**Overview of the Implementation Process**

**Step 1: Architecture Design**

* We designed our system to work with either a **centralized index server** (which keeps track of stored files) or a **peer-to-peer model** (where each node shares storage responsibility dynamically).
* The system ensures that files are evenly distributed across storage nodes, reducing bottlenecks and preventing single points of failure.

**Step 2: Technology Stack**

* **Backend:** Python (Flask, FastAPI) / Node.js
* **Storage:** MySQL / MongoDB / Local File System
* **Networking:** REST API / gRPC for communication between nodes
* **Containerization:** Docker for deployment and scalability

**Step 3: Implementation Steps**

1. **Set up storage nodes** – Each node is configured to store, retrieve, and replicate files.
2. **Implement file distribution & retrieval** – A hashing algorithm is used to distribute files efficiently.
3. **Develop REST API** – Users interact with the system via APIs for uploading and retrieving files.
4. **Add redundancy & fault tolerance mechanisms** – Replication ensures no data is lost when a node fails.
5. **Test & optimize performance** – Conducted failure tests and load balancing to ensure system stability.