Resumable File Upload System - Technical Documentation

1. Executive Summary

This system implements a functional resumable file upload solution that meets the core technical assessment requirements. The application successfully handles large file uploads (50MB+) with pause/resume capabilities and survives network interruptions, page refreshes, and system standby modes.

Current Tech Stack:

- **Frontend:** React 18 with Zustand state management
- **Backend:** Node.js with Express framework
- **Storage:** AWS S3 for file chunks, DynamoDB for upload metadata
- **Upload Strategy:** 5MB chunked uploads with progress tracking
- **Persistence:** IndexedDB for file storage, localStorage for metadata

Important: This implementation focuses on core resumable upload functionality. Security features like authentication, malware scanning, and production-grade monitoring are designed for but not yet implemented.

2. Architecture Overview

System Components

The system follows a client-server architecture with the following implemented components:

- **React Frontend:** Handles file selection, chunking, progress tracking, and resume logic
- **Express API:** Manages upload sessions, chunk processing, and status tracking
- **AWS S3:** Stores individual file chunks in organized key structure
- **DynamoDB:** Tracks upload metadata, chunk completion status, and TTL cleanup
- **IndexedDB:** Client-side file persistence for resume capability

Data Flow

1. User selects file → File stored in IndexedDB → Upload session initiated
2. File split into 5MB chunks → Each chunk uploaded to S3 via API
3. API updates DynamoDB with chunk completion status
4. Frontend tracks progress and handles interruption recovery
5. Upload marked complete when all chunks successfully uploaded

3. IMPLEMENTED FEATURES

The following features are fully functional and tested:

Feature	Status	Description
Chunked Upload	**IMPLEMENTED**	Files split into 5MB chunks for reliable transfer
Progress Tracking	**IMPLEMENTED**	Real-time progress bar with bytes transferred
Pause/Resume	**IMPLEMENTED**	Manual pause/resume controls with state persistence
Page Refresh Recovery	**IMPLEMENTED**	Uploads resume automatically after page reload
Network Interruption Handling	**IMPLEMENTED**	Automatic retry with exponential backoff
System Standby Recovery	**IMPLEMENTED**	Resumes after machine sleep/hibernate
Drag & Drop Interface	**IMPLEMENTED**	Modern file selection with drag-drop support
Multiple File Support	**IMPLEMENTED**	Handle multiple concurrent uploads
Upload Cancellation	**IMPLEMENTED**	Cancel uploads and cleanup resources
Basic File Validation	**PARTIAL**	File size and MIME type checking only

4. NOT IMPLEMENTED / PLANNED FEATURES

The following features are designed for but not currently implemented:

Feature	Status	Impact
JWT Authentication	**NOT IMPLEMENTED**	All uploads are anonymous - security risk
Malware Scanning	**NOT IMPLEMENTED**	Files uploaded directly to S3 without scanning
Rate Limiting	**NOT IMPLEMENTED**	No protection against abuse or DoS attacks
File Type Restrictions	**NOT IMPLEMENTED**	Any file type can be uploaded
User Management	**NOT IMPLEMENTED**	No user accounts or upload ownership
Hash Verification	**NOT IMPLEMENTED**	No integrity checking of uploaded chunks
Production Monitoring	**NOT IMPLEMENTED**	No metrics, alerting, or performance monitoring

5. Technical Implementation Details

Frontend Architecture

The React frontend uses a modular component structure with Zustand for state management:

- `UploadDropzone`: File selection and drag-drop handling
- `UploadList`: Display of active uploads with controls
- `ActiveUpload`: Individual upload progress and status
- `uploadStore`: Zustand store managing upload state and persistence

Backend Architecture

Node.js/Express API with clean separation of concerns:

- `uploadController`: Upload session management (initiate, status, complete)
- `chunkController`: Individual chunk upload processing
- `Upload` model: DynamoDB operations and business logic
- AWS SDK integration for S3 and DynamoDB operations

Chunking Strategy

Files are split into 5MB chunks on the client side using the File API slice() method. Each chunk is uploaded as a separate multipart request with metadata including:

- Upload session ID
- Chunk index and total chunk count
- Chunk size and file metadata

Resume Logic

The system tracks completed chunks in DynamoDB and compares against client state to identify missing chunks. Resume process:

1. Check DynamoDB for upload session status
2. Compare completed chunks with local state
3. Resume upload from first missing chunk
4. Skip already completed chunks

6. Known Limitations

Security Limitations

- **No Authentication:** Anyone can initiate uploads and access upload endpoints
- **No File Scanning:** Malicious files can be uploaded without detection
- **No Rate Limiting:** System vulnerable to abuse and resource exhaustion
- **Basic Validation:** Only MIME type checking, no deep file inspection

Scalability Limitations

- **Single-threaded:** No worker processes for heavy operations
- **Memory Usage:** Large chunks held in memory during processing
- **No CDN:** All traffic goes through single server instance
- **Basic Error Handling:** Limited retry logic and error categorization

Operational Limitations

- **No Monitoring:** No metrics, health checks, or alerting
- **Basic Logging:** Console logging only, no structured logging
- **No Admin Interface:** No way to manage uploads or view system status
- **Manual Deployment:** No CI/CD or automated deployment process

7. Setup and Usage

Prerequisites

- Node.js 16+ and npm
- AWS account with S3 bucket and DynamoDB access
- AWS credentials configured

Backend Setup

1. Clone repository and install dependencies: `npm install`
2. Create `.env` file with AWS credentials
3. Run database setup: `npm run setup-db`
4. Start server: `npm start` (runs on port 5000)

Frontend Setup

1. Navigate to frontend directory: `cd frontend`
2. Install dependencies: `npm install`
3. Start development server: `npm start` (runs on port 3000)

Testing Resumable Uploads

1. Select a file larger than 50MB
2. Start upload and observe progress
3. Test pause/resume functionality
4. Refresh page during upload to test recovery
5. Put machine to sleep and wake to test standby recovery

8. Future Development Roadmap

Phase 1: Security Implementation

- JWT authentication with refresh token rotation
- User registration and login system
- File type validation and restrictions
- Basic rate limiting implementation

Phase 2: Production Readiness

- Malware scanning integration (ClamAV or VirusTotal)
- Structured logging with Winston or similar
- Health check endpoints and monitoring
- Error tracking and alerting

Phase 3: Scalability Improvements

- Worker process architecture for chunk processing
- CDN integration for static assets
- Advanced retry logic and circuit breakers
- Performance optimization and caching

Phase 4: Enterprise Features

- Admin dashboard for upload management
- Audit logging and compliance features
- Advanced user permissions and quotas
- Integration APIs for third-party systems

> **Conclusion:** This implementation successfully demonstrates core resumable upload functionality and meets the technical assessment requirements. However, it should not be deployed to production without implementing the missing security and operational features outlined above.