DU Compliance Agent MVP - Solution Architecture Documentation

Table of Contents

Executive Summary

The DU Compliance Agent MVP is an intelligent document compliance verification system designed to automate the process of checking telecommunications and domain service documents against regulatory requirements. The solution leverages Large Language Models (LLMs) to analyze uploaded documents and identify compliance issues in real-time, providing actionable insights to ensure regulatory adherence.

Key Highlights

- **Automated Compliance Verification**: AI-powered analysis of documents against multiple regulations simultaneously
- **Real-Time Processing**: Streaming architecture provides immediate feedback during document analysis
- **Intelligent Document Validation**: Document gatekeeper ensures only relevant documents are processed
- **Multi-Format Support**: Handles PDF, DOCX, TXT, and Markdown files
- **User-Friendly Interface**: Modern React-based frontend with drag-and-drop file upload
- **Scalable Architecture**: FastAPI backend with async processing capabilities

Target Users

- **Compliance Officers**: Verify marketing materials, service plans, and policy documents
- **Legal Teams**: Ensure regulatory compliance before document publication
- **Product Managers**: Validate product descriptions and service offerings
- **Business Analysts**: Review documentation for compliance gaps

---

Business Value & ROI

Business Benefits

1. **Time Savings**: Reduces manual compliance review time from hours to minutes

- Traditional manual review: 2-4 hours per document
- Automated verification: 2-5 minutes per document
- **Time savings: 95%+**

2. **Cost Reduction**: Minimizes need for extensive legal/compliance team review
   - Reduces dependency on specialized compliance reviewers
   - Enables faster time-to-market for new products and services
   - Lowers risk of non-compliance penalties
3. **Consistency**: Ensures uniform compliance checking across all documents
   - Eliminates human error and oversight
   - Provides standardized evaluation criteria
   - Maintains audit trail of compliance checks
4. **Risk Mitigation**: Proactively identifies compliance issues before publication
   - Prevents regulatory violations
   - Reduces legal exposure
   - Protects brand reputation
5. **Scalability**: Handles multiple documents and regulations simultaneously
   - Processes multiple files in parallel
   - Checks against multiple regulations in one pass
   - Scales with business growth

Return on Investment

- **Efficiency Gains**: 95% reduction in compliance review time
- **Risk Avoidance**: Prevents costly regulatory penalties and legal issues
- **Resource Optimization**: Frees compliance team for strategic work
- **Faster Time-to-Market**: Accelerates product launches with confidence

---

Use Cases & Scenarios

Use Case 1: Marketing Material Compliance Review

Scenario: Marketing team creates a new service plan brochure that needs compliance verification before publication.

Process:

1. Marketing team uploads the brochure (PDF/DOCX)
2. Selects relevant regulations (e.g., Consumer Protection Regulations, Code of Practice)
3. System validates document relevance
4. System checks compliance against selected regulations
5. Team receives detailed compliance report with issues and recommendations
6. Team addresses issues and re-uploads for verification

Outcome: Marketing materials are compliant before publication, reducing risk of regulatory violations.

Use Case 2: Service Policy Document Review

Scenario: Legal team needs to verify a new terms of service document against multiple domain and telecom regulations.

Process:

1. Legal team uploads terms of service document
2. Selects all relevant regulations (Domain Name policies, Consumer Protection, etc.)
3. System processes document and identifies compliance gaps
4. Legal team reviews critical issues first
5. Team updates document based on recommendations
6. Re-verification confirms compliance

Outcome: Comprehensive compliance verification ensures all regulatory requirements are met.

Use Case 3: Product Feature Documentation Review

Scenario: Product team creates documentation for a new telecom service feature and needs to ensure compliance.

Process:

1. Product team uploads feature documentation
2. System validates document is relevant to telecom services
3. System checks against relevant regulations
4. Product team receives prioritized list of compliance issues
5. Team addresses critical issues before launch

Outcome: New features are compliant from launch, avoiding post-launch compliance issues.

Use Case 4: Batch Document Review

Scenario: Compliance team needs to review multiple documents for quarterly compliance audit.

Process:

1. Team uploads multiple documents simultaneously
2. System validates each document
3. Invalid documents are flagged with reasons
4. Valid documents are processed against selected regulations
5. Team receives comprehensive compliance report for all documents

Outcome: Efficient batch processing enables comprehensive compliance audits.

Key Features & Capabilities

Core Features

1. **Multi-Format Document Support**
    - PDF documents (text extraction)
    - DOCX/DOC files (Word documents)
    - TXT files (plain text)
    - Markdown files (direct processing)
2. **Intelligent Document Validation**
    - Document gatekeeper validates relevance before processing
    - Confidence scoring (0-100) for validation decisions
    - Configurable confidence threshold (default: 70%)
    - Automatic rejection of irrelevant documents
3. **Multi-Regulation Compliance Checking**
    - Support for 11+ UAE telecom and domain regulations
    - Selective regulation checking (users choose which regulations to check)
    - Parallel processing of multiple regulations
    - Real-time progress updates
4. **Real-Time Streaming Updates**
    - Live progress updates during processing
    - Incremental issue reporting as regulations are checked
    - Status indicators for each processing stage
    - Error handling with user-friendly messages
5. **Detailed Compliance Reporting**
    - Issue identification with exact regulation clauses
    - Criticality assessment (Critical/Medium/Low)
    - Evidence extraction from documents
    - Actionable recommendations
    - Links to source regulation PDFs
6. **User-Friendly Interface**
    - Drag-and-drop file upload
    - Regulation selection with checkboxes
    - Real-time processing status
    - Interactive compliance issue table
    - Responsive design for all devices

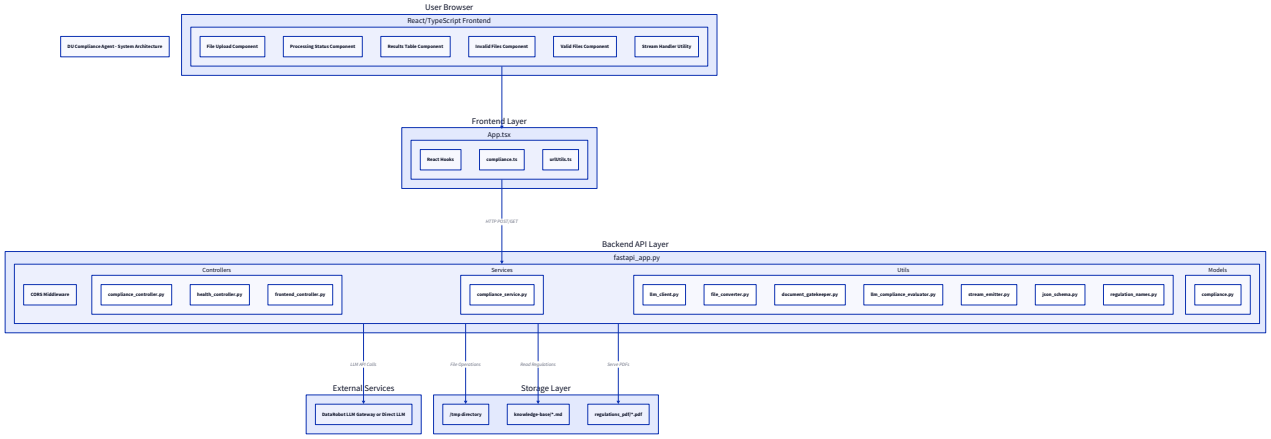
Advanced Capabilities

- **Async Processing**: Non-blocking file processing for better performance
- **Error Recovery**: Graceful handling of processing errors
- **File Cleanup**: Automatic cleanup of temporary files
- **Streaming Architecture**: Server-Sent Events (SSE) for real-time updates

- **LLM Integration**: Flexible LLM backend (DataRobot Gateway or direct LLM)

System Architecture

High-Level System Architecture



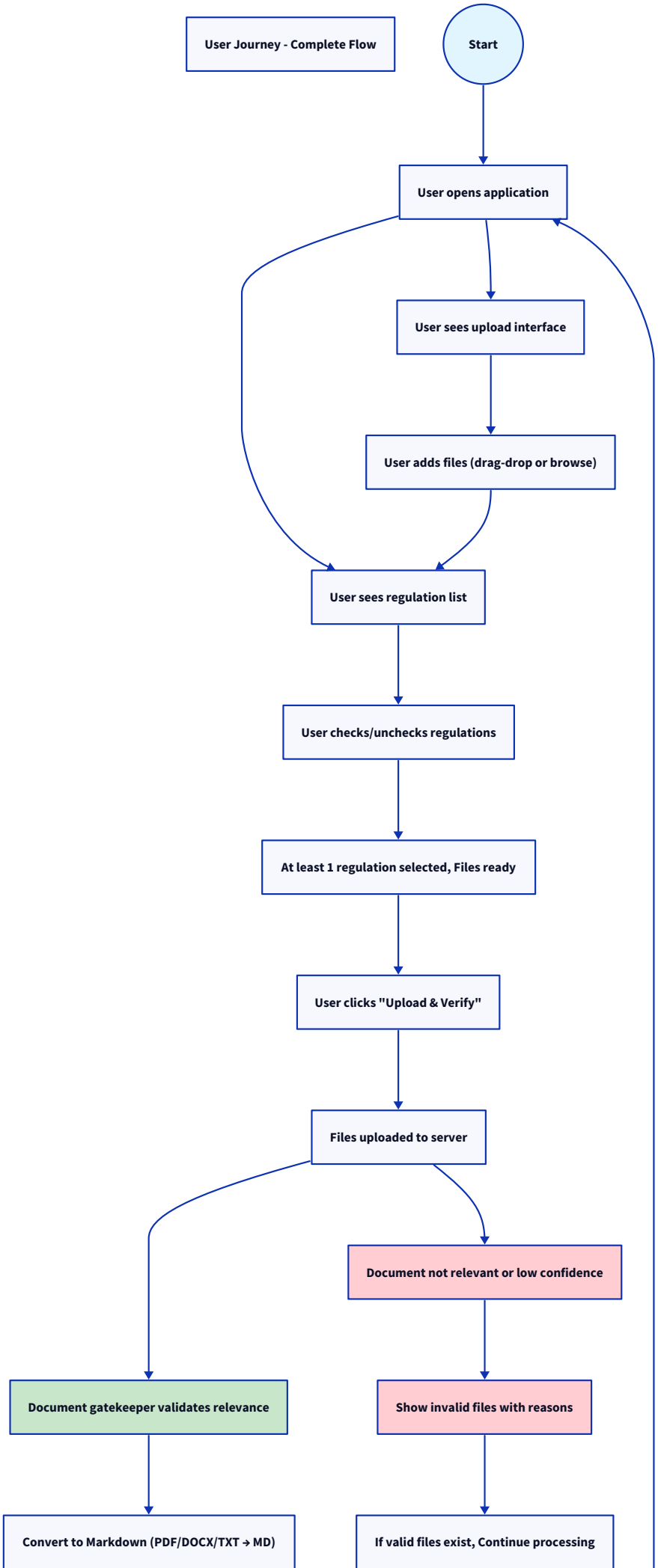
Architecture Layers

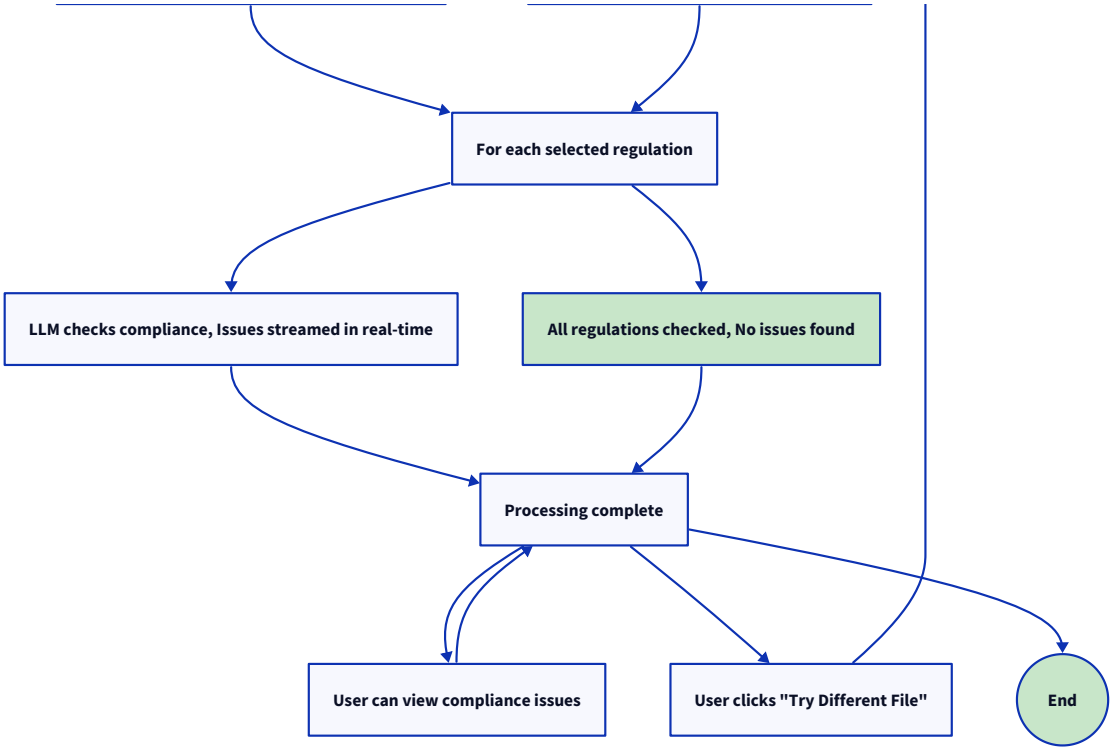
1. **Presentation Layer**: React/TypeScript frontend providing user interface
2. **API Layer**: FastAPI backend handling requests and orchestration
3. **Service Layer**: Business logic for compliance verification
4. **Integration Layer**: LLM client for AI-powered analysis
5. **Storage Layer**: Temporary file storage and knowledge base

---

User Journey

Complete User Journey Flow


User Journey - Complete Flow
Start
User opens application
User sees upload interface
User adds files (drag-drop or browse)
User sees regulation list
User checks/unchecks regulations
At least 1 regulation selected, Files ready
User clicks "Upload & Verify"
Files uploaded to server
Document not relevant or low confidence
Document gatekeeper validates relevance
Show invalid files with reasons
Convert to Markdown (PDF/DOCX/TXT → MD)
If valid files exist, Continue processing

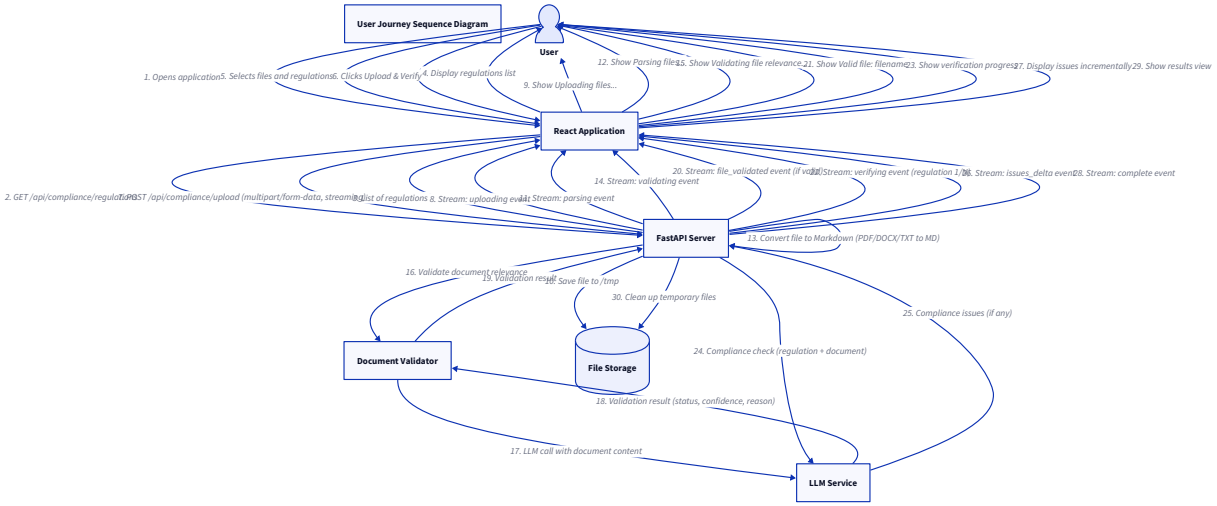


User Journey States

The application operates in four main states:

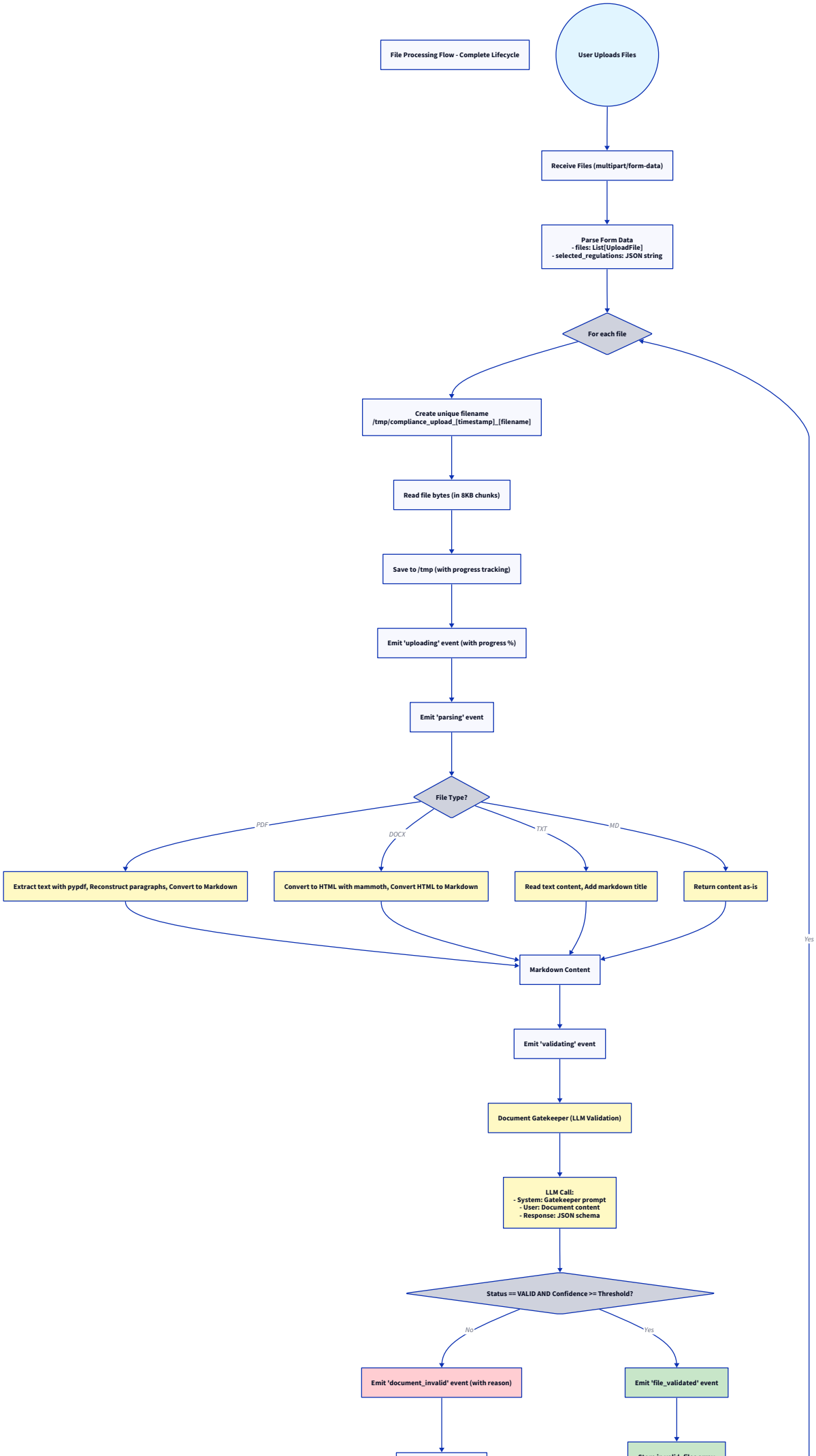
1. **Upload State**: User selects files and regulations
2. **Processing State**: Files are being validated and checked
3. **Results State**: Compliance issues are displayed
4. **Invalid State**: Invalid files are shown with reasons

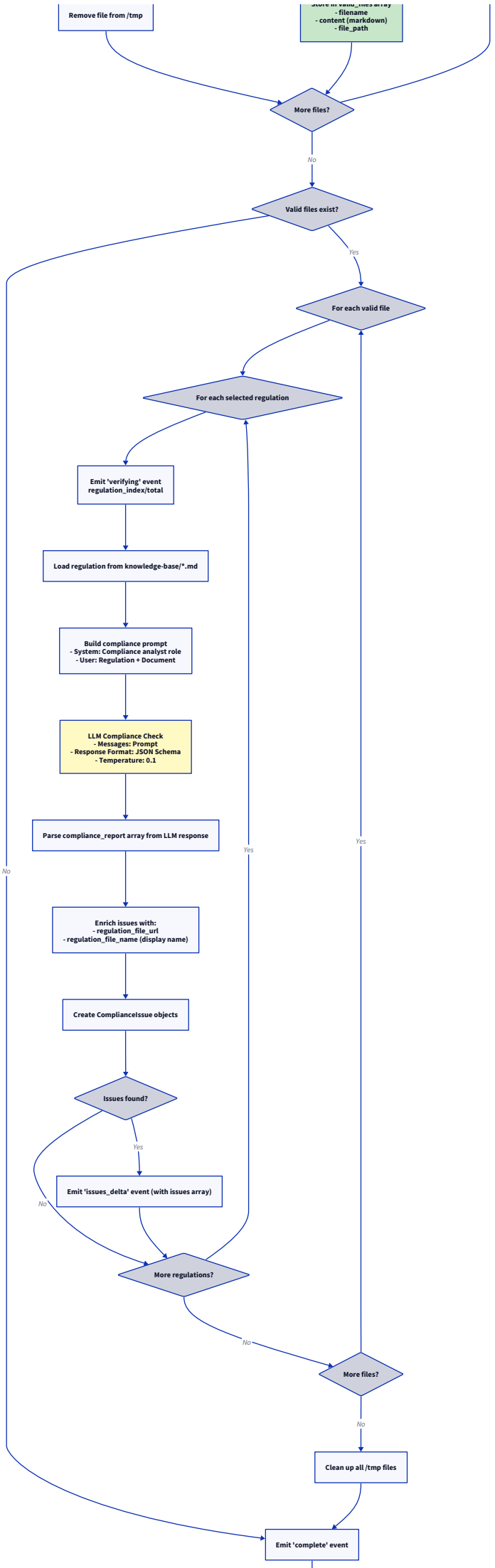
User Journey Sequence

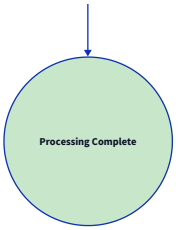


File Processing Flow

Detailed File Processing Flow Diagram


File Processing Flow - Complete Lifecycle
User Uploads Files
Receive Files (multipart/form-data)
Parse Form Data
- files: List[UploadFile]
- selected_regulations: JSON string
For each file
Create unique filename
/tmp/compliance_upload_[timestamp]_[filename]
Read file bytes (in 8KB chunks)
Save to /tmp (with progress tracking)
Emit 'uploading' event (with progress %)
Emit 'parsing' event
File Type?
PDF
DOCX
TXT
MD
Extract text with pypdf, Reconstruct paragraphs, Convert to Markdown
Convert to HTML with mammoth, Convert HTML to Markdown
Read text content, Add markdown title
Return content as-is
Markdown Content
Emit 'validating' event
Document Gatekeeper (LLM Validation)
LLM Call:
- System: Gatekeeper prompt
- User: Document content
- Response: JSON schema
Status == VALID AND Confidence >= Threshold?
No
Yes
Emit 'document_invalid' event (with reason)
Emit 'file_validated' event
Yes


Remove file from /tmp
- filename
- content (markdown)
- file_path
More files?
No
Valid files exist?
Yes
For each valid file
For each selected regulation
Emit 'verifying' event
regulation_index/total
Load regulation from knowledge-base/*.md
Build compliance prompt
- System: Compliance analyst role
- User: Regulation + Document
LLM Compliance Check
- Messages: Prompt
- Response Format: JSON Schema
- Temperature: 0.1
Parse compliance_report array from LLM response
Enrich issues with:
- regulation_file_url
- regulation_file_name (display name)
Create ComplianceIssue objects
Issues found?
Yes
No
Emit 'issues_delta' event (with issues array)
More regulations?
Yes
No
Yes
More files?
Yes
No
No
Clean up all /tmp files
Emit 'complete' event



File Processing Stages

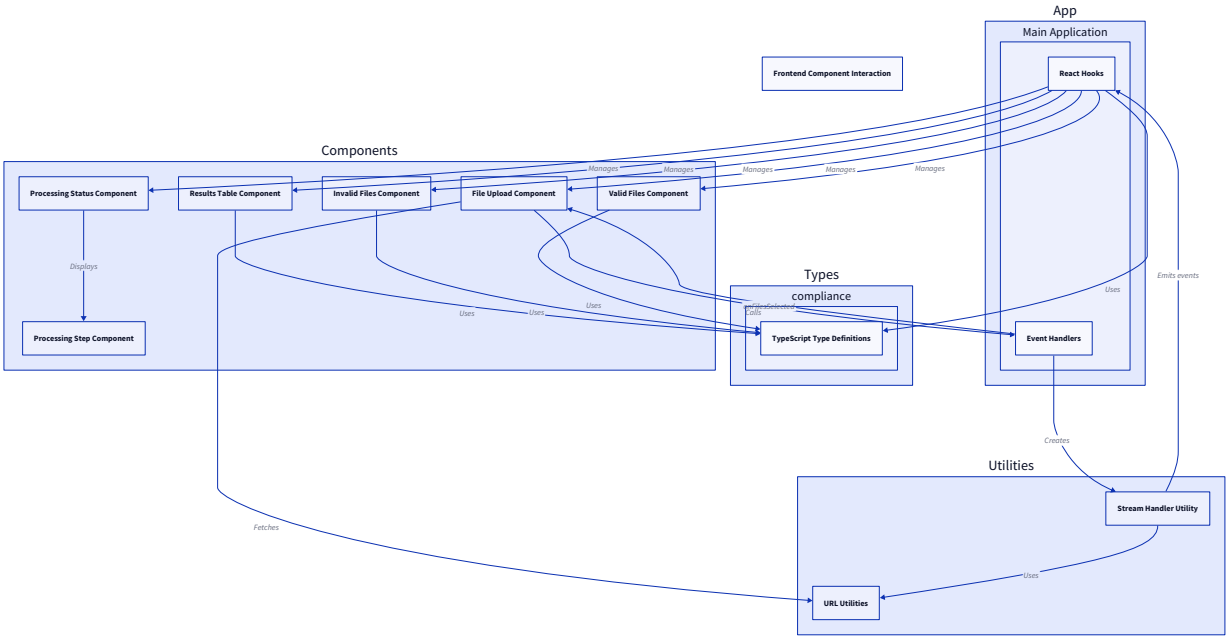
1. **Upload Stage**: Files are received and saved to temporary storage
2. **Parsing Stage**: Files are converted to Markdown format
3. **Validation Stage**: Document gatekeeper validates relevance
4. **Verification Stage**: Compliance checking against selected regulations
5. **Aggregation Stage**: Issues are collected and streamed to frontend
6. **Cleanup Stage**: Temporary files are removed

File Type Conversion

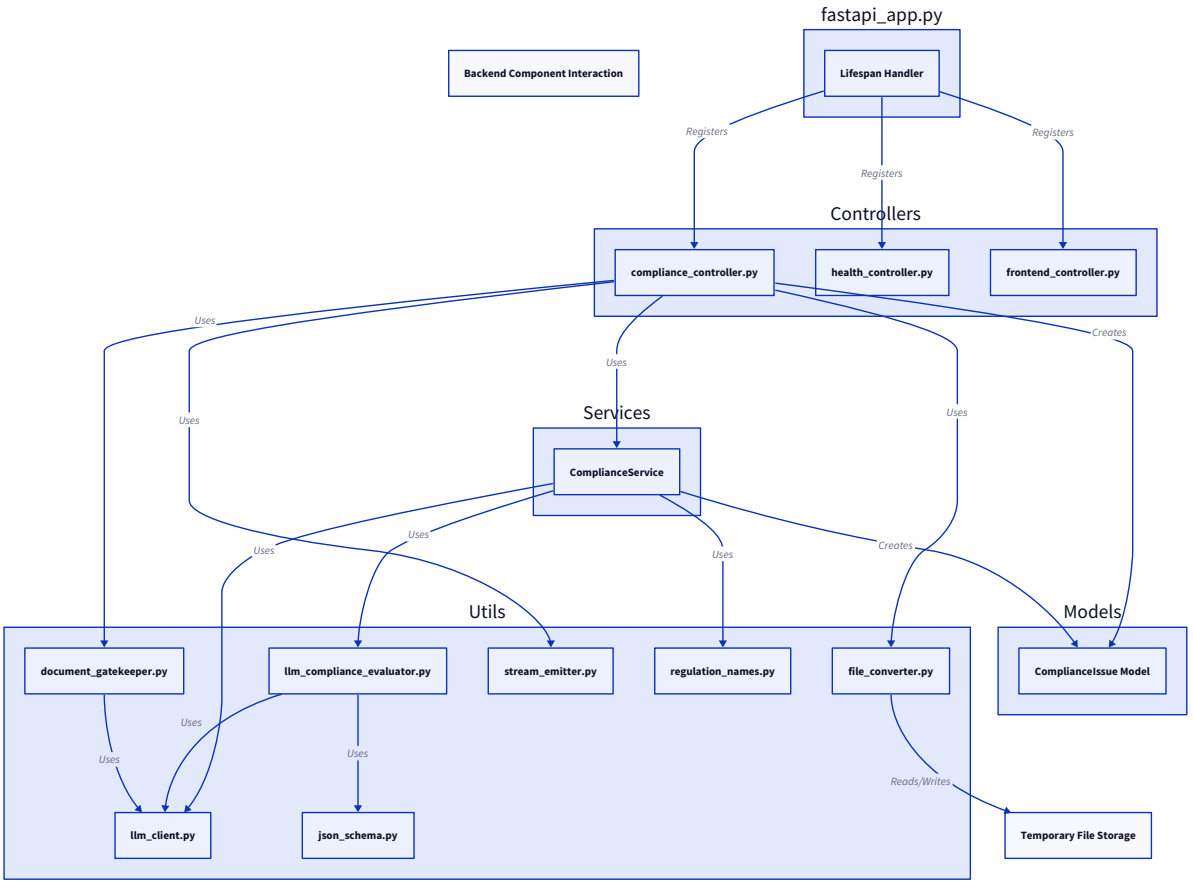
- **PDF**: Text extraction using pypdf, paragraph reconstruction, Markdown formatting
- **DOCX**: HTML conversion using mammoth, HTML-to-Markdown conversion
- **TXT**: Direct text reading with Markdown title addition
- **MD**: Passthrough (no conversion needed)

Component Interaction

Frontend Component Interaction

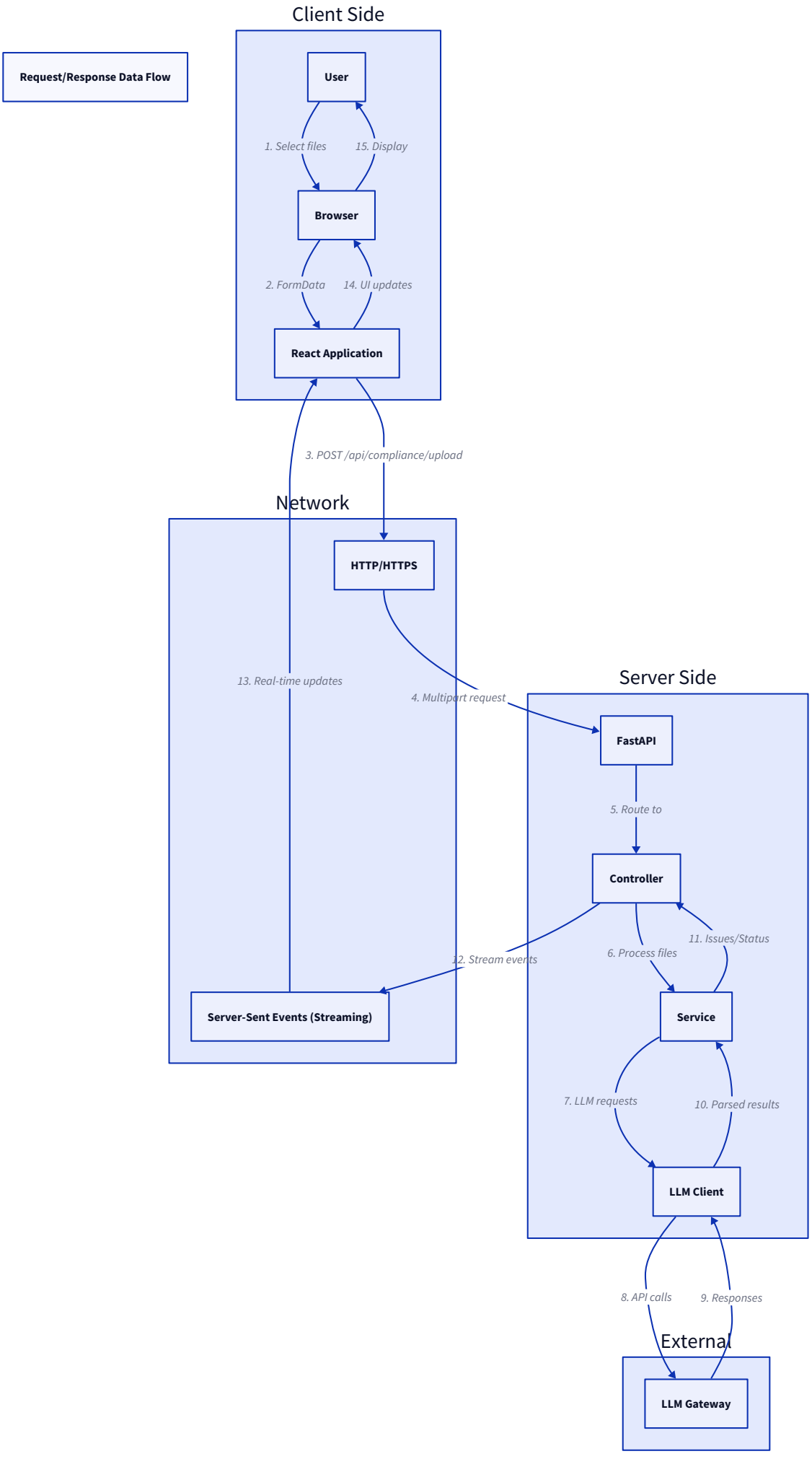


Backend Component Interaction

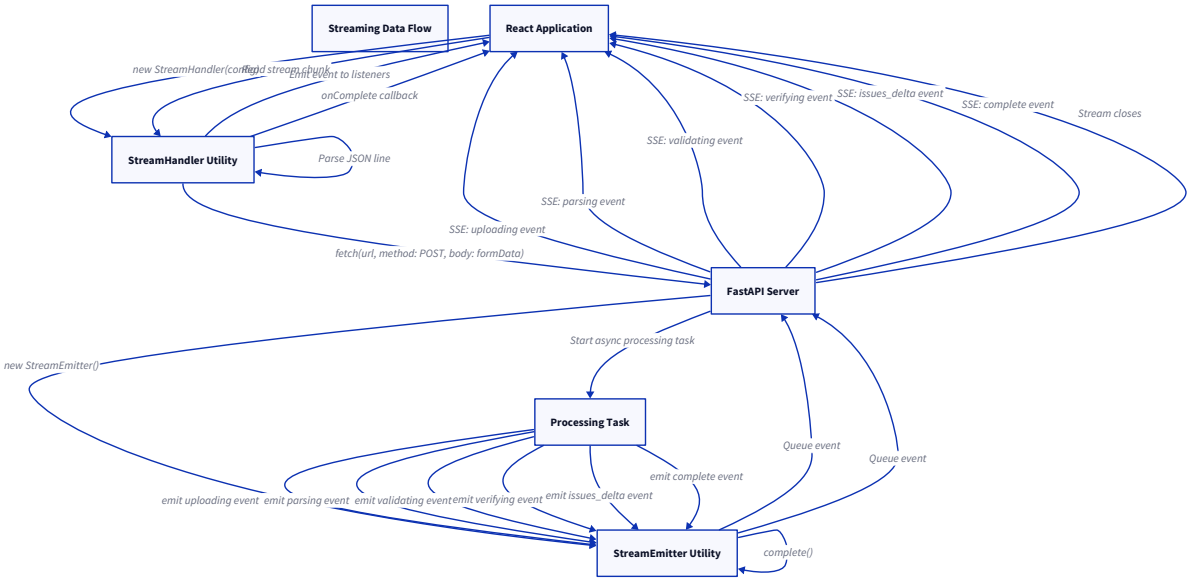


Data Flow

Request/Response Data Flow

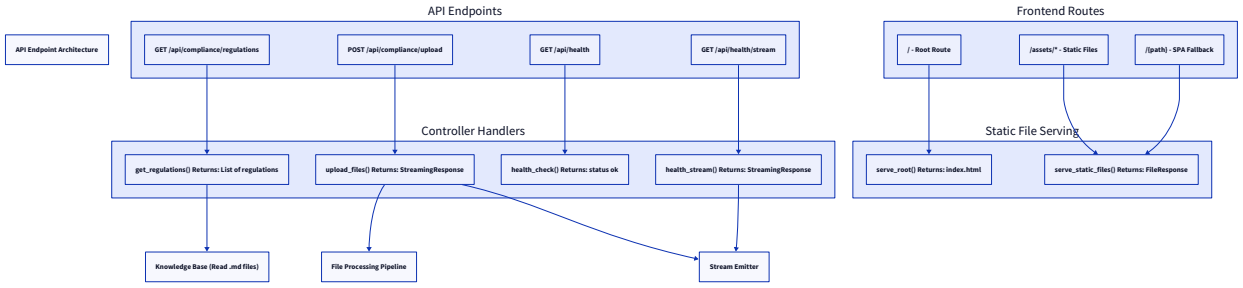

Request/Response Data Flow
Client Side
User
1. Select files
15. Display
Browser
2. FormData
14. UI updates
React Application
3. POST /api/compliance/upload
Network
HTTP/HTTPS
13. Real-time updates
4. Multipart request
Server-Sent Events (Streaming)
Server Side
FastAPI
5. Route to
Controller
12. Stream events
6. Process files
11. Issues/Status
Service
7. LLM requests
10. Parsed results
LLM Client
8. API calls
9. Responses
External
LLM Gateway

Streaming Data Flow

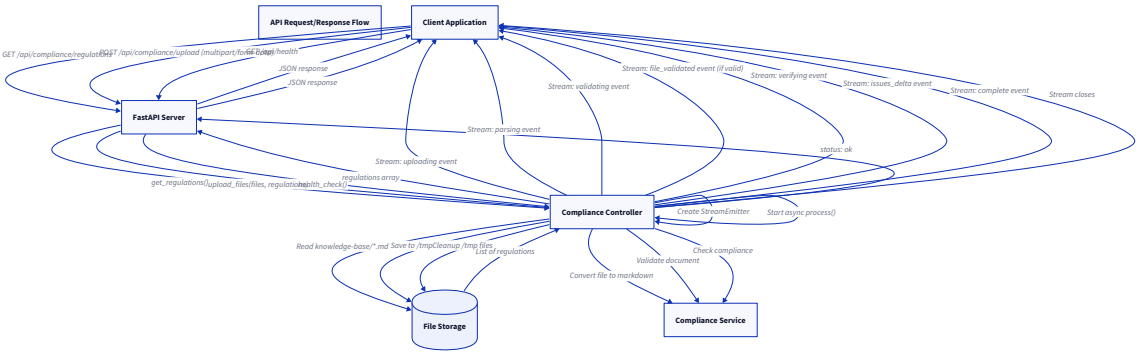


API Endpoints

API Endpoint Architecture



API Request/Response Flow



API Endpoint Details

GET /api/compliance/regulations

Description: Get list of available regulations with their display names.

Request: None

Response:

```
{
  "regulations": [
    {
      "filename": "AEDAPOL001v11CommonDefinitions.md",
      "displayName": "Common Definitions Policy"
    },
    ...
  ]
}
```

POST /api/compliance/upload

Description: Upload and process files for compliance verification. Returns a streaming response with status updates.

Request:

- `files`: List of uploaded files (multipart/form-data)
- `selected_regulations`: Optional JSON string of selected regulation filenames

Response: Streaming response (Server-Sent Events) with events:

- `uploading`: File upload progress
- `parsing`: File parsing status
- `validating`: Document validation status
- `file_validated`: File passed validation
- `document_invalid`: File failed validation
- `verifying`: Compliance verification progress
- `issues_delta`: Incremental compliance issues
- `complete`: Processing complete
- `error`: Error occurred

GET /api/health

Description: Health check endpoint.

Request: None

Response:

```
{
  "status": "ok"
}
```

GET /api/health/stream

Description: Streaming health check endpoint for testing.

Request: None

Response: Streaming response with keep-alive events.

Technology Stack

Frontend Technology Stack

Technology	Version	Purpose
React	^19.1.1	UI framework
TypeScript	~5.9.3	Type safety
Vite	^7.1.7	Build tool and dev server
TailwindCSS	^4.1.14	Styling
React Icons	^5.5.0	Icon library
React Markdown	^10.1.0	Markdown rendering
Recharts	^3.3.0	Charts (if used)
Zustand	^5.0.8	State management (if used)

Backend Technology Stack

Technology	Version	Purpose
Python	3.x	Programming language
FastAPI	0.119.0	Web framework
Uvicorn	0.37.0	ASGI server
OpenAI	1.109.1	LLM client library
pypdf	5.3.0	PDF parsing
mammoth	1.7.1	DOCX to HTML conversion
markdownify	0.14.1	HTML to Markdown conversion
python-dotenv	1.1.1	Environment variable management
python-multipart	0.0.20	File upload handling
sse-starlette	3.0.2	Server-Sent Events support

External Services

Service	Purpose	Configuration
DataRobot LLM Gateway	LLM API access	`DATAROBOT_ENDPOINT`, `DATAROBOT_API_TOKEN`
Direct LLM	Alternative LLM access	`DATAROBOT_DEPLOYED_LLM_URL`
Temporary Storage	File processing	`/tmp` directory

Component Details

Frontend Components

App.tsx

- **Purpose**: Main application component managing overall state and routing
- **State Management**:
  - `state`: 'upload' | 'processing' | 'results' | 'invalid'
  - `processingSteps`: Array of processing step messages
  - `complianceIssues`: Array of compliance issues
  - `invalidFiles`: Array of invalid files with reasons
  - `validFiles`: Array of valid file names
  - `showProcessingSidebar`: Boolean for sidebar visibility
  - `isSidebarMinimized`: Boolean for sidebar state

FileDropZone Component

- **Purpose**: File upload interface with regulation selection
- **Features**:
  - Drag-and-drop file upload
  - File browser selection
  - Regulation checkbox selection
  - File type validation (PDF, DOCX, DOC, TXT, MD)
  - File list management
  - Regulation list fetching from API

ProcessingView Component

- **Purpose**: Display processing status and steps
- **Features**:
  - Real-time step updates
  - Processing status indicators
  - Step history display

ComplianceTable Component

- **Purpose**: Display compliance issues in tabular format
- **Features**:
  - Issue listing with details
  - Regulation links
  - Criticality indicators

    - Evidence and recommendations display

InvalidDocumentView Component

- **Purpose**: Display invalid files with rejection reasons
- **Features**:
    - Invalid file list
    - Rejection reason display
    - Option to try different files

ValidFilesSection Component

- **Purpose**: Display valid files and their processing status
- **Features**:
    - Valid file list
    - Processing status
    - Results navigation

StreamHandler Utility

- **Purpose**: Handle streaming responses from backend
- **Features**:
    - Event-based stream parsing
    - Type-safe event handling
    - Error handling
    - Connection management
    - Event listener registration

Backend Components

FastAPI Application (fastapi_app.py)

- **Purpose**: Main application entry point
- **Features**:
    - CORS middleware configuration
    - Controller auto-loading
    - Lifespan management
    - Script name support for DataRobot deployments

Compliance Controller

- **Endpoints**:
    - `GET /api/compliance/regulations`: Get list of available regulations
    - `POST /api/compliance/upload`: Upload and process files (streaming)
- **Features**:
    - File upload handling
    - File validation pipeline
    - Compliance verification orchestration
    - Streaming response management
    - Error handling

Compliance Service

- **Purpose**: Business logic for compliance verification
- **Methods**:
  - `verify_against_regulations()`: Main verification method
  - `_verify_single_regulation()`: Verify against single regulation
- **Features**:
  - Regulation filtering
  - Progress callbacks
  - Issue aggregation
  - Regulation URL generation

LLM Client

- **Purpose**: Create and configure LLM client
- **Modes**:
  - `dr-gateway`: DataRobot LLM Gateway mode
  - `direct-llm`: Direct LLM endpoint mode
- **Configuration**: Environment variables for endpoints and authentication

File Converter

- **Purpose**: Convert various file formats to Markdown
- **Supported Formats**:
  - PDF → Markdown (pypdf extraction)
  - DOCX → Markdown (mammoth + markdownify)
  - TXT → Markdown (direct with title)
  - MD → Markdown (passthrough)

Document Gatekeeper

- **Purpose**: Validate document relevance before processing
- **Features**:
  - LLM-based validation
  - Confidence scoring (0-100)
  - Configurable threshold (default: 70)
  - Structured JSON response

Compliance Evaluator

- **Purpose**: Evaluate compliance using LLM
- **Features**:
  - Regulation + document analysis
  - Structured JSON output
  - Issue extraction and parsing
  - Multiple response format fallbacks

Stream Emitter

- **Purpose**: Emit streaming events to frontend
- **Features**:

- Async event queue
- NDJSON format
- Keep-alive support
- Error handling
- Task management

ComplianceIssue Model

- **Fields**:
  - `regulation_file_name`: Display name of regulation
  - `regulation_file_url`: URL to PDF regulation
  - `regulation_clause_section`: Section identifier
  - `regulation_clause_text`: Exact clause text
  - `cvp_evidence`: Evidence from document
  - `recommended_criticality`: Critical/Medium/Low
  - `explanation`: Issue explanation
  - `recommendation`: Recommended action

Deployment Architecture

Deployment Options

Option 1: DataRobot Custom Application

The application can be deployed as a DataRobot Custom Application, which provides:

- Integrated hosting within DataRobot platform
- Automatic scaling and load balancing
- Built-in authentication and security
- Easy integration with DataRobot LLM Gateway

Configuration:

- Set `SCRIPT_NAME` environment variable to the custom application path
- Configure DataRobot API credentials
- Deploy frontend build to `app/frontend_dist/`

Option 2: Standalone Server Deployment

The application can be deployed as a standalone FastAPI server:

Requirements:

- Python 3.x environment
- Uvicorn ASGI server
- Frontend build served as static files

Deployment Steps:

1. Build frontend: `cd frontend && npm run build`

2. Copy build to `server/app/frontend_dist/`
3. Install Python dependencies: `pip install -r requirements.txt`
4. Configure environment variables
5. Run server: `uvicorn fastapi_app:app --host 0.0.0.0 --port 8000`

Option 3: Docker Deployment

The application can be containerized using Docker:

Dockerfile Structure:

- Base Python image
- Install dependencies
- Copy application code
- Build and serve frontend
- Expose port 8000

Deployment:

```
docker build -t du-compliance-agent .
docker run -p 8000:8000 --env-file .env du-compliance-agent
```

Infrastructure Requirements

- **Compute**: Minimum 2 CPU cores, 4GB RAM
- **Storage**: 10GB for temporary file storage
- **Network**: HTTPS endpoint for API access
- **External Services**: LLM Gateway or Direct LLM endpoint

Security & Compliance

Security Measures

1. **File Upload Security**:
    - File type validation
    - File size limits (configurable)
    - Temporary file storage with automatic cleanup
    - No persistent storage of uploaded documents
2. **API Security**:
    - CORS configuration for allowed origins
    - Environment variable-based configuration
    - No sensitive data in logs
    - Secure file handling
3. **LLM Integration Security**:

    - API token authentication
    - Secure endpoint communication (HTTPS)
    - No data persistence in LLM calls
    - Configurable LLM endpoints

4. **Data Privacy**:
    - Files processed in memory when possible
    - Temporary files cleaned up immediately after processing
    - No long-term storage of document content
    - Compliance issues stored only in user session

Compliance Considerations

- **Regulatory Compliance**: System helps ensure compliance with UAE telecom and domain regulations
- **Data Handling**: Temporary file storage with automatic cleanup
- **Audit Trail**: Processing events logged for audit purposes
- **Error Handling**: Graceful error handling prevents data leakage

---

Integration Points

LLM Integration

The system integrates with LLM services through two modes:

1. **DataRobot LLM Gateway Mode**:
    - Endpoint: `{DATAROBOT_ENDPOINT}/genai/llmgw`
    - Authentication: DataRobot API token
    - Model: Configurable via `CHAT_COMPLETIONS_MODEL`

2. **Direct LLM Mode**:
    - Endpoint: `DATAROBOT_DEPLOYED_LLM_URL`
    - Authentication: Configurable
    - Model: Specified in endpoint configuration

Knowledge Base Integration

- **Regulation Storage**: Markdown files in `app/knowledge-base/`
- **PDF Storage**: PDF files in `app/regulations_pdf/`
- **Regulation Mapping**: Display names mapped in `regulation_names.py`

Frontend Integration

- **API Endpoints**: RESTful API with streaming support
- **Static File Serving**: Frontend build served from FastAPI

- **Real-Time Updates**: Server-Sent Events for streaming

Environment Configuration

Required Environment Variables

Variable	Purpose	Example	Required
`MODE`	LLM connection mode	`dr-gateway` or `direct-llm`	Yes
`DATAROBOT_ENDPOINT`	DataRobot API endpoint	`https://app.datarobot.com/api/v2`	Yes (if MODE=dr-gateway)
`DATAROBOT_API_TOKEN`	API authentication token	`your-token-here`	Yes (if MODE=dr-gateway)
`DATAROBOT_DEPLOYED_LLM_URL`	Direct LLM URL	`https://your-llm-endpoint.com`	Yes (if MODE=direllm)
`CHAT_COMPLETIONS_MODEL`	LLM model name	`gpt-4o-mini`	No (default gpt-4o-min
`GATEKEEPER_CONFIDENCE_THRESHOLD`	Validation confidence threshold	`70` (0-100)	No (default 70)
`SCRIPT_NAME`	Base path for DataRobot deployments	`/custom_applications/{appId}`	No

Environment Setup

Create a `.env` file in the server directory:

```
MODE=dr-gateway
DATAROBOT_ENDPOINT=https://app.datarobot.com/api/v2
DATAROBOT_API_TOKEN=your-token-here
CHAT_COMPLETIONS_MODEL=gpt-4o-mini
GATEKEEPER_CONFIDENCE_THRESHOLD=70
SCRIPT_NAME=/custom_applications/your-app-id
```

Error Handling & Edge Cases

Error Handling Strategy

1. **File Upload Errors**:
    - Invalid file types: Rejected with user-friendly message
    - File size limits: Error message displayed
    - Network errors: Retry mechanism with error notification
2. **File Conversion Errors**:
    - Corrupted files: Error logged, file skipped
    - Unsupported formats: Error message displayed
    - Conversion failures: Fallback to error message
3. **Document Validation Errors**:
    - Gatekeeper failures: Document marked as invalid
    - LLM errors: Error logged, validation skipped
    - Low confidence: Document rejected with reason
4. **Compliance Verification Errors**:
    - LLM API errors: Error logged, regulation skipped
    - Parsing errors: Error logged, issues not included
    - Timeout errors: Error message displayed
5. **Streaming Errors**:
    - Connection drops: Error event emitted
    - Parsing errors: Error logged, stream continues
    - Timeout errors: Stream closed gracefully

Edge Cases Handled

1. **Empty File Upload**: Rejected with validation message
2. **Multiple Invalid Files**: All invalid files displayed with reasons
3. **Partial Processing**: Valid files processed even if some invalid
4. **No Issues Found**: Success message displayed
5. **All Regulations Selected**: All regulations checked
6. **No Regulations Selected**: Error message displayed
7. **Large Files**: Chunked processing with progress updates
8. **Concurrent Uploads**: Sequential processing with queue
9. **LLM Rate Limiting**: Error handling with retry logic
10. **Temporary Storage Full**: Error logged, processing stopped

Error Recovery

- **Automatic Cleanup**: Temporary files cleaned up on error
- **Graceful Degradation**: Partial results returned when possible
- **User Feedback**: Clear error messages for all failure scenarios
- **Logging**: Comprehensive error logging for debugging

File Structure

```
du-compliance-agent-mvp/
├── frontend/
│   ├── src/
│   │   ├── App.tsx                    # Main application
│   │   ├── components/
│   │   │   ├── compliance/
│   │   │   │   ├── FileDropZone.tsx   # File upload UI
│   │   │   │   ├── ProcessingView.tsx # Processing status
│   │   │   │   ├── ComplianceTable.tsx # Results table
│   │   │   │   ├── InvalidDocumentView.tsx
│   │   │   │   └── ValidFilesSection.tsx
│   │   │   └── layout/
│   │   │       ├── PageLayout.tsx
│   │   │       └── Topbar.tsx
│   │   ├── types/
│   │   │   └── compliance.ts          # TypeScript types
│   │   └── utils/
│   │       ├── StreamHandler.ts      # Stream handling
│   │       └── urlUtils.ts           # URL utilities
│   └── package.json
│
├── server/
│   ├── fastapi_app.py                 # FastAPI application
│   ├── main.py                        # Entry point
│   ├── app/
│   │   ├── controllers/
│   │   │   ├── compliance_controller.py
│   │   │   ├── health_controller.py
│   │   │   └── frontend_controller.py
│   │   ├── services/
│   │   │   └── compliance_service.py
│   │   ├── utils/
│   │   │   ├── llm_client.py
│   │   │   ├── file_converter.py
│   │   │   ├── document_gatekeeper.py
│   │   │   ├── llm_compliance_evaluator.py
│   │   │   ├── stream_emitter.py
│   │   │   ├── json_schema.py
│   │   │   └── regulation_names.py
│   │   ├── models/
│   │   │   └── compliance.py
│   │   ├── knowledge-base/            # Markdown regulations
│   │   │   └── *.md
│   │   └── regulations_pdf/          # PDF regulations
│   │       └── *.pdf
│   └── requirements.txt
```

Summary

This architecture documentation provides a comprehensive view of the DU Compliance Agent MVP system, covering:

1. **Business Value**: ROI, use cases, and business benefits
2. **System Architecture**: Complete component hierarchy with D2 diagrams
3. **User Journey**: Detailed user interaction flow from upload to results
4. **File Processing Flow**: Step-by-step file lifecycle with comprehensive D2 diagram
5. **Component Interaction**: How components communicate within and across layers
6. **Data Flow**: Request/response and streaming data flows

7. **API Endpoints**: Complete API documentation with request/response flows
8. **Technology Stack**: All technologies and versions used
9. **Component Details**: Detailed description of each major component
10. **Deployment Architecture**: Deployment options and infrastructure requirements
11. **Security & Compliance**: Security measures and compliance considerations
12. **Integration Points**: External service integrations
13. **Environment Configuration**: Environment variable setup
14. **Error Handling**: Comprehensive error handling and edge case management

The system follows a modern architecture pattern with:

- **Frontend**: React/TypeScript SPA with real-time streaming updates
- **Backend**: FastAPI with async processing and streaming responses
- **External**: LLM Gateway for compliance analysis
- **Storage**: Temporary file storage and knowledge base for regulations

All diagrams are in D2 format and can be rendered using D2 tools or compatible viewers. The documentation is structured to serve both business executives and technical stakeholders, providing comprehensive information for decision-making and implementation.