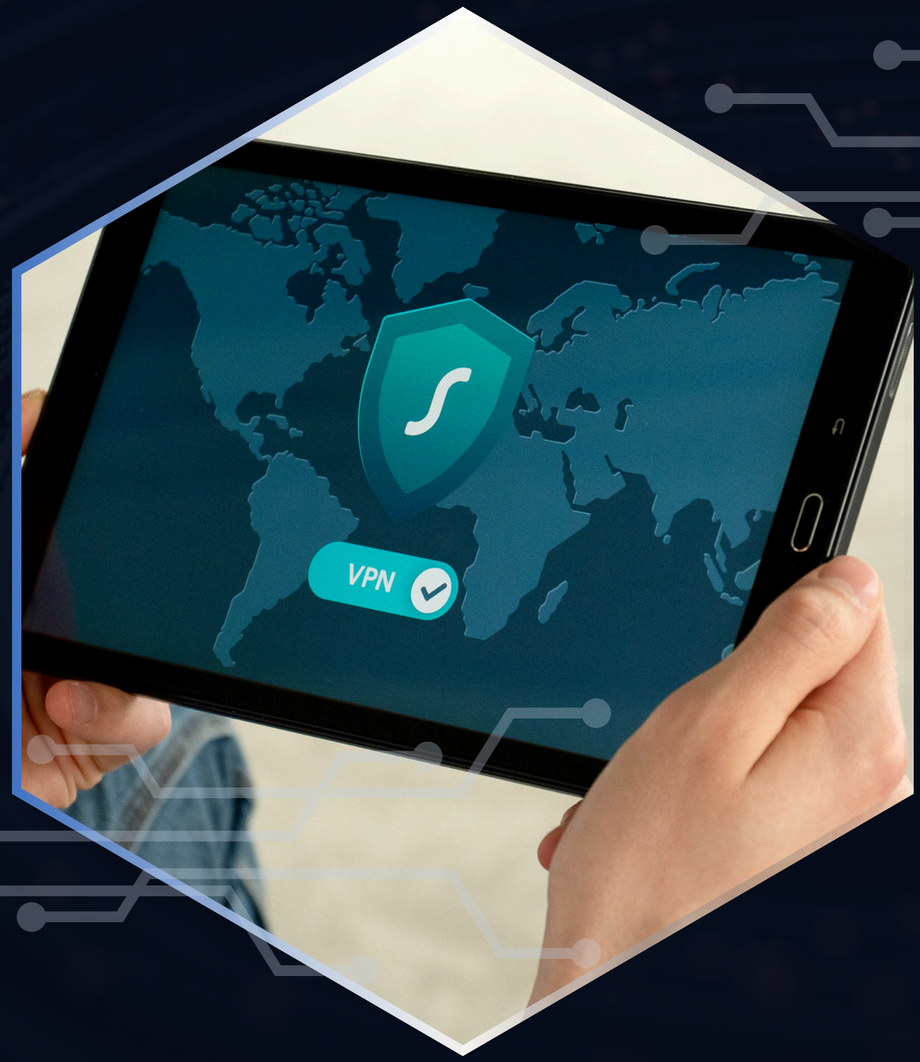
DATAQUEST 2.0 – DQCS_3

AUTOMATED DATA LEAK PREVENTION (DLP) SYSTEM FOR INTERNAL FILES

Team: NextGen

PROBLEM STATEMENT

- Organizations struggle to prevent sensitive data from leaking through internal file shares or email systems.
- Manual monitoring is inefficient and error-prone.
- Challenge: Create an automated tool that detects transmission of sensitive data (like passwords, PII, or intellectual property).
- Must take actions in real-time such as blocking or alerting.



CATEGORIES OF INTERNAL DATA LEAK PROBLEMS

Category	How It Happens	Examples
Electronic Communication Leaks	Data leaves via digital channels	- Sending sensitive files to personal emails - Wrong CC in emails - Posting code/API keys publicly - Unauthorized cloud storage (Dropbox, Google Drive) - Sharing info via chat platforms (Slack, Teams, WhatsApp)
Endpoint & Physical Leaks	Data leaves via devices or physical media	- Copying files to USB drives - Printing and taking documents out - Screenshots of sensitive data - Files on lost/stolen laptops
Internal Storage Misuse	Data is exposed internally due to poor access control	- Saving sensitive files to public network drives - Misconfigured databases exposed online - Uploading files with overly broad access permissions

PROJECT OBJECTIVE

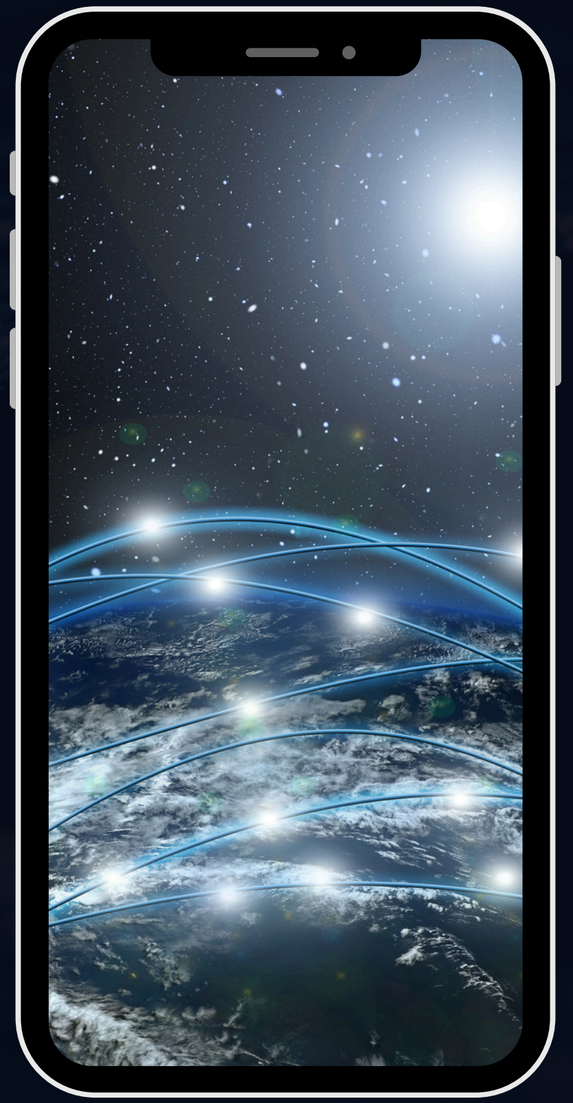
- Build an automated DLP system to detect sensitive data in files and emails.
- Classify files/emails based on sensitivity levels.
- Generate real-time alerts or block suspicious transmissions.
- Maintain audit logs for compliance and forensic analysis.

HOW WE SOLVE IT

- Use regex patterns to scan files and emails for sensitive data signatures (e.g., credit card numbers, SSNs).
- Train machine learning models to classify files/emails as confidential, internal, or public.
- Implement an automated policy engine for real-time actions (alerts/blocks).
- Maintain audit logs for traceability and compliance.

PROJECT ARCHITECTURE

- Input: Files / Emails
- Step 1: Regex Scanner (Email, Credit Card, SSN)
- Step 2: ML Classifier (Confidential / Internal / Public)
- Step 3: Policy Engine (Alert / Block)
- Step 4: Logging System (Audit logs / Forensics)
- Output: Alerts / Blocked files / Logs



KEY FEATURES

- Detect sensitive data in files & emails.
- Classify content using ML zero-shot classifier.
- Automated actions: alert or block suspicious files.
- Audit logs: track user activity for compliance.
- Extensible: new patterns or ML models can be added easily.

TOOLS & TECHNOLOGIES

- Programming Language: Python
- Web Framework: Flask / FastAPI
- ML Library: Hugging Face Transformers (facebook/bart-large-mnli)
- Regex Engine: Python re module
- Logging: SQLite / JSON
- Frontend: HTML, CSS
- Version Control: Git + GitHub

PROJECT OUTCOME

- The system automates detection of sensitive data in files and emails.
- Reduces manual monitoring efforts.
- Provides real-time alerts and audit logging for compliance.
- Scalable and extensible for future improvements.