🔒 Data Loss Prevention (DLP) System

Project Report

📋 Project Overview

Project Name: Data Loss Prevention (DLP) System
Project Type: Web-Based Security Application
Technology: Node.js, Express.js, MySQL, HTML/CSS/JavaScript
Team Size: 4 Members

Abstract

The Data Loss Prevention (DLP) System is an enterprise-grade web application designed to protect organizations from accidental or intentional data breaches. It scans uploaded documents for sensitive information such as passwords, credit card numbers, social security numbers, PAN cards, Aadhar numbers, and other confidential data patterns. The system provides real-time detection, comprehensive audit trails, and administrative controls for security policy management.

🎯 Objectives

1. Detect and prevent sensitive data leakage through document scanning
2. Provide real-time alerts for policy violations
3. Maintain comprehensive audit logs for compliance
4. Enable administrators to define custom security policies
5. Support multiple document formats (TXT, PDF, DOC, DOCX)

🚀 Key Features

User Features

- **Secure Authentication** - User registration and login with encrypted passwords
- **Document Upload** - Drag-and-drop file upload interface
- **Real-time Scanning** - Instant detection of sensitive data patterns
- **Scan History** - View past scan results and violations
- **Profile Management** - Update personal information

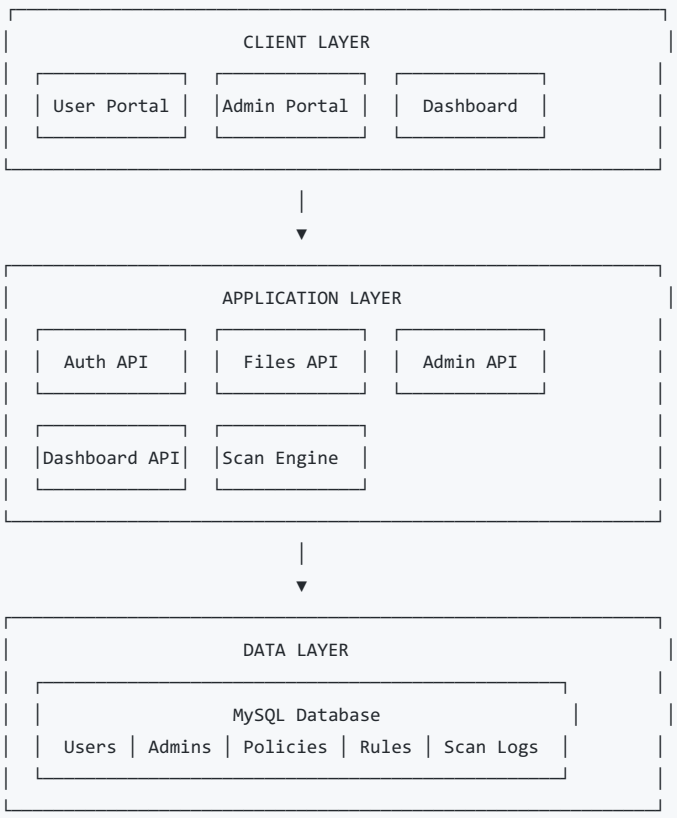
Admin Features

- **Dashboard Analytics** - Real-time statistics and metrics
- **Incident Management** - View and manage security violations
- **Policy Management** - Create and configure detection policies
- **Rule Configuration** - Define custom detection patterns
- **Audit Logs** - Complete activity tracking
- **User Management** - Monitor user activities
- **System Configuration** - Configure scanning parameters

🛠️ Technology Stack

Layer	Technology
Frontend	HTML5, CSS3, JavaScript (ES6+)
Backend	Node.js, Express.js
Database	MySQL
Authentication	bcrypt, Session Tokens
File Processing	Multer, pdf-parse, mammoth
Styling	Custom CSS with Modern Design

System Architecture



Database Schema

Tables Overview

Table	Purpose
`users`	Store user account information
`admin_users`	Store admin account information
`sessions`	Manage active sessions
`security_policies`	Define security policy groups
`sensitive_data_rules`	Detection patterns and rules
`file_scan_logs`	Record of all file scans
`detected_violations`	Details of detected sensitive data
`audit_logs`	System activity tracking
`system_config`	Application configuration

Detection Patterns

The system detects the following sensitive data types:

Data Type	Severity	Example Pattern
Credit Card Numbers	Critical	4532-XXXX-XXXX-XXXX
Social Security Numbers (SSN)	Critical	XXX-XX-XXXX
PAN Card Numbers	High	ABCDE1234F
Aadhar Numbers	High	XXXX XXXX XXXX
Passwords	High	password, secret, api_key
Email Addresses	Medium	user@domain.com
Confidential Keywords	Medium	confidential, internal

Project Structure

```
DLP/
├── backend/
│   ├── config/
│   │   └── database.js          # Database connection
│   ├── controllers/
│   │   ├── authController.js    # Authentication logic
│   │   ├── fileController.js    # File handling logic
│   │   ├── adminController.js   # Admin operations
│   │   └── dashboardController.js # Analytics
│   ├── middleware/
│   │   └── auth.js              # Session verification
│   ├── routes/
│   │   ├── auth.js              # Auth endpoints
│   │   ├── files.js             # File endpoints
│   │   ├── admin.js             # Admin endpoints
│   │   └── dashboard.js         # Dashboard endpoints
│   ├── utils/
│   │   ├── encryption.js        # Password hashing
│   │   └── scanner.js           # Sensitive data scanner
│   ├── server.js                # Main server file
│   └── init.js                  # Database initialization
│
├── frontend/
│   ├── css/
│   │   └── style.css            # Styling
│   ├── js/
│   │   ├── api.js               # API utilities
│   │   └── admin.js             # Admin functions
│   ├── index.html               # User portal
│   ├── user_dashboard.html      # User dashboard
│   ├── admin_login.html         # Admin login
│   └── admin_dashboard.html     # Admin dashboard
│
├── db/
│   └── schema.sql               # Database schema
│
└── uploads/                     # Uploaded files storage
```

Team Work Distribution

Member 1: Frontend Development

Responsibilities:

- Design and implement user interface (HTML/CSS)
- Create responsive layouts for all pages
- Implement user portal (login, registration, dashboard)
- Design admin portal interface
- Handle form validations on client-side
- Create notification system
- Ensure cross-browser compatibility

Files to Work On:

- `frontend/css/style.css`
- `frontend/index.html`
- `frontend/user_dashboard.html`
- `frontend/admin_login.html`
- `frontend/admin_dashboard.html`

Member 2: Backend API Development

Responsibilities:

- Set up Node.js/Express server
- Create RESTful API endpoints
- Implement authentication system (login, register, sessions)
- Handle file upload and processing
- Create middleware for request validation
- Implement error handling
- Set up CORS and security headers

Files to Work On:

- `backend/server.js`
- `backend/routes/auth.js`
- `backend/routes/files.js`
- `backend/routes/admin.js`
- `backend/routes/dashboard.js`
- `backend/middleware/auth.js`

Member 3: Database & Scanning Engine

Responsibilities:

- Design database schema
- Implement database connection and queries
- Create sensitive data scanning algorithms
- Implement pattern matching for various data types
- Handle file content extraction (PDF, DOC, DOCX)
- Create violation detection logic
- Optimize database queries for performance

Files to Work On:

- `db/schema.sql`
- `backend/config/database.js`
- `backend/utils/scanner.js`
- `backend/utils/encryption.js`
- `backend/controllers/fileController.js`

Member 4: Admin Module & Documentation

Responsibilities:

- Implement admin dashboard functionality
- Create policy and rules management system
- Implement audit logging system
- Create analytics and reporting features
- Handle system configuration
- Write project documentation
- Perform testing and bug fixes
- Create user manual

Files to Work On:

- `backend/controllers/adminController.js`
- `backend/controllers/dashboardController.js`

- `frontend/js/admin.js`
- `frontend/js/api.js`
- `README.md`
- `PROJECT_REPORT.md`

Installation & Setup

Prerequisites

- Node.js (v14 or higher)
- MySQL Server
- XAMPP (optional, for Apache)

Step 1: Database Setup

```
-- Run in MySQL
CREATE DATABASE dlp_system;
USE dlp_system;
-- Then run schema.sql
```

Step 2: Install Dependencies

```
cd c:\xampp\htdocs\DLP\backend
npm install
```

Step 3: Initialize Database

```
node init.js
```

Step 4: Start Server

```
node server.js
```

Step 5: Access Application

- **User Portal:** http://localhost:3000/index.html
- **Admin Panel:** http://localhost:3000/admin_login.html
- **Admin Credentials:** admin / admin123

Future Enhancements

1. **Machine Learning Integration** - AI-based pattern detection
2. **Email Notifications** - Alert admins via email for critical violations
3. **Role-Based Access Control** - Multiple admin roles with different permissions
4. **Cloud Storage Integration** - Scan files from cloud services
5. **API Integration** - REST API for third-party integrations
6. **Mobile Application** - React Native mobile app
7. **Advanced Reporting** - PDF report generation with charts

---

Conclusion

The Data Loss Prevention System successfully provides a comprehensive solution for detecting and preventing sensitive data leakage. The application features a modern, user-friendly interface with robust backend processing capabilities. The modular architecture allows for easy maintenance and future enhancements.

References

1. Node.js Documentation - https://nodejs.org/docs

2. Express.js Guide - https://expressjs.com
3. MySQL Documentation - https://dev.mysql.com/doc
4. OWASP Data Loss Prevention - https://owasp.org
5. GDPR Compliance Guidelines

Project Submitted By:

- Member 1: _____ (Frontend Development)
- Member 2: _____ (Backend API Development)
- Member 3: _____ (Database & Scanning Engine)
- Member 4: _____ (Admin Module & Documentation)

Date: December 2025
