Steps Followed:   
  
**Work Plan**

1. **Setup Development Environment** (30 minutes)
   * Follow Spring tutorial to setup Spring Boot/Java project
   * Setup PostgreSQL database and run provided DDL file to create tables and insert sample data
2. **Implement RESTful APIs for Data Collections** (2 hours)
   * Implement CRUD operations for Data Collections (create, read, update, delete)
   * Implement API to list Data Collections with filtering and sorting capabilities
3. **Develop Test Cases for Data Collections** (30 minutes)
   * Write a list of test cases to validate Data Collections API functionality

**Technical Aspects**

* Use Spring Boot and Java to create RESTful APIs
* Use PostgreSQL as the database and run provided DDL file to setup tables and sample data
* Implement CRUD operations for Data Collections using standard RESTful API patterns
* Use filtering and sorting capabilities to list Data Collections
* Write test cases to validate Data Collections API functionality (note: do not implement test cases)

**Diagram**

Here is a simple diagram showing the architecture of the application:

    +---------------+

| Client |

+---------------+

|

|

v

+———————+.

| Spring Boot |

| RESTful API |

+---------------+

|

|

v

+---------------+

| PostgreSQL |

| Database |

+---------------+    ….>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>   +---------------+

| Client |

| (Web Interface)|

+---------------+

|

|

v

+---------------+

| Spring Boot |

| RESTful API |

| (Data Collections)|

+---------------+

|

|

v

+---------------+

| Validation |

| (Data File Type, |

| Data File Validation,|

| Data Collection Integrity)|

+---------------+

|

|

v

+---------------+

| PostgreSQL |

| Database |

| (Data Files, |

| Data Collections)|

+---------------+

|

|

v

+---------------+

| Data File |

| (Input Information)|

| (Type 1, 2, or 3) |

+---------------+

|

|

v

+---------------+

| Data Collection |

| (Set of Data Files)|

| (One of each type) |

+---------------+   
  
  
  
  
Summary

The DataCollectionService class is a service class that provides methods for creating, updating, listing, and deleting data collections. It also includes a method for validating data files associated with a data collection.

Example Usage

DataCollectionService dataCollectionService = new DataCollectionService(dataCollectionRepository, dataFileRepository);

// Create a new data collection

DataCollection newDataCollection = new DataCollection();

newDataCollection.setCreatedOn(new Timestamp(System.currentTimeMillis()));

newDataCollection.setFileOrders(fileOrders);

newDataCollection.setFileAssets(fileAssets);

newDataCollection.setFileInventory(fileInventory);

newDataCollection.setStatus("Pending");

DataCollection createdDataCollection = dataCollectionService.createDataCollection(newDataCollection);

// Update an existing data collection

Long dataCollectionId = 1L;

DataCollection updatedDataCollection = dataCollectionService.getDataCollectionById(dataCollectionId).orElseThrow();

updatedDataCollection.setStatus("Completed");

DataCollection updatedCollection = dataCollectionService.updateDataCollection(dataCollectionId, updatedDataCollection);

// List data collections sorted by createdOn in descending order

List<DataCollection> sortedDataCollections = dataCollectionService.listDataCollectionsSortedBy("created\_on");

// Delete a data collection

Long dataCollectionIdToDelete = 2L;

dataCollectionService.deleteDataCollection(dataCollectionIdToDelete);

Code Analysis

Main functionalities

Create a new data collection

Update an existing data collection

List data collections sorted by a specified field

Delete a data collection

Validate data files associated with a data collection

Methods

createDataCollection(newDataCollection: DataCollection): DataCollection: Creates a new data collection by saving it to the data collection repository after validating the associated data files.

updateDataCollection(id: Long, updatedDataCollection: DataCollection): DataCollection: Updates an existing data collection by finding it in the data collection repository, updating its fields with the provided values, and saving it back to the repository.

listDataCollectionsSortedBy(sortBy: String): List<DataCollection>: Lists all data collections sorted by the specified field. If no field is provided, it returns all data collections.

deleteDataCollection(id: Long): void: Deletes a data collection by its ID if it exists in the data collection repository.

validateDataFiles(dataCollection: DataCollection): void: Validates the data files associated with a data collection by checking if they exist in the data file repository.

getDataCollectionById(id: Long): Optional<DataCollection>: Retrieves a data collection by its ID from the data collection repository.

Fields

dataCollectionRepository: DataCollectionRepository: The repository for data collections.

dataFileRepository: DataFileRepository: The repository for data files.

logger: Logger: The logger for logging messages in the DataCollectionService class.  
