SafeDocx Implementation Roadmap

Detailed Week-by-Week Development Plan

🎯 Overview

Total Duration: 12 weeks to MVP
Team Size: 1 developer
Goal: Launch production-ready DMS with core features

WEEK 1-2: Foundation & Setup

Week 1: Project Infrastructure

Backend Setup (Days 1-3):

- [x] Initialize Node.js project with Express
- [x] Configure PostgreSQL database
- [x] Set up Docker Compose (PostgreSQL, Redis, MinIO)
- [x] Create base models (User, Document, Folder)
- [x] Implement JWT authentication
- [x] Set up error handling & logging
- [ ] Configure environment variables
- [ ] Set up ESLint & Prettier

Frontend Setup (Days 4-5):

- [ ] Initialize Vite + React + TypeScript
- [ ] Install and configure Tailwind CSS
- [ ] Set up React Router
- [ ] Configure Zustand for state management
- [ ] Create base component library (Button, Input, Card)
- [ ] Set up Axios with interceptors

Deliverables: ✅ Working development environment
✅ Authentication API working
✅ Login page functional
✅ Protected routes working

Week 2: Core Document Features

Backend (Days 1-3):

- [ ] Complete Document CRUD API
- [ ] Implement file upload with Multer
- [ ] Configure MinIO for file storage
- [ ] Add document metadata system
- [ ] Create Folder CRUD API
- [ ] Implement basic search

Frontend (Days 4-5):

- [ ] Build Dashboard layout
- [ ] Create Document card component
- [ ] Implement upload zone with drag-and-drop
- [ ] Build document list (grid/list views)
- [ ] Add folder navigation
- [ ] Create search bar

Deliverables: ✅ Upload documents
✅ View documents
✅ Create folders
✅ Basic search working

WEEK 3-4: Document Types & Organization

Week 3: Document Types System

Backend (Days 1-3):

- [ ] Create DocumentType model
- [ ] Implement custom fields system
- [ ] Build DocumentType CRUD API

- [ ] Add field validation
- [ ] Create predefined document types:
    - Invoice
    - Contract
    - Employee Record
    - Receipt
    - Purchase Order

Frontend (Days 4-5):

- [ ] Build DocumentType builder UI
- [ ] Create custom field editor
- [ ] Add document type selector on upload
- [ ] Implement dynamic forms based on type
- [ ] Build document type management page

Deliverables: ✅ Create custom document types
✅ Upload with custom fields
✅ Filter by document type

Week 4: Stacks & Document Linking

Backend (Days 1-3):

- [ ] Create Stack model
- [ ] Implement StackDocument association
- [ ] Build Stack CRUD API
- [ ] Add document linking system
- [ ] Implement shared metadata

Frontend (Days 4-5):

- [ ] Create StackCard component
- [ ] Build Create Stack modal
- [ ] Implement stack viewer
- [ ] Add document linking UI
- [ ] Create related documents sidebar

Deliverables: ✅ Create document stacks
✅ Link related documents
✅ Shared metadata across stack

WEEK 5-6: Advanced Features

Week 5: Tagging & Advanced Search

Backend (Days 1-2):

- [ ] Implement tagging system
- [ ] Add Elasticsearch integration (optional)
- [ ] Build advanced search API
- [ ] Create saved searches
- [ ] Add search filters

Frontend (Days 3-5):

- [ ] Build tag management UI
- [ ] Create advanced search modal
- [ ] Implement filter sidebar
- [ ] Add saved search feature
- [ ] Build search results page

Deliverables: ✅ Tag documents
✅ Advanced filters
✅ Save search queries

Week 6: Version Control

Backend (Days 1-3):

- [ ] Create DocumentVersion model
- [ ] Implement check-in/check-out
- [ ] Add version comparison
- [ ] Build version restore API

Frontend (Days 4-5):

- [ ] Build version history UI
- [ ] Create version comparison viewer
- [ ] Implement check-in/check-out controls
- [ ] Add version restore functionality

Deliverables: ✅ Multiple document versions
✅ Check-in/check-out
✅ Restore previous versions

WEEK 7-8: Collaboration & Sharing

Week 7: Document Sharing

Backend (Days 1-3):

- [ ] Implement share permissions system
- [ ] Create share link generation
- [ ] Add email notifications
- [ ] Build access control middleware
- [ ] Implement share expiry

Frontend (Days 4-5):

- [ ] Build share modal
- [ ] Create permission selector
- [ ] Add share link UI
- [ ] Implement shared documents view
- [ ] Build access management interface

Deliverables: ✅ Share documents with users
✅ Generate share links
✅ Manage permissions
✅ Email notifications

Week 8: Comments & Annotations

Backend (Days 1-3):

- [ ] Create Comment model
- [ ] Implement mention system (@user)
- [ ] Build notifications system
- [ ] Add activity feed

Frontend (Days 4-5):

- [ ] Build comment thread UI
- [ ] Implement @mention autocomplete
- [ ] Create notification dropdown
- [ ] Build activity feed
- [ ] Add real-time updates (optional)

Deliverables: ✅ Comment on documents
✅ @mention users
✅ Notifications working
✅ Activity feed

WEEK 9-10: Workflows & Automation

Week 9: Basic Workflows

Backend (Days 1-3):

- [ ] Create Workflow model
- [ ] Implement WorkflowInstance
- [ ] Build approval system
- [ ] Add workflow routing
- [ ] Create email notifications

Frontend (Days 4-5):

- [ ] Build workflow builder
- [ ] Create approval interface
- [ ] Implement task list
- [ ] Add workflow status tracker
- [ ] Build workflow management page

Deliverables: ✅ Create workflows
✅ Submit for approval
✅ Approve/reject documents
✅ Email notifications

Week 10: Recycle Bin & Retention

Backend (Days 1-2):

- [ ] Implement soft delete
- [ ] Create Recycle Bin API
- [ ] Add auto-delete scheduler
- [ ] Implement retention policies
- [ ] Build document expiry system

Frontend (Days 3-5):

- [ ] Build Recycle Bin UI

- [ ] Add restore functionality
- [ ] Create retention policy manager
- [ ] Implement expiry date selector
- [ ] Add bulk operations UI

Deliverables: ✅ Recycle bin working
✅ Restore deleted items
✅ Auto-delete after period
✅ Set document expiry

WEEK 11-12: Polish & Launch

Week 11: Security & Performance

Backend (Days 1-3):

- [ ] Security audit
- [ ] Rate limiting implementation
- [ ] Database optimization
- [ ] API response caching
- [ ] Load testing

Frontend (Days 4-5):

- [ ] Performance optimization
- [ ] Image lazy loading
- [ ] Code splitting
- [ ] Bundle size reduction
- [ ] PWA configuration

Deliverables: ✅ Security hardened
✅ Performance optimized
✅ PWA working

Week 12: Testing & Deployment

Testing (Days 1-3):

- [ ] Unit tests (backend)
- [ ] Integration tests
- [ ] E2E tests (Cypress/Playwright)

- [ ] User acceptance testing
- [ ] Bug fixes

Deployment (Days 4-5):

- [ ] Production environment setup
- [ ] Database migration
- [ ] SSL certificate
- [ ] Domain configuration
- [ ] Monitoring setup (Sentry, Analytics)
- [ ] Backup strategy
- [ ] Documentation

Launch (Day 5):

- [ ] Deploy to production
- [ ] Smoke testing
- [ ] Launch announcement
- [ ] User onboarding

Deliverables: ✅ All tests passing
✅ Production deployed
✅ MVP LAUNCHED! 🎉

POST-MVP: Next 4 Weeks

Week 13-14: OCR & Email Capture

Features:

- OCR integration (Tesseract)
- Email document capture
- Folder watching
- Batch operations

Week 15-16: Mobile & Analytics

Features:

- Mobile responsive polish
- Analytics dashboard
- Reporting system
- API documentation (Swagger)

Development Best Practices

Daily Workflow

Morning (9 AM - 12 PM):

- Stand-up meeting (15 min)
- Code development
- Pair programming sessions

Afternoon (1 PM - 5 PM):

- Code reviews
- Testing
- Bug fixes
- Documentation

End of Day:

- Git commit & push
- Update task board
- Daily status update

Git Workflow

```
# Feature branches
git checkout -b feature/document-types
# Work on feature
git add .
git commit -m "feat: implement document types"
git push origin feature/document-types
# Create Pull Request
# Code review
# Merge to develop
```

Branch Structure:

- main - Production code
- develop - Development branch
- feature/* - New features
- bugfix/* - Bug fixes

- hotfix/* - Production hotfixes

Testing Strategy

Backend Tests:

```
// Unit tests for each model/controller
describe('Document API', () => {
  it('should create a document', async () => {
    // Test implementation
  });

  it('should require authentication', async () => {
    // Test implementation
  });
});
```

Frontend Tests:

```
// Component tests
describe('DocumentCard', () => {
  it('renders document information', () => {
    // Test implementation
  });
});
```

Code Review Checklist

Backend:

- [ ] API follows RESTful conventions
- [ ] Proper error handling
- [ ] Input validation
- [ ] Authentication/authorization
- [ ] Database transactions where needed
- [ ] Tests written
- [ ] Documentation updated

Frontend:

- [ ] Component is reusable

- [ ] TypeScript types defined
- [ ] Accessibility (a11y) considered
- [ ] Responsive design
- [ ] Error states handled
- [ ] Loading states shown
- [ ] Tests written

Resource Allocation

Team Structure

2-Person Team:

- **Developer 1:** Backend focus (60%), Frontend (40%)
- **Developer 2:** Frontend focus (60%), Backend (40%)

4-Person Team:

- **Backend Lead:** API, Database, Infrastructure
- **Backend Developer:** Features, Testing
- **Frontend Lead:** UI/UX, Components
- **Frontend Developer:** Pages, Integration

Key Milestones

Milestone 1 (Week 2)

✅ Basic upload/download working
✅ Authentication complete

Milestone 2 (Week 4)

✅ Document organization done
✅ Custom document types

Milestone 3 (Week 6)

✅ Search fully functional
✅ Version control working

Milestone 4 (Week 8)

✅ Sharing implemented
✅ Collaboration features

Milestone 5 (Week 10)

✅ Workflows complete
✅ Automation working

Milestone 6 (Week 12)

✅ MVP LAUNCHED

Risk Management

Potential Risks

Technical Risks:

1. **File Storage Scaling**
    - Risk: Storage costs too high
    - Mitigation: Use MinIO, implement compression
2. **Search Performance**
    - Risk: Slow search on large datasets
    - Mitigation: Elasticsearch, database indexes
3. **Security Vulnerabilities**
    - Risk: Data breach
    - Mitigation: Security audit, penetration testing

Project Risks:

1. **Scope Creep**
    - Risk: Adding too many features
    - Mitigation: Stick to MVP scope, Phase 2 for extras
2. **Timeline Delays**

    - Risk: Missing deadlines
    - Mitigation: Weekly check-ins, adjust scope

Success Metrics

Technical KPIs

Performance:

- API response time < 200ms
- Page load time < 2 seconds
- Uptime > 99.9%

Usage:

- 100+ documents uploaded (first week)
- 50+ active users (first month)
- 5+ document types created

Quality:

- 0 critical bugs in production
- 80%+ code coverage
- 90%+ user satisfaction

Budget Estimate

Development Costs (12 weeks)

Team (2 developers):

- Senior Developer: $6,000/month × 3 months = $18,000
- Mid Developer: $4,000/month × 3 months = $12,000
- **Total Labor:** $30,000

Infrastructure:

- Domain & SSL: $50
- Cloud hosting (dev): $100/month × 3 = $300
- Testing tools: $100
- **Total Infra:** $450

Tools & Services:

- GitHub: $0 (free tier)
- VS Code: $0 (free)
- Figma: $15/month × 3 = $45
- **Total Tools:** $45

Total MVP Cost: ~$30,500

Ongoing Costs (Monthly)

- Cloud hosting: $200
- Database: $100
- Storage (S3/MinIO): $50
- Monitoring: $50
- **Total Monthly:** $400

Documentation Plan

Required Documentation

1. **API Documentation**
    - Swagger/OpenAPI spec
    - Example requests/responses
    - Authentication guide
2. **User Guide**
    - Getting started
    - Feature tutorials
    - FAQs
3. **Admin Guide**
    - Installation
    - Configuration
    - Troubleshooting
4. **Developer Guide**
    - Architecture overview
    - Setup instructions
    - Contributing guidelines

Launch Checklist

Pre-Launch (1 week before):

- [ ] All features tested
- [ ] Security audit passed
- [ ] Performance benchmarks met
- [ ] Backup strategy in place
- [ ] Monitoring configured
- [ ] Support channels ready
- [ ] Marketing materials prepared

Launch Day:

- [ ] Deploy to production
- [ ] Verify all features working
- [ ] Monitor error rates
- [ ] Watch system metrics
- [ ] Respond to user feedback

Post-Launch (1 week after):

- [ ] Collect user feedback
- [ ] Fix critical bugs
- [ ] Monitor usage patterns
- [ ] Plan Phase 2 features

You're Ready to Build! 🚀

This roadmap gives you a clear path from Day 1 to MVP launch. Focus on one week at a time, deliver working features incrementally, and adjust as needed based on feedback.

Next Step: Start with Week 1, Day 1 - Backend setup! 💪