DocuSign Clone - Product Requirements Document

Version: 1.0
Date: June 23, 2025
Project Duration: 14 Days
Development Type: Full-Stack MERN Application

---

1. Overview

The Document Signature App is a secure, web-based digital signature platform that enables users to upload, share, and collect legally binding electronic signatures on PDF documents. Built as a lightweight alternative to DocuSign, the application streamlines document workflows by allowing document owners to send signing requests via tokenized links and track signature status in real-time.

Primary Use Cases:

- Contract signing for small businesses and freelancers
- Legal document execution requiring multiple parties
- HR onboarding document collection
- Vendor agreement processing
- Personal document signing (rental agreements, consent forms, etc.)

The application eliminates the need for physical document printing, scanning, and mailing while maintaining document integrity and providing comprehensive audit trails for compliance purposes.

---

2. Objectives

2.1 Primary Goals

- **Usability**: Deliver an intuitive, responsive interface that requires minimal learning curve for both document owners and signers
- **Security**: Implement enterprise-grade security measures including JWT authentication, password hashing, and secure tokenized access
- **Document Authenticity**: Ensure signed documents maintain legal validity through embedded signatures and comprehensive audit trails
- **Accessibility**: Provide public signing capabilities without requiring signer account creation

2.2 Success Metrics

- Document upload and signature completion rate > 95%
- Average time to complete signature workflow < 3 minutes
- Zero security breaches during development and initial deployment
- Cross-browser compatibility (Chrome, Firefox, Safari, Edge)
- Mobile-responsive design supporting tablets and smartphones

2.3 Non-Goals (Out of Scope)

- Multi-format document support (Word, Excel) - PDF only
- Advanced PDF editing features beyond signature placement
- Enterprise SSO integration
- Payment processing capabilities
- Mobile native applications

---

3. Key Features

3.1 User Authentication System

- **JWT-based Authentication**: Secure token-based authentication with configurable expiration
- **User Registration**: Email-based account creation with password strength validation
- **Login/Logout**: Secure session management with refresh token support
- **Password Reset**: Email-based password recovery workflow

3.2 Document Management

- **PDF Upload**: Support for PDF files up to 10MB with file validation
- **Document Viewing**: High-quality PDF rendering with zoom and navigation controls
- **Document Listing**: Paginated dashboard showing user's uploaded documents
- **Document Sharing**: Generate secure, tokenized links for external sharing

3.3 Digital Signature Workflow

- **Signature Field Placement**: Drag-and-drop interface for positioning signature areas
- **Multiple Signature Types**: Support for drawn signatures, typed signatures, and uploaded signature images
- **Signature Status Tracking**: Real-time status updates (Pending, Signed, Rejected, Expired)
- **Multi-party Signing**: Support for documents requiring multiple signatures in sequence

3.4 Public Signing Interface

- **Tokenized Access**: Secure, time-limited access to signing interface without authentication
- **Signer Information Collection**: Capture signer name, email, and optional phone number
- **Signature Placement**: Intuitive interface for signers to place signatures in designated areas
- **Completion Confirmation**: Email notifications to both signer and document owner upon completion

3.5 Audit Trail System

- **Comprehensive Logging**: Track all document interactions with timestamps and IP addresses
- **Signature Metadata**: Record signature method, device information, and geolocation (if permitted)
- **Document History**: Maintain complete audit trail from upload to final signature
- **Tamper Evidence**: Detect and log any attempts to modify signed documents

3.6 Final Document Generation

- **Embedded Signatures**: Permanently embed signatures into PDF structure
- **Audit Certificate**: Generate tamper-evident audit certificate as separate PDF
- **Document Sealing**: Apply cryptographic sealing to prevent post-signature modifications
- **Download Options**: Provide original and signed versions with audit reports

---

4. Technical Requirements

4.1 Frontend Technology Stack

- **Framework**: React 18+ with functional components and hooks
- **Styling**: Tailwind CSS 3+ for responsive design and component styling
- **PDF Rendering**: React-PDF library for document preview and annotation
- **State Management**: React Context API or Redux Toolkit for global state
- **HTTP Client**: Axios for API communication with interceptors for auth
- **UI Components**: Custom components built with Tailwind, no external UI library dependency

4.2 Backend Technology Stack

- **Runtime**: Node.js 18+ with Express.js framework
- **Database**: MongoDB with Mongoose ODM (Alternative: PostgreSQL with Sequelize/Prisma)
- **Authentication**: JWT with bcrypt for password hashing
- **File Handling**: Multer for multipart file uploads with size limitations

- **PDF Processing**: PDF-Lib for signature embedding and document manipulation
- **Email Service**: Nodemailer with SMTP configuration for notifications

4.3 Database Schema Requirements

Users Collection/Table:

```
- _id/id (ObjectId/UUID)
- email (String, unique, indexed)
- passwordHash (String)
- firstName (String)
- lastName (String)
- createdAt (DateTime)
- updatedAt (DateTime)
```

Documents Collection/Table:

```
- _id/id (ObjectId/UUID)
- userId (ObjectId/UUID, foreign key)
- fileName (String)
- originalName (String)
- fileSize (Number)
- filePath (String)
- mimeType (String)
- status (Enum: draft, pending, completed, rejected)
- createdAt (DateTime)
- updatedAt (DateTime)
```

Signatures Collection/Table:

```
- _id/id (ObjectId/UUID)
- documentId (ObjectId/UUID, foreign key)
- signerEmail (String)
- signerName (String)
- signatureData (String, base64 encoded)
- signatureType (Enum: drawn, typed, uploaded)
- position (Object: {x, y, width, height, page})
- status (Enum: pending, signed, rejected, expired)
- tokenHash (String, indexed)
- signedAt (DateTime)
- ipAddress (String)
- userAgent (String)
- createdAt (DateTime)
```

AuditLogs Collection/Table:

```
- _id/id (ObjectId/UUID)
- documentId (ObjectId/UUID, foreign key)
- action (String)
- userId (ObjectId/UUID, nullable)
- ipAddress (String)
- userAgent (String)
- metadata (Object)
- timestamp (DateTime)
```

4.4 Infrastructure Requirements

- **Deployment**: Docker containers with Docker Compose for local development
- **Production**: Cloud deployment (Vercel/Netlify for frontend, Railway/Render for backend)
- **File Storage**: Local filesystem for development, cloud storage (AWS S3/Cloudinary) for production
- **Environment Management**: Dotenv for configuration management
- **Security**: CORS configuration, rate limiting, input validation

---

5. API Endpoints

5.1 Authentication Endpoints

```
POST /api/auth/register
Body: { email, password, firstName, lastName }
Response: { user, token, refreshToken }

POST /api/auth/login
Body: { email, password }
Response: { user, token, refreshToken }

POST /api/auth/refresh
Body: { refreshToken }
Response: { token }

POST /api/auth/logout
Headers: Authorization: Bearer {token}
Response: { message }

POST /api/auth/forgot-password
Body: { email }
Response: { message }

POST /api/auth/reset-password
Body: { token, newPassword }
Response: { message }
```

5.2 Document Management Endpoints

```
POST /api/docs/upload
Headers: Authorization: Bearer {token}
Body: FormData with PDF file
Response: { document }

GET /api/docs
Headers: Authorization: Bearer {token}
Query: ?page=1&limit=10&status=all
Response: { documents, pagination }

GET /api/docs/:id
Headers: Authorization: Bearer {token}
Response: { document, signatures }

DELETE /api/docs/:id
Headers: Authorization: Bearer {token}
Response: { message }

GET /api/docs/:id/download
Headers: Authorization: Bearer {token}
Response: PDF file stream
```

5.3 Signature Management Endpoints

```
POST /api/signatures
Headers: Authorization: Bearer {token}
Body: { documentId, signerEmail, signerName, position }
Response: { signature, signingLink }

GET /api/signatures/:id
Query: ?token={signingToken}
Response: { signature, document }

PUT /api/signatures/:id
Body: { signatureData, signerName, signerEmail }
Query: ?token={signingToken}
Response: { signature }

POST /api/signatures/:id/reject
Query: ?token={signingToken}
Body: { reason }
Response: { message }

POST /api/signatures/finalize/:documentId
Headers: Authorization: Bearer {token}
Response: { finalDocument, auditReport }
```

5.4 Audit Trail Endpoints

```
GET /api/audit/:docId
Headers: Authorization: Bearer {token}
Response: { auditLogs }

GET /api/audit/:docId/report
Headers: Authorization: Bearer {token}
Response: PDF audit report file
```

---

6. User Roles and Permissions

6.1 Document Owner (Uploader)

Capabilities:

- Upload and manage PDF documents
- Create signature requests with signer details

- Configure signature field positions and requirements
- Monitor signature status and progress
- Download completed documents with embedded signatures
- Access complete audit trails
- Delete documents and revoke pending signatures

Restrictions:

- Cannot sign their own documents (conflict of interest prevention)
- Limited to 50MB total storage per account (configurable)
- Maximum 10 active signature requests per document

6.2 Document Signer (Invited User)

Capabilities:

- Access documents via secure tokenized links
- View document content and signature requirements
- Place digital signatures in designated areas
- Reject signature requests with reason
- Receive email confirmations of completed actions

Restrictions:

- No account creation required, but limited session duration
- Cannot modify document content or signature positions
- Single-use tokens (cannot be reused after signing/rejection)
- 7-day token expiration for security

---

7. User Experience Considerations

7.1 Document Owner Dashboard

- **Clean Interface**: Minimalist design focusing on document status and actions
- **Filter and Search**: Quick filtering by status, date, and signer
- **Bulk Actions**: Select multiple documents for batch operations
- **Progress Indicators**: Visual progress bars for multi-signature workflows
- **Responsive Design**: Optimized for desktop and tablet usage

7.2 Signature Interface

- **PDF Rendering**: High-quality document preview with React-PDF
- **Signature Tools**: Intuitive signature creation with draw, type, and upload options
- **Drag-and-Drop**: Smooth signature field positioning with visual guides
- **Mobile Optimization**: Touch-friendly interface for smartphone signing
- **Progress Indication**: Clear steps showing signing progress

7.3 Accessibility Features

- **Keyboard Navigation**: Full keyboard accessibility for signature placement
- **Screen Reader Support**: ARIA labels and semantic HTML structure
- **High Contrast Mode**: Support for users with visual impairments
- **Font Size Controls**: Adjustable text size for better readability

---

8. Security Requirements

8.1 Authentication Security

- **JWT Implementation**: RSA256 algorithm with 15-minute access token expiration
- **Password Requirements**: Minimum 8 characters with complexity requirements
- **Brute Force Protection**: Account lockout after 5 failed login attempts
- **Session Management**: Secure refresh token rotation and blacklisting

8.2 Data Protection

- **Password Hashing**: bcrypt with 12 salt rounds minimum
- **Input Validation**: Comprehensive server-side validation for all endpoints
- **SQL/NoSQL Injection Prevention**: Parameterized queries and ORM usage
- **File Upload Security**: MIME type validation, file size limits, virus scanning

8.3 Communication Security

- **HTTPS Enforcement**: TLS 1.3 for all client-server communication
- **CORS Configuration**: Restrictive CORS policy for production environment
- **Rate Limiting**: API endpoint rate limiting to prevent abuse
- **Token Security**: Secure token generation with cryptographically strong randomness

8.4 Document Security

- **Access Control**: Token-based document access with expiration
- **Audit Logging**: Comprehensive logging of all document interactions
- **Tamper Detection**: Cryptographic signatures for document integrity
- **Secure Deletion**: Proper data sanitization for deleted documents

---

9. Development Timeline (14 Days)

Week 1: Foundation and Core Features

Days 1-2: Project Setup and Authentication

- Initialize MERN stack with development environment
- Set up MongoDB database with user schema
- Implement JWT authentication system
- Create user registration and login endpoints
- Build basic React frontend with routing
- Implement login/register forms with validation

Days 3-4: Document Management System

- Implement PDF upload functionality with Multer
- Create document storage and retrieval system
- Build document listing dashboard with pagination
- Implement PDF viewing with React-PDF
- Create document management API endpoints
- Add file validation and error handling

Days 5-7: Signature Workflow Implementation

- Build signature field positioning interface
- Implement drag-and-drop signature placement
- Create signature creation tools (draw, type, upload)
- Develop signature request generation system
- Build public signing interface with token validation
- Implement signature status tracking

Week 2: Advanced Features and Deployment

Days 8-9: Audit System and Document Finalization

- Implement comprehensive audit logging system
- Build audit trail viewing interface
- Create PDF signature embedding with PDF-Lib
- Develop final document generation with signatures
- Add email notification system with Nodemailer
- Implement signature rejection workflow

Days 10-11: Security and Testing

- Implement comprehensive input validation
- Add rate limiting and security headers
- Create automated tests for critical paths
- Perform security audit and penetration testing
- Optimize database queries and API performance
- Add error logging and monitoring

Days 12-13: UI/UX Polish and Mobile Optimization

- Refine user interface with Tailwind CSS
- Implement responsive design for mobile devices
- Add loading states and progress indicators
- Create user onboarding and help documentation
- Optimize PDF rendering performance
- Add accessibility improvements

Day 14: Deployment and Documentation

- Deploy frontend to Vercel/Netlify
- Deploy backend to Railway/Render
- Configure production environment variables
- Create comprehensive README documentation
- Record demo video showcasing all features
- Perform final testing in production environment

10. Deliverables

10.1 Technical Deliverables

- **Full-Stack Application**: Complete MERN-based document signature platform
- **Source Code**: Clean, well-documented codebase with proper version control
- **Database Schema**: Properly normalized database with indexes and constraints
- **API Documentation**: Comprehensive API documentation with example requests/responses
- **Deployment Configuration**: Docker containers and deployment scripts

10.2 Documentation Deliverables

- **README File**: Installation, setup, and usage instructions
- **Technical Documentation**: Architecture overview and development guidelines
- **User Guide**: End-user documentation with screenshots and workflows
- **Security Report**: Security measures implemented and recommendations
- **Test Coverage Report**: Automated test results and coverage metrics

10.3 Demonstration Deliverables

- **Demo Video**: 5-10 minute walkthrough of core functionality
- **Live Demo Environment**: Deployed application with sample data
- **Presentation Slides**: Technical presentation for stakeholders
- **Performance Metrics**: Application performance benchmarks

10.4 Quality Assurance

- **Cross-Browser Testing**: Compatibility verification across major browsers
- **Mobile Responsiveness**: Testing on various device sizes and orientations
- **Security Testing**: Vulnerability assessment and penetration testing
- **Load Testing**: Performance testing under concurrent user scenarios
- **Accessibility Audit**: WCAG 2.1 compliance verification

---

11. Risk Assessment and Mitigation

11.1 Technical Risks

- **PDF Processing Complexity**: Mitigation through thorough testing of PDF-Lib integration

- **Cross-Browser Compatibility**: Early testing across target browsers
- **Mobile Performance**: Optimization for low-bandwidth and older devices
- **File Upload Security**: Comprehensive validation and sanitization

11.2 Timeline Risks

- **Feature Scope Creep**: Strict adherence to defined MVP features
- **Third-Party Dependencies**: Backup plans for critical library failures
- **Deployment Issues**: Early deployment testing and rollback procedures

11.3 Security Risks

- **Authentication Vulnerabilities**: Regular security audits and updates
- **Data Breach Prevention**: Encryption at rest and in transit
- **Token Security**: Proper token management and expiration policies

---

12. Success Criteria

The DocuSign Clone project will be considered successful upon meeting the following criteria:

- All core features implemented and fully functional
- Security requirements met with no critical vulnerabilities
- Application deployed and accessible via public URLs
- Complete documentation and demo materials delivered
- Cross-browser compatibility verified
- Mobile responsiveness achieved
- Performance benchmarks met (page load < 3 seconds)
- Zero data loss or corruption during testing phase

Document Approval:

Role	Name	Signature	Date
Product Manager	[Name]		
Lead Developer	[Name]		
Security Architect	[Name]		
QA Lead	[Name]		

This PRD serves as the definitive guide for the DocuSign Clone development project and should be referenced throughout the development lifecycle for scope validation and feature verification.