**Storage Microservice Design Document**

## 1) Function and Nonfunctional Requirements

### Functional Requirements:

* **File Storage:**
  + Store files of any extension securely.
  + Support operations like upload, download, and delete.
* **Access Control:** 
  + Implement authentication and authorization mechanisms to control access. Not implemented in this phase
* **Scalability:**
  + Scale horizontally to handle increasing file storage needs.
* **Fault Tolerance:**
  + Ensure high availability and resilience against failures.
* **Monitoring and Logging:**
  + Provide monitoring metrics and logging for operations. Not implemented in this phase

### Nonfunctional Requirements:

* **Performance:**
  + Response time for file operations should be optimized.
* **Security:**
  + Encrypt files at rest and in transit to maintain data confidentiality.
* **Compliance:**
  + Adhere to regulatory compliance requirements for data storage.
* **Integration:**
  + Easily integrate with other microservices and applications.
* **Documentation:**
  + Comprehensive documentation for API usage and system architecture.

## 2) High-Level Design

### Components:

* **API Gateway:** Routes requests to appropriate services.
* **Storage Service:** Handles file storage operations.
* **Authentication Service:** Manages user authentication and authorization. Not implemented in this phase

### Architecture:

* **Microservices Architecture:** Uses a modular approach for scalability and flexibility.
* **RESTful API:** Exposes endpoints for file operations.
* **Event-Driven Architecture:** Publishes events for file upload, download, and deletion.

## 3) Low-Level Design

### Storage Types: depend on configurations

##### Local Storage

* **Description:**
  + Files are stored on the local file system of the microservice server or  
     any server on-premises.
  + Suitable for small-scale deployments or when direct access to files is needed.
* **Pros:**
  + Simple setup and management.
  + Direct access to files without network latency.
* **Cons:**
  + Limited scalability compared to cloud storage.
  + Risk of data loss if server fails or disk malfunctions.

##### Azure Blob Storage

* **Description:**
  + Scalable cloud storage solution provided by Microsoft Azure.
  + Stores unstructured data as files share accessible via URLs.
* **Reason for Choosing:**
  + Highly scalable, durable, and available.
  + Integrated with Azure ecosystem, supports global distribution and CDN.

#### File Operations

* **Upload File:**
  + Client sends file to API Gateway.
  + Gateway routes request to Storage Service.
  + Service saves file to chosen storage depend on configuration (local or Azure Storage), returns file metadata.
* **Download File:**
  + Client requests file by name.
  + Gateway forwards request to Storage Service.
  + Service retrieves file from chosen storage, streams it to client.
* **Delete File:**
  + Client requests file deletion.
  + Gateway routes request to Storage Service.
  + Service deletes file from chosen storage.

## 4) Communication with Other Microservices

### API Gateway (Ocelot):

* Routes requests from other microservices to appropriate endpoints.
* Manages API requests and responses, handles authentication and authorization.

### Example Flow:

* **Microservice A** needs to upload a file:
  1. Microservice A sends a POST request to /storage/upload via API Gateway.
  2. API Gateway forwards request to Storage Microservice.
  3. Storage Microservice saves file to Azure Storage.
  4. Storage Microservice returns file metadata to Microservice A.

