TODO: Add permission checks

3. Data Privacy

- Soft delete for data retention
- Audit trail for compliance
- File encryption support (TODO)

Future Enhancements

1. Authentication & Authorization

python

```python
# Add to views:
from rest_framework.permissions import IsAuthenticated
from config.authentication import CustomJWTAuthentication

class AuditedFinancialUploadView(APIView):
    authentication_classes = [CustomJWTAuthentication]
    permission_classes = [IsAuthenticated]
    # ... rest of the code
```

2. Rate Limiting

python

```python
# Add to views:
from rest_framework.throttling import UserRateThrottle

class AuditedFinancialUploadView(APIView):
    throttle_classes = [UserRateThrottle]
    throttle_scope = 'uploads'
    # ... rest of the code
```

3. Celery Tasks

python

```python
# Create tasks for:
# - Async file uploads
# - Document validation
# - Virus scanning
# - Thumbnail generation
# - Audit log creation
```

4. Cloud Storage Integration

python

```
# Integrate with AWS S3, Azure Blob, or Google Cloud Storage
# for production file storage with CDN
```

5. WebSocket Support

```python
# Add real-time upload progress tracking
# using Django Channels
```

Monitoring & Logging

1. Add Logging Configuration

```python
# In settings.py
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'file': {
            'level': 'INFO',
            'class': 'logging.FileHandler',
            'filename': 'financials.log',
        },
    },
    'loggers': {
        'apps.issuer.financials': {
            'handlers': ['file'],
            'level': 'INFO',
            'propagate': True,
        },
    },
}
```

2. Add Performance Monitoring

```python
# Use Django Debug Toolbar for development
# Use APM tools (New Relic, DataDog) for production
```

Common Issues & Solutions

Issue 1: Migration Conflicts

Solution: Reset migrations if needed

```bash
python manage.py migrate issuer zero
python manage.py makemigrations issuer
python manage.py migrate issuer
```

Issue 2: File Upload Errors

Solution: Check media directory permissions

```bash
chmod -R 755 Backend/media
chown -R www-data:www-data Backend/media
```

Issue 3: Database Performance

Solution: Add additional indexes if needed

```python
# In models.py
class Meta:
    indexes = [
        models.Index(fields=['custom_field']),
    ]
```

Next Steps

1. **Implement Remaining Tables**:
    - Provisional GST Returns
    - Signatory Details
    - Upload Session Management APIs
2. **Add Third-Party Integrations**:
    - MCA API for DIN verification
    - Income Tax API for PAN verification

- Aadhaar verification service

3. **Add Testing**:
   - Unit tests for models
   - Integration tests for APIs
   - Load testing for file uploads
4. **Documentation**:
   - API documentation with Swagger/OpenAPI
   - Developer guide
   - User guide