

Abnormal File Vault – Product Requirements

Overview

Abnormal File Vault is a **file hosting application** designed to optimize storage efficiency and enhance file retrieval through **deduplication and intelligent search capabilities**. The project consists of a **React frontend** and a **Django backend**, containerized using **Docker** for easy setup and deployment.

A **starter project** will be provided as a [**ZIP file**](#), containing the base structure for the frontend and backend. Candidates are expected to extend and enhance this project to implement the required functionality.

Time Expectation

This project is designed to be completed in approximately **2 – 4 hours**.

Business Case

As Abnormal Security continues to build **AI-powered security solutions**, efficient data storage and retrieval are essential for managing files, reports, and forensic evidence related to cybersecurity threats. A **smart file management system** like Abnormal File Vault could provide:

- **Optimized Storage** – Reducing redundancy through deduplication lowers storage costs and improves performance.
- **Faster Incident Investigations** – A powerful search and filtering system enables security teams to retrieve relevant files quickly.
- **Scalability & Performance** – Handling large datasets efficiently ensures seamless operations as the system scales.
- **Improved User Experience** – A well-structured file system enhances usability for internal teams and customers.

The ability to **intelligently store, organize, and retrieve files** aligns with Abnormal Security's mission to streamline and automate security workflows.

Technical Requirements

- **Frontend:** React
- **Backend:** Django/DRF
- **Database:** SQLite

- **Containerization:** Docker

Features & Functionality

Below two features are to be added to existing starter project to full fill the business need

1 File Deduplication System

Objective: Optimize storage efficiency by detecting and handling duplicate file uploads.

Requirements:

- Identify duplicate files during upload
- Store references to existing files instead of saving duplicates
- Provide a way to track and display storage savings

2 Search & Filtering System

Objective: Enable efficient retrieval of stored files through search and filtering options.

Requirements:

- Search files by filename
- Filter files by:
 - File type
 - Size range
 - Upload date
- Allow multiple filters to be applied simultaneously
- Optimize search performance for large datasets

Development Guidelines

- A **starter project** will be provided as a [ZIP file](#), including the initial frontend and backend setup
- Please follow through the README.md in the starter project for setup instructions
- Follow best practices for **code organization, maintainability, and performance**
- Optimize queries and indexing for efficient file searches

This document outlines the core functionality and business case for the project. The implementation should focus on **efficiency, scalability, and clean code**, while adhering to the given time constraints.