PART 3

AWS Architecture Diagram

Here I am mentioning the components I can think of for the given problem statement in part 3.

Key components:

1. Data Ingestion: The process starts with the user uploading data to an Amazon S3 bucket as this is a highly scalable storage service provided by AWS.

2. Data Processing: An Amazon EventBridge rule is set to trigger an AWS Lambda function whenever a new object is added to the S3 bucket. This Lambda function is responsible for processing the data and storing the processed information in an Amazon DynamoDB table or any other DB services approved by management.

3. Error Handling: If any errors occur during the data processing, the Lambda function will send a notification to the user's contact details using Amazon SNS (Simple Notification Service). This ensures that the user is promptly informed about the issue.

4. Search Functionality: To enable high-performing search queries, the processed data is stored in an Amazon DynamoDB table. DynamoDB is a fast and flexible NoSQL database service that can handle large amounts of data and provide low-latency performance for any applications.

5. Search Queries: The user can perform search queries against the DynamoDB table using the AWS SDK or the AWS Management Console or if we choose to query in python specially, we can use boto3 as SDK and also leverage moto for mocking the services like S3, DynamoDB etc. DynamoDB's flexible schema and indexing capabilities allow for efficient querying based on various attributes, such as color, size, material, etc.

6. Monitoring and Logging: Amazon CloudWatch is used to monitor the overall system, including the Lambda function's execution, any errors or alarms, and the DynamoDB table's performance. This provides visibility into the system's health and helps with troubleshooting and optimization.

AWS Service Diagram

The key AWS services used in this architecture are:

1. Amazon S3: Highly scalable and durable object storage service for storing the user's uploaded data.

2. Amazon EventBridge: Event bus service that triggers the Lambda function when a new object is added to the S3 bucket.

3. AWS Lambda: Serverless compute service that processes the data and stores the processed information in DynamoDB.

4. Amazon DynamoDB: Fast and flexible NoSQL database service for storing the processed data and enabling high-performing search queries.

5. Amazon SNS: Fully managed pub/sub messaging service for sending notifications to the user in case of any errors during data processing.

6. Amazon CloudWatch: Monitoring and observability service for tracking the system's health and performance.

7. And my favorite, if we need GraphQL like service, on top of the above services, we can add layer of AppSync, and also get most out from resolvers designing as well.

8. Apart from this, if we decide to use Django or FastAPI as backend handling is easy by using Swagger UI and ReDoc, we also can deploy them on Lambda with API Gateway using wsgi features of Django and FastAPI.

9. For managing the secrets, we can definitely choose SecretsManager with IAM.

This architecture leverages various AWS services to provide a scalable, reliable, and efficient solution for the given requirements. The use of serverless components, such as Lambda and DynamoDB, ensures that the system can automatically scale to handle varying workloads without the need for manual provisioning or management of infrastructure.
I am not adding a diagram here as it is really hard to use the tool mentioned in the problem statement. Hence I just gave a brief description above.