Secure Data Room Product Requirements Document

1. Abstract

This document outlines the requirements for building a secure virtual data room (VDR) solution for Taghash, a software suite for Venture Capital, Private Equity, Family Offices, and Fund of Funds. The data room will enable secure document sharing, advanced analytics, e-signatures, and white-labeling capabilities. The product is inspired by Papermark's architecture but will be built from scratch using Taghash's preferred technology stack to ensure compatibility with existing products and operational requirements.

The solution will provide enterprise-grade security features including dynamic watermarking, screenshot protection, granular permissions, and comprehensive audit trails. It will support both individual and team-based workflows, with seamless integration capabilities for other Taghash products through a well-designed API architecture.

2. Problem Statement

Current Challenges

1. **Security Concerns**: Traditional file-sharing methods (email, cloud storage) lack adequate security controls for sensitive financial documents, leading to potential data breaches and unauthorized access.
2. **Limited Visibility**: Organizations cannot track who views their documents, for how long, and which sections receive the most attention, making it difficult to gauge investor interest or deal progress.
3. **Inefficient Due Diligence**: M&A transactions and fundraising processes involve sharing hundreds of documents with multiple parties, requiring manual coordination and lacking centralized control.
4. **Compliance Requirements**: Financial institutions need audit trails and access controls to meet regulatory requirements, which generic file-sharing solutions cannot provide.
5. **Brand Consistency**: Organizations cannot maintain their brand identity when sharing documents through third-party platforms.
6. **Integration Gaps**: Existing solutions operate in silos, preventing seamless workflow integration with deal flow management and portfolio monitoring systems.

Target Users

- **Primary Users**: Investment professionals at VC firms, PE firms, Family Offices, and Fund of Funds
- **Secondary Users**: Legal teams, compliance officers, portfolio company executives, and external stakeholders (LPs, potential investors, auditors)

3. Solution

Core Value Proposition

A secure, white-labeled data room solution that provides:

1. **Enterprise Security**: Bank-level security with encryption, dynamic watermarking, and granular access controls
2. **Advanced Analytics**: Page-by-page engagement tracking with real-time insights
3. **Seamless Integration**: Native integration with Taghash's existing product suite
4. **White-Label Capabilities**: Full customization to maintain brand consistency
5. **Collaborative Workflows**: Team-based management with role-based permissions
6. **Compliance Ready**: Comprehensive audit logs and regulatory compliance features

Key Differentiators

1. **Native Integration**: Built specifically for the Taghash ecosystem with shared authentication and data models
2. **AI-Powered Features**: Intelligent document Q&A and automated insights using LLMs
3. **Flexible Deployment**: Support for both SaaS and self-hosted options
4. **Industry-Specific**: Tailored for investment management workflows and terminology

4. User Stories

Epic 1: Account Setup and Team Management

User Story 1.1: Individual Sign Up

- **As a** new user
- **I want to** create an individual account
- **So that** I can start using the data room immediately

Happy Path:

1. User navigates to sign-up page
2. Enters email and password
3. Receives verification email
4. Clicks verification link
5. Completes profile setup
6. Lands on dashboard

Unhappy Path:

1. Email already exists → Show error with login option
2. Invalid email format → Show validation error
3. Weak password → Show password requirements
4. Verification email not received → Provide resend option

User Story 1.2: Team Creation

- **As a** team admin
- **I want to** create a team workspace
- **So that** my colleagues can collaborate on data rooms

Happy Path:

1. Admin creates team from dashboard
2. Names the team and sets description
3. Invites team members via email
4. Members accept invitations
5. Team workspace is active

Unhappy Path:

1. Team name already exists → Suggest alternatives
2. Invalid email addresses → Show validation errors
3. Invitation expires → Allow resend
4. User declines invitation → Notify admin

Epic 2: Data Room Creation and Management

User Story 2.1: Create Data Room

- **As a** team member
- **I want to** create a new data room
- **So that** I can organize documents for a specific deal

Happy Path:

1. Click "Create Data Room" button
2. Enter name and description
3. Configure initial settings
4. Data room created successfully
5. Redirected to data room dashboard

Unhappy Path:

1. Reached data room limit → Show upgrade prompt
2. Invalid characters in name → Show validation error
3. Network error → Show retry option

User Story 2.2: Upload Documents

- **As a** data room owner
- **I want to** upload multiple documents
- **So that** I can share them securely

Happy Path:

1. Navigate to data room
2. Drag and drop files or click upload
3. Files upload with progress indicator
4. Documents processed and indexed
5. Documents appear in folder structure

Unhappy Path:

1. Unsupported file type → Show error with supported types

2. File too large → Show size limit error
3. Upload fails → Allow retry with resume capability
4. Processing fails → Show error with support contact

Epic 3: Access Control and Sharing

User Story 3.1: Create Secure Link

- **As a** data room manager
- **I want to** create a secure sharing link
- **So that** I can control document access

Happy Path:

1. Select documents/folders to share
2. Click "Create Link"
3. Configure permissions (view, download, expiry)
4. Add password protection (optional)
5. Generate and copy link
6. Share via email or messaging

Unhappy Path:

1. No documents selected → Show selection prompt
2. Invalid expiry date → Show date validation
3. Weak password → Show password requirements

User Story 3.2: Email Verification

- **As a** document viewer
- **I want to** verify my email before accessing documents
- **So that** access is properly controlled

Happy Path:

1. Click on shared link
2. Enter email address
3. Receive verification code
4. Enter code
5. Access granted to documents

Unhappy Path:

1. Email not on allow list → Show access denied
2. Invalid verification code → Allow retry
3. Code expired → Provide resend option
4. Email on deny list → Show blocked message

Epic 4: Document Viewing and Analytics

User Story 4.1: View Documents

- **As a** authorized viewer
- **I want to** view documents in the browser
- **So that** I don't need to download files

Happy Path:

1. Access data room via link
2. Navigate folder structure
3. Click on document
4. Document renders in viewer
5. Navigate pages smoothly

Unhappy Path:

1. Document fails to load → Show retry option
2. Browser not supported → Show browser requirements
3. Network timeout → Show offline message

User Story 4.2: Track Engagement

- **As a** data room owner
- **I want to** see detailed analytics
- **So that** I can understand viewer engagement

Happy Path:

1. Navigate to analytics dashboard
2. View summary statistics
3. Drill down to page-level analytics
4. Export analytics report
5. Set up notifications

Unhappy Path:

1. No data available → Show empty state
2. Export fails → Show error with retry
3. Real-time updates lag → Show last updated time

Epic 5: Advanced Features

User Story 5.1: NDA Agreement

- **As a** data room owner
- **I want to** require NDA acceptance
- **So that** viewers acknowledge confidentiality

Happy Path:

1. Enable NDA requirement
2. Upload or create NDA text
3. Viewer sees NDA on first access
4. Viewer accepts NDA

5. Access granted and logged

Unhappy Path:

1. Viewer declines NDA → Deny access and notify owner
2. NDA text too long → Show scroll indicator
3. System error → Allow retry

User Story 5.2: Q&A with AI

- **As a** document viewer
- **I want to** ask questions about documents
- **So that** I can quickly find information

Happy Path:

1. Open Q&A panel
2. Type question in natural language
3. AI processes and responds
4. View source references
5. Rate response quality

Unhappy Path:

1. AI unavailable → Show fallback message
2. Question too vague → Request clarification
3. No relevant content → Suggest rephrase
4. Rate limit exceeded → Show limit message

5. Edge Cases

Authentication and Authorization

1. **Concurrent Sessions**: User logs in from multiple devices → Allow with session management
2. **Password Reset During Active Session**: → Invalidate all sessions, force re-login
3. **Team Member Removed While Viewing**: → Revoke access immediately, show access denied
4. **Invitation to Existing User**: → Add to team without new account creation

Document Management

1. **Duplicate File Names**: → Append timestamp or increment counter
2. **Corrupted File Upload**: → Detect and show specific error message
3. **File Deleted During Processing**: → Handle gracefully, clean up partial data
4. **Version Conflict**: → Show version comparison, allow manual resolution

Viewing and Analytics

1. **Viewer Loses Connection Mid-Session**: → Save progress, allow resume
2. **Multiple Tabs Same Document**: → Track as single session with multiple views
3. **Print Attempt with Protection**: → Block and log attempt
4. **Screenshot with Protection**: → Overlay warning or blank screen

Permissions and Access

1. **Circular Permission Groups**: → Detect and prevent during configuration
2. **Expired Link Access Attempt**: → Show clear expiration message with owner contact
3. **Permission Change During Active View**: → Apply immediately or after session
4. **Bulk Permission Updates**: → Queue and process with progress indicator

System Limits

1. **Storage Quota Exceeded**: → Prevent new uploads, show upgrade options
2. **Concurrent Viewer Limit**: → Queue viewers, show wait time
3. **API Rate Limit**: → Return 429 with retry-after header
4. **Database Connection Pool Exhausted**: → Queue requests, scale automatically

6. Limitations

Technical Limitations

1. **File Size**: Maximum 5GB per file (configurable based on plan)
2. **File Types**: Limited to documented formats (PDF, Office, images, videos)
3. **Concurrent Viewers**: Based on plan (10/50/unlimited)
4. **API Rate Limits**: 1000 requests per minute per API key
5. **Storage**: Based on plan (100GB/1TB/unlimited)

Feature Limitations

1. **Real-time Collaboration**: No simultaneous editing capabilities
2. **Offline Access**: Requires internet connection for security
3. **Mobile Editing**: View-only on mobile devices
4. **Browser Support**: Modern browsers only (Chrome, Firefox, Safari, Edge)
5. **Video Streaming**: Limited to MP4, WebM formats

Security Limitations

1. **Screen Recording**: Cannot prevent at OS level, only detect
2. **Photography**: Physical photography cannot be prevented
3. **Memory Forensics**: Cannot prevent memory dumps
4. **Network Sniffing**: Relies on HTTPS, cannot prevent MITM with compromised certificates

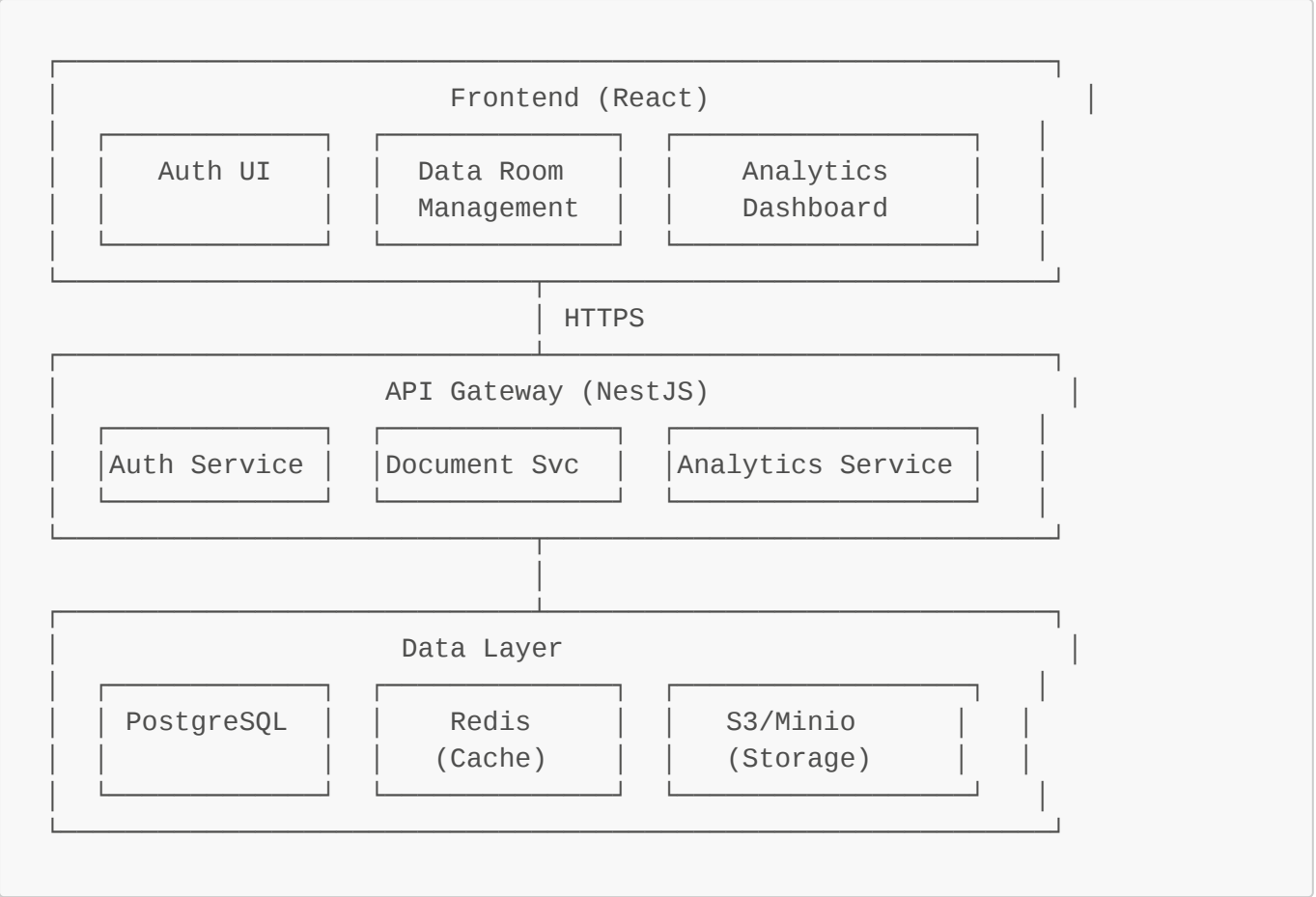
Integration Limitations

1. **Third-party Storage**: Only S3-compatible storage supported
2. **SSO Providers**: Limited to SAML 2.0 and OAuth 2.0
3. **Email Providers**: SMTP or supported transaction email services
4. **LLM Providers**: OpenAI API compatible endpoints only

7. Details of Solution

Architecture Overview

The solution follows a microservices architecture with clear separation of concerns:



Core Components

1. Authentication Service

- **JWT-based authentication** with refresh tokens
- **OAuth2 server** for API integration
- **Multi-factor authentication** support
- **Session management** with Redis
- **Role-based access control** (RBAC)

2. Document Management Service

- **File upload pipeline** with virus scanning
- **Document processing** (PDF generation, thumbnail creation)
- **Version control** system
- **Full-text search** with PostgreSQL
- **Metadata extraction** and indexing

3. Data Room Service

- **Hierarchical folder structure**
- **Bulk operations** support
- **Permission inheritance** system
- **Activity logging** and audit trails

- **Real-time notifications** via WebSocket

4. Analytics Service

- **Event streaming** for real-time analytics
- **Aggregation pipelines** for reports
- **Custom metrics** definition
- **Export capabilities** (CSV, PDF)
- **Scheduled reports** via email

5. Security Service

- **Encryption at rest** using AES-256
- **Encryption in transit** using TLS 1.3
- **Dynamic watermarking** engine
- **DRM protection** for documents
- **Anomaly detection** for suspicious activities

Database Schema (Key Entities)

```
// Team Entity
interface Team {
  id: string;
  name: string;
  plan: PlanType;
  customDomain?: string;
  settings: TeamSettings;
  createdAt: Date;
  updatedAt: Date;
}

// User Entity
interface User {
  id: string;
  email: string;
  name?: string;
  teams: TeamMembership[];
  createdAt: Date;
}

// Data Room Entity
interface DataRoom {
  id: string;
  name: string;
  description?: string;
  teamId: string;
  settings: DataRoomSettings;
  createdAt: Date;
}

// Document Entity
```

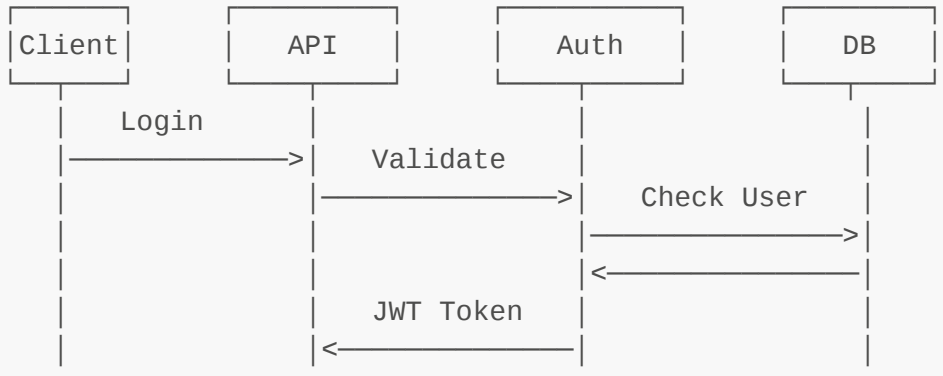
```typescript
interface Document {
  id: string;
  name: string;
  mimeType: string;
  size: bigint;
  storageUrl: string;
  dataRoomId: string;
  folderId?: string;
  metadata: DocumentMetadata;
  createdAt: Date;
}

// Link Entity
interface Link {
  id: string;
  url: string;
  dataRoomId: string;
  permissions: LinkPermissions;
  expiresAt?: Date;
  password?: string;
  createdAt: Date;
}

// View Analytics Entity
interface ViewAnalytics {
  id: string;
  linkId: string;
  viewerId: string;
  documentId: string;
  duration: number;
  pagesViewed: number[];
  downloadedAt?: Date;
  createdAt: Date;
}
```

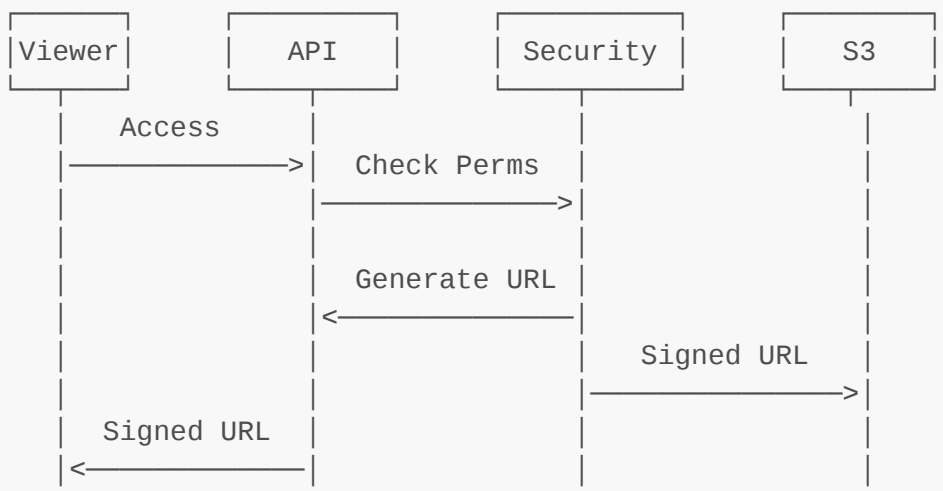
Security Architecture

1. Authentication Flow



```
    |   Token       |                |                |
    |<--------------|                |                |
```

2. Document Access Flow



API Design

The API follows RESTful principles with consistent naming conventions:

Authentication Endpoints

```
POST   /api/auth/register
POST   /api/auth/login
POST   /api/auth/logout
POST   /api/auth/refresh
POST   /api/auth/verify-email
POST   /api/auth/reset-password
```

Team Management Endpoints

```
GET    /api/teams
POST   /api/teams
GET    /api/teams/:teamId
PUT    /api/teams/:teamId
DELETE /api/teams/:teamId
POST   /api/teams/:teamId/invite
POST   /api/teams/:teamId/members
DELETE /api/teams/:teamId/members/:userId
```

Data Room Endpoints

```
GET    /api/teams/:teamId/datarooms
POST   /api/teams/:teamId/datarooms
GET    /api/datarooms/:dataroomId
PUT    /api/datarooms/:dataroomId
DELETE /api/datarooms/:dataroomId
POST   /api/datarooms/:dataroomId/duplicate
```

Document Management Endpoints

```
GET    /api/datarooms/:dataroomId/documents
POST   /api/datarooms/:dataroomId/documents/upload
GET    /api/documents/:documentId
PUT    /api/documents/:documentId
DELETE /api/documents/:documentId
POST   /api/documents/:documentId/move
GET    /api/documents/:documentId/download
```

Link Management Endpoints

```
GET    /api/datarooms/:dataroomId/links
POST   /api/datarooms/:dataroomId/links
GET    /api/links/:linkId
PUT    /api/links/:linkId
DELETE /api/links/:linkId
POST   /api/links/:linkId/verify-access
```

Analytics Endpoints

```
GET    /api/datarooms/:dataroomId/analytics
GET    /api/documents/:documentId/analytics
GET    /api/links/:linkId/analytics
GET    /api/analytics/export
POST   /api/analytics/events
```

Frontend Architecture

The frontend uses React with a component-based architecture:

```
src/
├── components/
│   ├── common/
│   │   ├── Button/
```

```
│   │   ├── Modal/
│   │   └── Table/
│   ├── auth/
│   │   ├── LoginForm/
│   │   └── RegisterForm/
│   ├── dataroom/
│   │   ├── DataRoomList/
│   │   ├── DocumentViewer/
│   │   └── FolderTree/
│   └── analytics/
│       ├── EngagementChart/
│       └── ViewerTable/
├── hooks/
│   ├── useAuth.ts
│   ├── useDataRoom.ts
│   └── useAnalytics.ts
├── services/
│   ├── api.ts
│   ├── auth.service.ts
│   └── dataroom.service.ts
├── store/
│   ├── auth.store.ts
│   └── dataroom.store.ts
└── utils/
    ├── constants.ts
    └── helpers.ts
```

8. Detailed User Acceptance Tests

Epic	User Stories	Acceptance Criteria
Account Setup	Individual Sign Up	• Email validation works correctly • Password meets security requirements (min 8 chars, 1 uppercase, 1 number, 1 special) • Verification email sent within 30 seconds • Verification link expires after 24 hours • Account activated upon verification • Welcome email sent after activation • User redirected to dashboard after first login
	Team Creation	• Team name must be unique per account • Team description limited to 500 characters • Admin can invite up to 50 members at once • Invitation emails sent within 1 minute • Invitations expire after 7 days • Members see team in their dashboard immediately • Admin receives notification when member joins
	Team Member Management	• Admin can change member roles (Admin/Manager/Member) • Admin can remove members with confirmation

Epic	User Stories	Acceptance Criteria
		• Removed members lose access immediately • Member's data remains but is anonymized • Audit log records all changes • Email notifications sent for role changes
Data Room Creation	Create Data Room	• Name field is required (max 100 chars) • Description is optional (max 1000 chars) • URL slug auto-generated from name • Creation limited by plan (1/10/unlimited) • Success message shown with link to data room • Data room appears in list immediately • Default settings applied from team preferences
	Configure Settings	• All settings have sensible defaults • Changes saved automatically with confirmation • Branding preview updates in real-time • Custom domain validates DNS settings • Watermark preview shows on sample document • Settings can be copied from another data room
Document Management	Upload Documents	• Drag-drop works for files and folders • Progress bar shows accurate percentage • Multiple files upload in parallel (max 5) • Failed uploads can be retried • Supported formats clearly indicated • File size limit enforced (5GB default) • Duplicate files detected and warned
	Organize Documents	• Folders can be nested up to 10 levels • Drag-drop to move documents between folders • Bulk select with shift/ctrl click • Right-click context menu for actions • Search finds documents by name/content • Sort by name/date/size/type • Folder permissions inherit from parent
	Process Documents	• PDF generated for all Office formats • Thumbnails created for all pages • Text extracted for search indexing • Processing status shown in real-time • Failed processing can be retried • Original files preserved • Metadata extracted and displayed
Access Control	Create Secure Links	• Unique URL generated each time • Custom slug can be specified • Expiry date picker with presets (1 day/week/month) • Password strength indicator

Epic	User Stories	Acceptance Criteria
		• Email domain restrictions work correctly • Download permission toggles properly • Link preview shows before sharing
	Manage Permissions	• Permission changes apply immediately • Bulk permission updates supported • Permission groups simplify management • Inheritance rules clearly shown • Conflicts resolved with clear precedence • Audit trail for all permission changes • Email notifications for access changes
	Email Verification	• Verification required for first-time viewers • Code sent within 30 seconds • Code expires after 10 minutes • Maximum 3 retry attempts • Clear error messages for failures • Verified emails remembered for 30 days • Allow/deny lists enforced
Document Viewing	View Documents	• Documents load within 3 seconds • Page navigation smooth and responsive • Zoom controls work correctly (25%-400%) • Full-screen mode available • Search within document works • Annotations display correctly • Mobile responsive design
	Download Documents	• Download requires explicit permission • Watermark applied if configured • Download tracked in analytics • Original format preserved • Bulk download as ZIP • Download speed acceptable (>1MB/s) • Resume supported for large files
	Protection Features	• Right-click disabled when configured • Print disabled when configured • Screenshot detection works • Watermark cannot be removed • Copy-paste blocked for text • Developer tools blocked • Mobile screenshot prevented
Analytics	View Analytics	• Real-time updates (within 1 minute) • Time spent accurate to 5 seconds • Page-by-page breakdown available • Geographic data shows country/city

Epic	User Stories	Acceptance Criteria
		• Device/browser detection accurate • Unique vs repeat visitors tracked • Download attempts logged
	Export Analytics	• CSV export includes all data • PDF report professionally formatted • Charts/graphs included in export • Date range filtering works • Scheduled reports delivered on time • Export completes within 2 minutes • Large datasets paginated
	Notifications	• Email alerts sent within 5 minutes • Notification preferences respected • Unsubscribe link works • In-app notifications real-time • Notification history available • Bulk notification management • Do-not-disturb hours respected
Advanced Features	NDA/Agreements	• NDA text clearly displayed • Scroll required to see full text • Accept/Decline buttons prominent • Acceptance logged with timestamp • IP address and details recorded • Declined access blocked completely • NDA PDF available for download
	Q&A with AI	• Natural language queries understood • Responses generated within 5 seconds • Source citations included • Confidence scores shown • Follow-up questions supported • Chat history preserved • Feedback mechanism works
	Custom Branding	• Logo uploads validated (size/format) • Color picker with brand presets • Preview updates immediately • Mobile responsive maintained • Favicon customizable • Email templates branded • White-label domain works
	Bulk Operations	• Select all/none toggles work • Operations show progress bar • Partial failures handled gracefully • Undo available for 5 minutes

Epic	User Stories	Acceptance Criteria
		• Confirmation required for destructive ops • Background processing for large ops • Email notification on completion
Integration	API Access	• API keys generated securely • Rate limits clearly documented • Webhook delivery reliable • OAuth flow standards-compliant • API documentation accurate • Postman collection provided • Sandbox environment available
	Third-party Storage	• S3 credentials validated • Connection test available • Fallback to default storage • Migration tools provided • Bandwidth limits respected • Error handling graceful • Multi-region support
Performance	Load Times	• Dashboard loads < 2 seconds • Document list loads < 1 second • Search results < 500ms • Analytics refresh < 3 seconds • File upload starts immediately • No blocking operations • Lazy loading implemented
	Scalability	• Supports 1000+ concurrent viewers • 10,000+ documents per data room • 1M+ analytics events/day • Auto-scaling configured • Database queries optimized • CDN utilized effectively • Cache hit rate > 80%
Security	Authentication	• Passwords hashed with bcrypt • Sessions expire after inactivity • MFA enrollment smooth • Password reset secure • Brute force protection active • Account lockout after failures • Security headers configured
	Data Protection	• All data encrypted at rest • TLS 1.3 for all connections • PII data anonymized • GDPR compliance tools

Epic	User Stories	Acceptance Criteria
		• Data retention policies enforced • Backup encryption verified • Audit logs tamper-proof

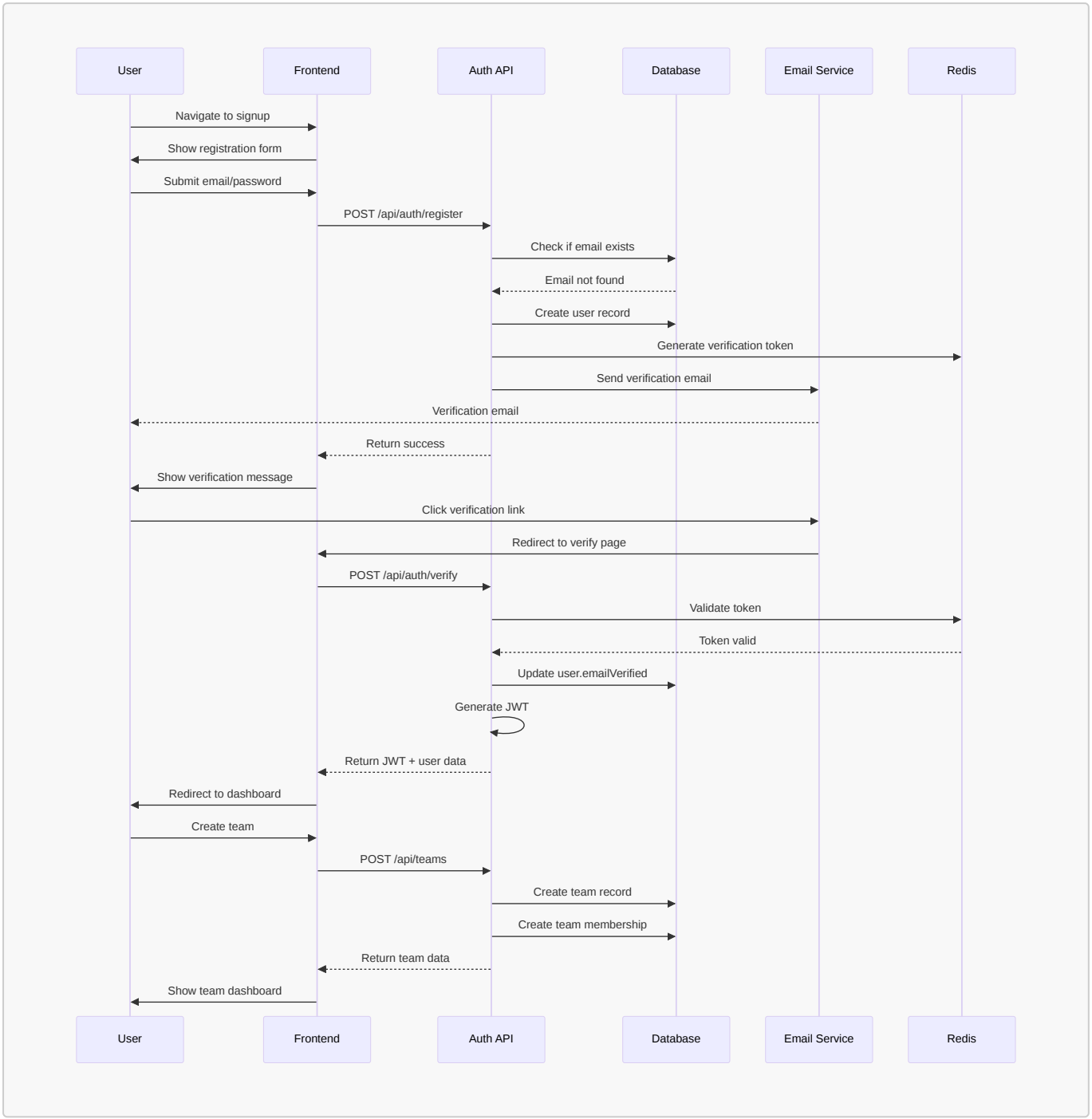
9. Technical Implementation

Phase 1: Foundation and Authentication

Objective

Establish the core infrastructure, authentication system, and basic team management functionality.

Sequence Diagram: User Registration and Team Creation



Tasks

1. **Backend Setup**
    - Initialize NestJS project with TypeScript configuration
    - Set up TypeORM with PostgreSQL connection
    - Configure Redis for session management
    - Set up Bull MQ for job queues
    - Implement environment configuration with dotenv
    - Set up logging with Winston
    - Configure CORS and security middleware

    Technologies:

    - Backend: `@nestjs/core`, `@nestjs/common`, `@nestjs/platform-express`, `typescript`, `@nestjs/typeorm`, `typeorm`, `pg`, `redis`, `bullmq`, `dotenv`, `winston`, `helmet`, `cors`
    - Ops: `@nestjs/cli`, `ts-node`, `nodemon`

2. **Database Schema**
    - Create migration for users table
    - Create migration for teams table
    - Create migration for team_members table
    - Create migration for sessions table
    - Set up database indexes for performance
    - Implement soft deletes for data retention

    Technologies:

    - Backend: `typeorm`, `pg`, `knex` (for migrations)
    - Ops: `typeorm-cli`, `knex-migrate`

3. **Authentication Module**
    - Implement JWT strategy with Passport.js
    - Create registration endpoint with validation
    - Implement email verification flow
    - Create login endpoint with rate limiting
    - Implement password reset functionality
    - Add refresh token mechanism
    - Set up OAuth2 server for API access

    Technologies:

    - Backend: `@nestjs/passport`, `passport`, `passport-jwt`, `@nestjs/jwt`, `bcrypt`, `class-validator`, `class-transformer`, `@node-oauth/express-oauth-server`, `express-rate-limit`
    - Frontend: `axios`

4. **Team Management Module**

- Create team CRUD endpoints
- Implement invitation system
- Add role-based permissions (Admin/Manager/Member)
- Create team switching functionality
- Implement team settings management

Technologies:

- Backend: `accesscontrol`, `uuid`, `nanoid`
- Frontend: `@tanstack/react-query`, `@reduxjs/toolkit`, `react-redux`

5. **Email Service**

- Configure Nodemailer with Mailgun
- Create email templates (verification, welcome, invitation)
- Implement email queue with Bull MQ
- Add email tracking for analytics

Technologies:

- Backend: `nodemailer`, `nodemailer-mailgun-transport`, `mjml`, `bullmq`

6. **Frontend Foundation**

- Set up React with Vite
- Configure Tailwind CSS with custom theme
- Implement routing with React Router
- Set up state management with Redux
- Create authentication context and hooks
- Implement protected routes
- Build authentication UI components

Technologies:

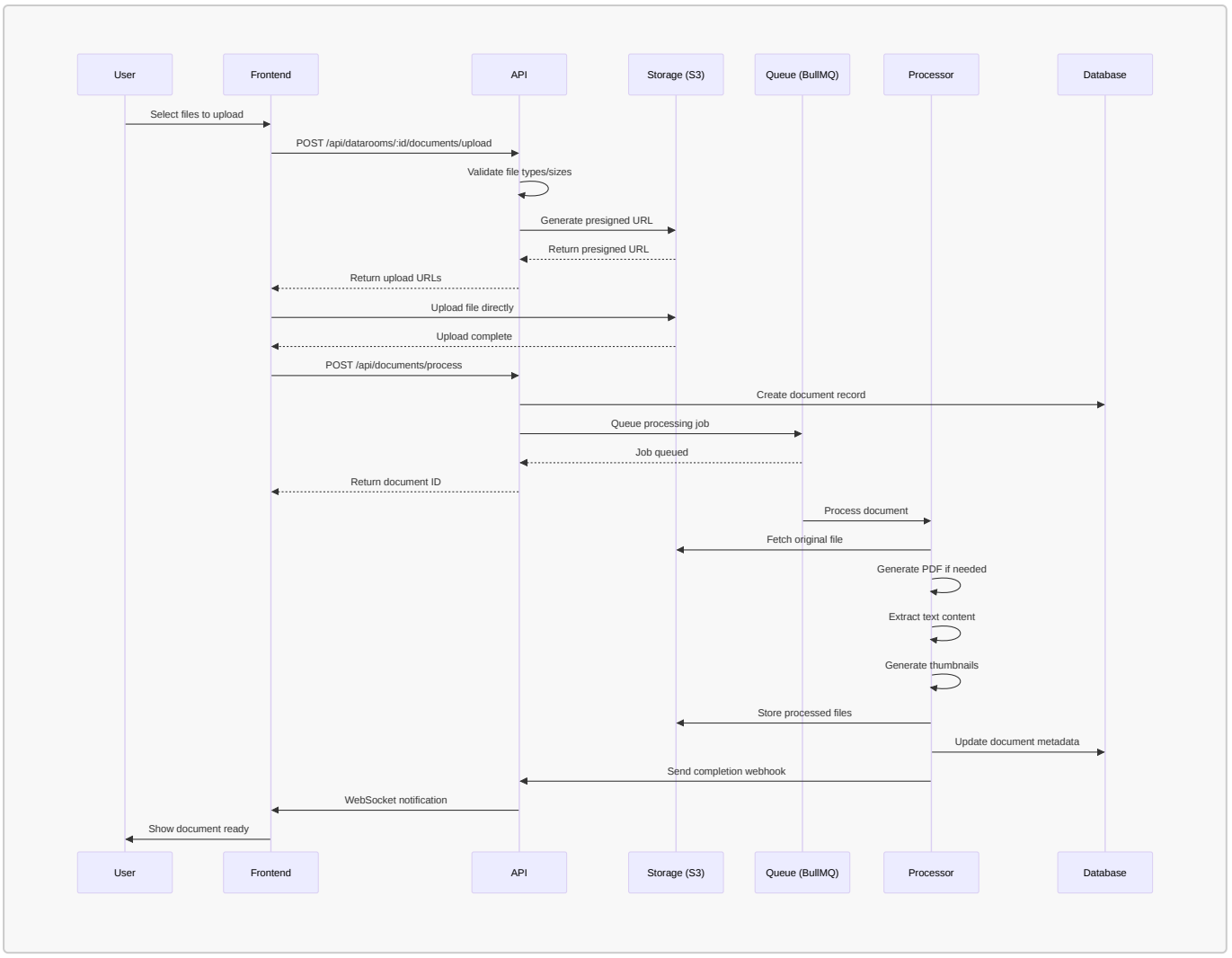
- Frontend: `react`, `react-dom`, `vite`, `@vitejs/plugin-react`, `tailwindcss`, `@tailwindcss/forms`, `react-router-dom`, `@reduxjs/toolkit`, `react-redux`, `shadcn/ui`, `lucide-react`, `react-hook-form`, `@hookform/resolvers`, `zod`
- Ops: `postcss`, `autoprefixer`, `prettier`, `eslint`, `@typescript-eslint/parser`

Phase 2: Data Room Core Functionality

Objective

Implement data room creation, document upload, and basic folder management.

Sequence Diagram: Document Upload and Processing



Tasks

1. **Data Room Module**
   - Create data room entity and migrations
   - Implement CRUD endpoints for data rooms
   - Add data room settings management
   - Create data room duplication functionality
   - Implement data room archiving

   Technologies:

   - Backend: `typeorm`, `class-validator`, `lodash`
   - Frontend: `@tanstack/react-table`, `date-fns`

2. **Document Upload Service**
   - Set up S3/Minio integration
   - Implement multipart upload for large files
   - Create presigned URL generation
   - Add file type validation
   - Implement virus scanning integration (using ClamAV)

   Technologies:

- Backend: `@aws-sdk/client-s3`, `@aws-sdk/s3-request-presigner`, `minio`, `multer`, `multer-s3`, `file-type`, `clamscan`
- Frontend: `react-dropzone`, `axios`
- Ops: `docker` (for ClamAV container)

3. **Document Processing Pipeline**

   - Set up document processing queue
   - Implement PDF conversion for Office files
   - Add text extraction for search (and index using OpenSearch)
   - Create thumbnail generation
   - Implement metadata extraction

   Technologies:

   - Backend: `bullmq`, `libreoffice-convert`, `pdf-parse`, `sharp`, `@opensearch-project/opensearch`, `exiftool-vendored`
   - Ops: `libreoffice` (system dependency), `opensearch` (docker container)

4. **Folder Management**

   - Create folder entity and migrations
   - Implement nested folder structure
   - Add drag-and-drop functionality
   - Create folder permissions system
   - Implement bulk move operations

   Technologies:

   - Backend: `typeorm`, `path`
   - Frontend: `@hello-pangea/dnd`, `react-arborist`, `immer`

5. **Frontend Data Room UI**

   - Build data room dashboard
   - Create file upload component with progress
   - Implement folder tree navigation
   - Add document list with sorting/filtering
   - Create document preview modal

   Technologies:

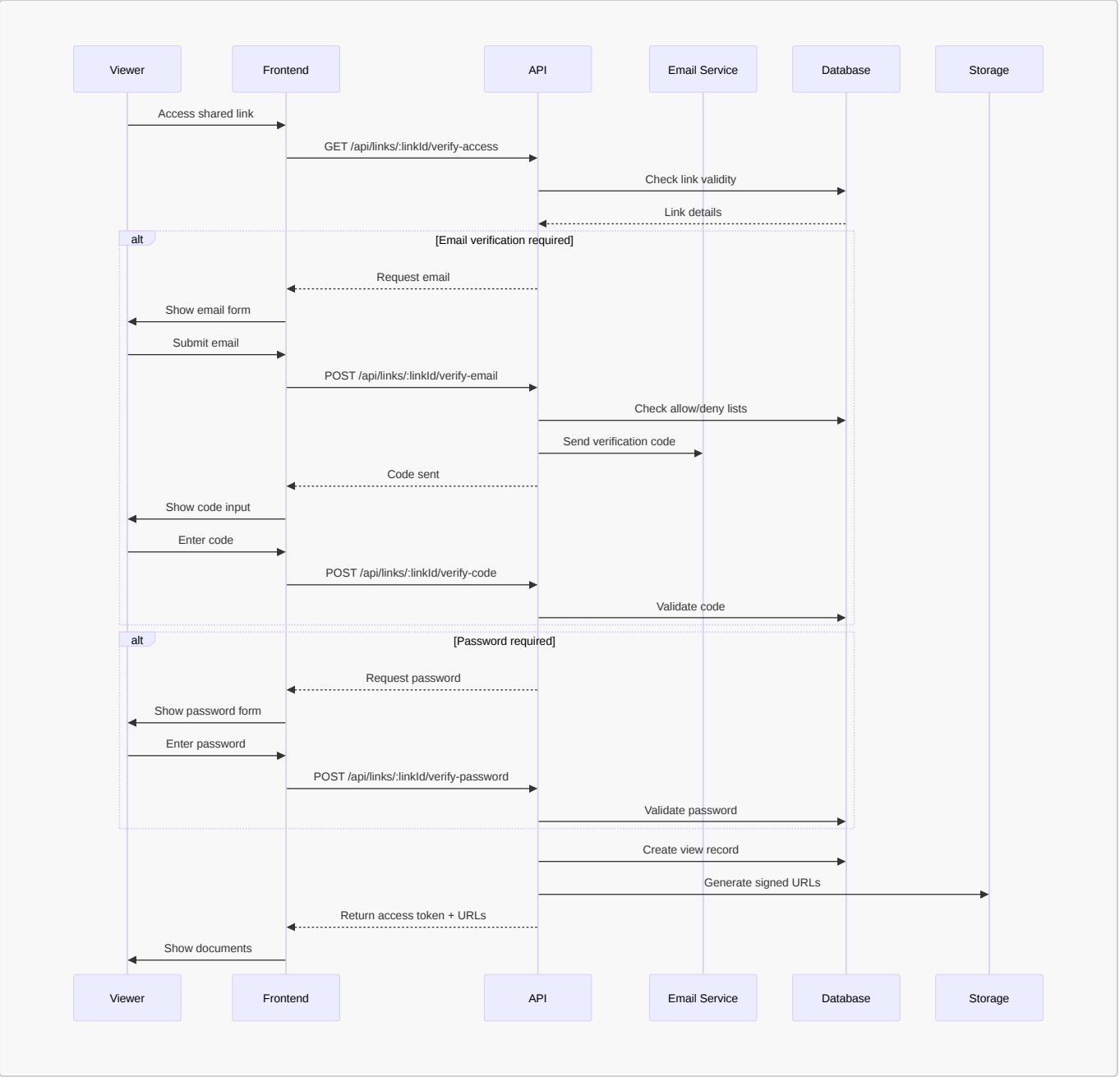
   - Frontend: `@tanstack/react-table`, `react-dropzone`, `tus-js-client`, `@radix-ui/react-dialog`, `@radix-ui/react-progress`, `react-intersection-observer`

Phase 3: Access Control and Sharing

Objective

Implement secure link generation, permission management, and viewer authentication.

Sequence Diagram: Secure Link Access



Tasks

1. **Link Management System**

   - Create link entity and migrations
   - Implement secure link generation
   - Add link customization options
   - Create link expiration handling (recurring job every 15 mins)
   - Implement link analytics tracking

   Technologies:

   - Backend: `nanoid`, `crypto`, `moment`, `bullmq` (for recurring expiration check job)
   - Frontend: `react-copy-to-clipboard`, `qrcode.react`

Note: Link expiration will be handled by a recurring BullMQ job that runs every 15 minutes to check and expire links, rather than creating individual delayed jobs for each link to avoid Redis memory issues.

2. **Permission System**
   - Design permission model
   - Implement permission inheritance
   - Create permission groups
   - Add bulk permission management
   - Implement permission caching

   Technologies:

   - Backend: `accesscontrol`, `redis`, `ioredis`
   - Frontend: `react` (using context for permission checks)

3. **Viewer Authentication**
   - Create email verification flow
   - Implement OTP generation and validation
   - Add password protection
   - Create viewer session management
   - Implement remember device feature

   Technologies:

   - Backend: `otpauth`, `express-session`, `connect-redis`, `ua-parser-js`
   - Frontend: `react-otp-input`, `js-cookie`

4. **Access Control Lists**
   - Implement email allow/deny lists
   - Add domain-based restrictions
   - Create IP-based access control
   - Implement geographic restrictions
   - Add time-based access windows

   Technologies:

   - Backend: `ipaddr.js`, `geoip-lite`, `validator`
   - Frontend: `react-select`, `react-datetime-picker`

   Note: Time-based access windows will be enforced at request time by checking the current time against the configured access windows, rather than using scheduled jobs.

5. **Frontend Sharing UI**
   - Build link creation wizard
   - Create permission configuration UI
   - Implement share modal with options
   - Add link management dashboard

- Create viewer authentication flow

Technologies:

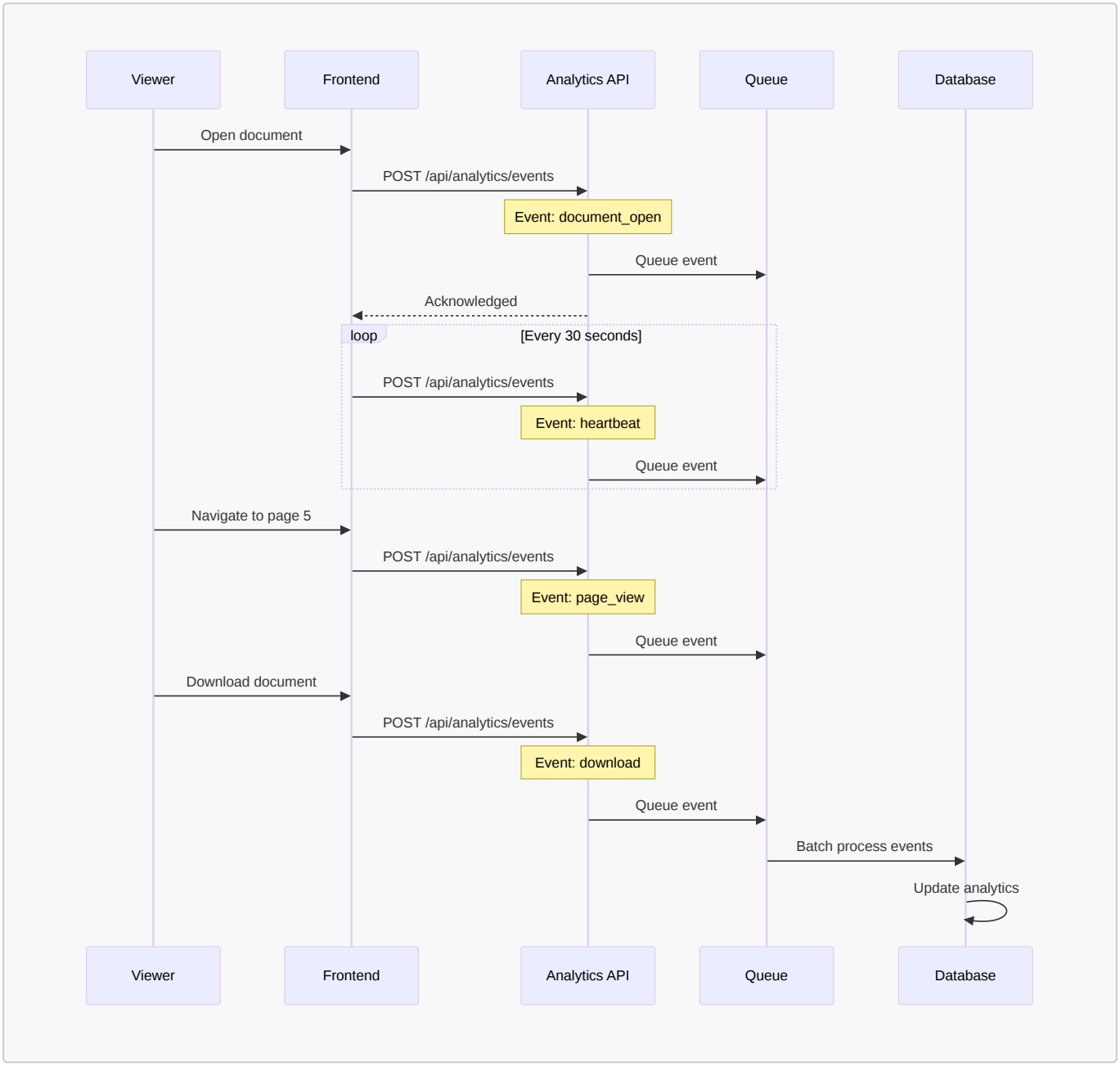
- Frontend: `react-hook-form`, `@radix-ui/react-dialog`, `@radix-ui/react-tabs`, `react-select`, `react-datepicker`

Phase 4: Document Viewing and Analytics

Objective

Build secure document viewer with tracking and comprehensive analytics dashboard.

Sequence Diagram: Document View Tracking



Tasks

1. **Document Viewer**

- Implement PDF.js integration
- Create image viewer component
- Add video player with tracking
- Implement Excel/Word preview
- Add text selection controls

Technologies:

- Frontend: `react-pdf`, `pdfjs-dist`, `react-image-gallery`, `video.js`, `react-player`, `@cyntler/react-doc-viewer`, `react-spreadsheet`
- Backend: `pdf-lib`

2. **View Protection**

- Implement watermark overlay
- Add print protection
- Create screenshot detection
- Implement right-click blocking
- Add keyboard shortcut blocking

Technologies:

- Frontend: `react-watermark`, `disable-devtool`, `react-hotkeys-hook`
- Backend: `pdf-lib`, `jimp`

3. **Analytics Tracking**

- Create event tracking system
- Implement page-level analytics
- Add time tracking algorithm
- Create download tracking
- Implement engagement scoring

Technologies:

- Backend: `bullmq`, `typeorm` (for storing analytics in PostgreSQL), `date-fns`
- Frontend: Custom event tracking implementation using `axios`

4. **Analytics Dashboard**

- Build analytics overview page
- Create detailed view reports
- Implement real-time updates
- Add export functionality
- Create custom date ranges

Technologies:

- Frontend: `echarts`, `echarts-for-react`, `react-datepicker`
- Backend: `exceljs`, `pdfkit`, `socket.io`, `typeorm` (for analytics queries)

5. **Frontend Viewer UI**

- Build responsive document viewer
- Create navigation controls
- Implement zoom functionality
- Add fullscreen mode
- Create mobile-optimized viewer

Technologies:

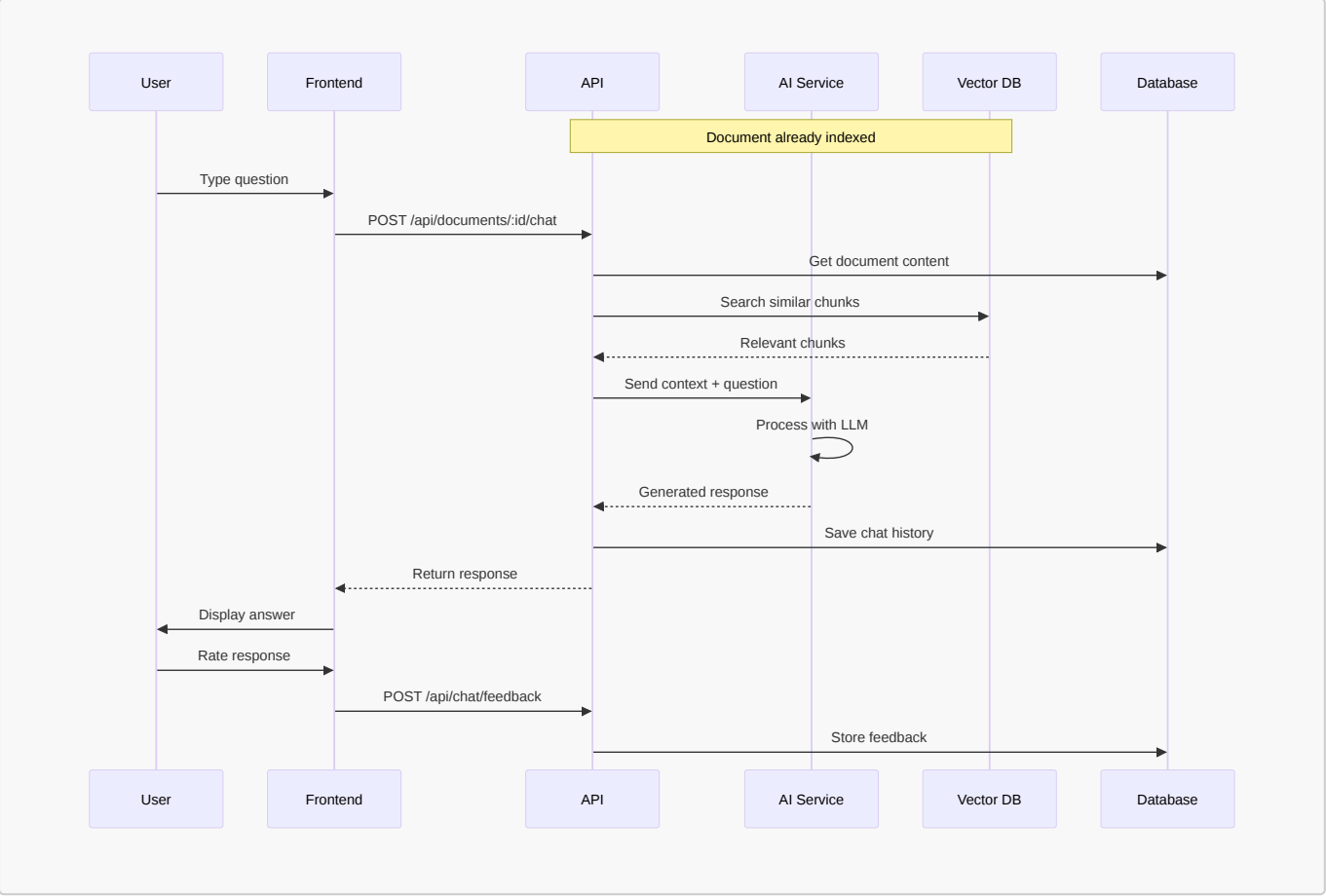
- Frontend: `react-zoom-pan-pinch`, `react-fullscreen`, `react-swipeable`, `@radix-ui/react-toolbar`

Phase 5: Advanced Features

Objective

Implement AI-powered Q&A, NDA management, and advanced security features.

Sequence Diagram: AI-Powered Q&A



Tasks

1. **AI Integration**
   - Set up OpenAI API integration
   - Implement document chunking
   - Create vector embeddings through background jobs
   - Set up OpenSearch with vector search capabilities

- Implement hybrid retrieval (BM25 + vector + filters)

Technologies:

- Backend: `openai`, `langchain`, `@opensearch-project/opensearch` (with k-NN plugin), `tiktoken`, `pdf-parse`, `bullmq` (for async embedding generation)
- Frontend: `@copilotkit/react-core`, `@copilotkit/react-ui`, `@copilotkit/react-textarea`
- Ops: `opensearch` (with k-NN plugin enabled)

2. **Q&A System**

- Create chat interface using CopilotKit
- Implement conversation history
- Add source attribution
- Create feedback system
- Implement rate limiting

Technologies:

- Frontend: `@copilotkit/react-core`, `@copilotkit/react-ui`, `react-markdown`, `remark-gfm`
- Backend: `express-rate-limit`, `rate-limiter-flexible`

3. **NDA Management**

- Create agreement entity
- Build NDA creation interface
- Implement acceptance tracking
- Add signature integration
- Create agreement templates

Technologies:

- Backend: `pdfkit`, `docxtemplater`
- Frontend: `react-signature-canvas`, `react-pdf-viewer`, `@radix-ui/react-checkbox`

4. **Advanced Security**

- Implement dynamic watermarking
- Add forensic watermarking
- Create activity anomaly detection
- Implement session recording
- Add security audit logs

Technologies:

- Backend: `pdf-lib`, `jimp`, `sharp`, `node-forge`
- Frontend: `rrweb`, `@fingerprintjs/fingerprintjs`

5. **Custom Branding**

- Create branding settings
- Implement theme customization
- Add white-label domains
- Create custom email templates
- Implement favicon customization

Technologies:

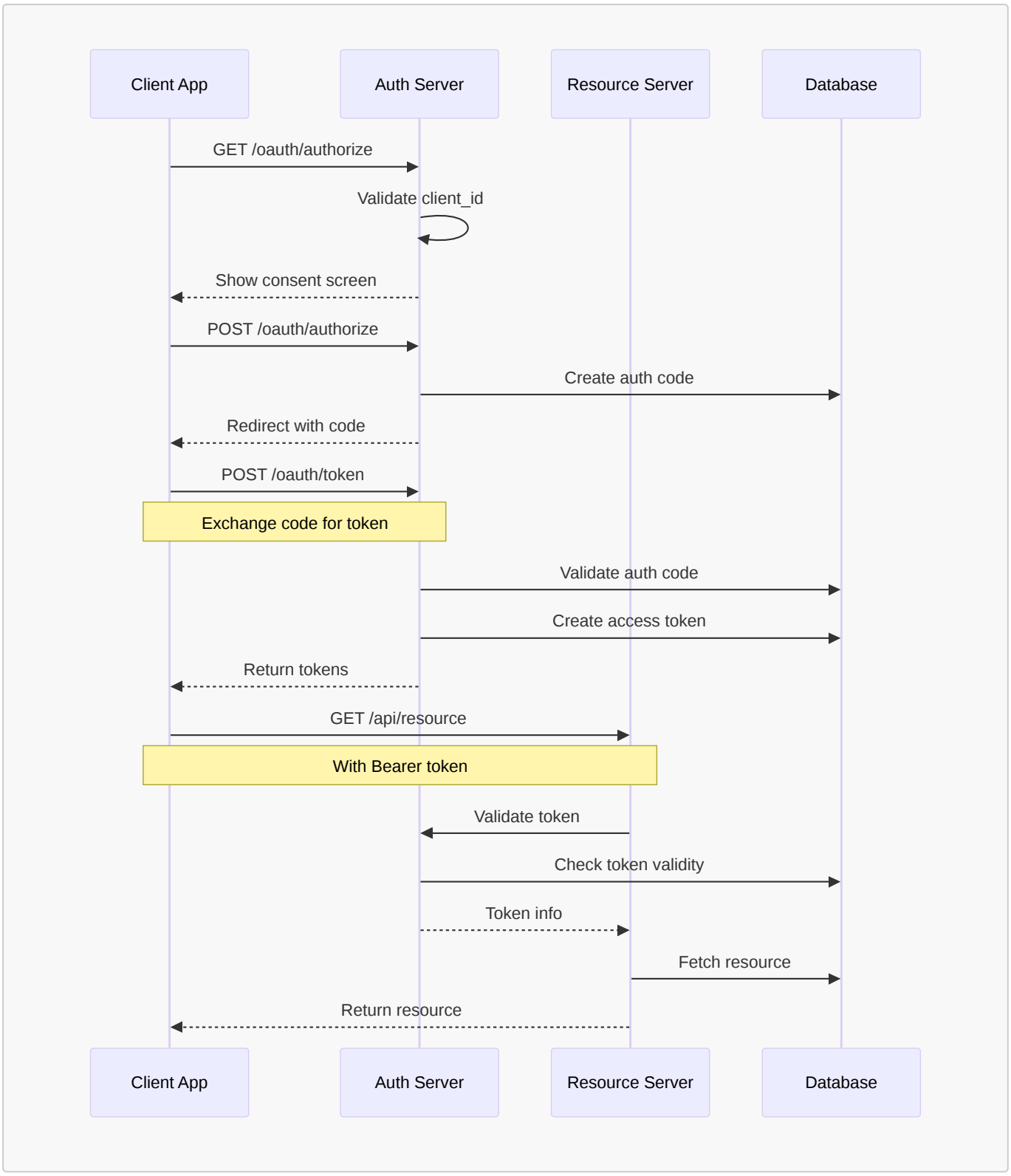
- Backend: `color`, `css-tree`, `mjml`
- Frontend: `react-color`, `styled-components`, `polished`

Phase 6: Integration and API

Objective

Build OAuth2 server, REST API, webhooks, and integration with Taghash ecosystem.

Sequence Diagram: OAuth2 Flow



Tasks

1. **OAuth2 Server**
   - Implement authorization code flow
   - Create client registration system
   - Add scope management
   - Implement token introspection
   - Create consent management

Technologies:

- Backend: `@node-oauth/express-oauth-server`, `jsonwebtoken`
- Frontend: `react-oauth-flow`

2. **REST API**

- Design API versioning strategy
- Implement rate limiting
- Add request validation
- Create API documentation
- Implement pagination

Technologies:

- Backend: `@nestjs/swagger`, `swagger-ui-express`, `express-rate-limit`, `class-validator`, `nestjs-paginate`
- Frontend: `swagger-ui-react`

3. **Webhook System**

- Create webhook registration
- Implement event delivery
- Add retry mechanism
- Create signature verification
- Implement webhook logs

Technologies:

- Backend: `bullmq`, `axios-retry`, `crypto`, `express-async-handler`
- Frontend: `react-hook-form`, `@monaco-editor/react` (for webhook payload preview)

4. **Integration Layer**

- Create TypeScript/JavaScript SDK for Data Room API
- Implement SDK authentication methods
- Add comprehensive type definitions
- Create reusable React components library
- Publish to private npm registry (GitLab)

Technologies:

- SDK: `typescript`, `axios`, `tsup` (for bundling), `typedoc` (for documentation)
- Frontend Components: `storybook`, `@storybook/react`, `rollup`, `@rollup/plugin-typescript`
- Publishing: `npm`, `semantic-release`, `@semantic-release/gitlab`

5. **Developer Portal**

- Build API documentation site
- Create interactive API explorer
- Add code examples

- Implement SDK generation
- Create developer onboarding

Technologies:

- Frontend: `docusaurus`, `redoc`, `@stoplight/elements`, `prism-react-renderer`
- Backend: `openapi-generator`, `@nestjs/swagger`

Phase 7: Performance and Deployment

Objective

Optimize performance, implement monitoring, and prepare for production deployment.

Tasks

1. **Performance Optimization**
   - Implement database query optimization
   - Add Redis caching layer
   - Create CDN integration
   - Implement lazy loading
   - Add connection pooling

   Technologies:

   - Backend: `ioredis`, `dataloader`, `@nestjs/cache-manager`
   - Frontend: `react-lazy-load-image-component`, `react-intersection-observer`
   - Ops: `aws-cloudfront`

2. **Monitoring and Logging**
   - Set up application monitoring
   - Implement distributed tracing
   - Create custom metrics
   - Add error tracking
   - Implement log aggregation

   Technologies:

   - Backend: `prom-client`
   - Frontend: `@sentry/react`, `web-vitals`
   - Ops: `prometheus`, `grafana`

3. **Testing Suite**
   - Write unit tests (80% coverage)
   - Create integration tests
   - Implement E2E tests with Playwright
   - Add performance tests
   - Create security tests

Technologies:

- Backend: `jest`, `@nestjs/testing`, `supertest`, `mocha`, `chai`, `sinon`
- Frontend: `@testing-library/react`, `@testing-library/jest-dom`, `vitest`, `playwright`
- Ops: `k6`, `artillery`, `owasp-zap`

4. **DevOps Setup**

- Create Docker containers
- Write docker-compose files
- Implement CI/CD pipelines
- Create Kubernetes manifests
- Add infrastructure as code

Technologies:

- Containerization: `docker`, `docker-compose`, `dockerfile-lint`
- CI/CD: `gitlab-ci`, `semantic-release`
- IaC: `terraform`, `ansible`

5. **Production Readiness**

- Implement health checks
- Create backup strategies
- Add disaster recovery
- Implement auto-scaling
- Create deployment runbooks

Technologies:

- Backend: `@nestjs/terminus`, `pg-backup-restore`, `bullmq` (for scheduled backups)
- Ops: `restic`, `runbook-template`
- Monitoring: `uptime-kuma`, `statuspage`

10. Conclusion

This Product Requirements Document outlines a comprehensive plan for building a secure data room solution tailored for the investment management industry. The solution addresses critical challenges in document sharing, due diligence processes, and compliance requirements while providing advanced features like AI-powered Q&A and detailed analytics.

The phased implementation approach ensures that core functionality is delivered early while allowing for iterative improvements and feature additions. By leveraging modern technologies and following security best practices, the data room will provide enterprise-grade security while maintaining ease of use.

Key success factors include:

1. **Security First**: Every feature is designed with security as the primary consideration
2. **User Experience**: Intuitive interfaces that don't sacrifice functionality for simplicity
3. **Scalability**: Architecture that can grow with customer needs

4. **Integration**: Seamless connection with existing Taghash products
5. **Compliance**: Built-in features to support regulatory requirements

The technical implementation guide provides a clear roadmap for development teams, with detailed sequence diagrams and task breakdowns for each phase. This structured approach minimizes risk and ensures consistent progress toward the final product.

By following this PRD, Taghash will deliver a data room solution that not only matches existing market offerings but exceeds them through deep integration with the investment management workflow and innovative features powered by modern AI capabilities.