Task Breakdown and Assignment

Your Name

Java, Java Swing

User Interface Lalida Krairit

1. Catalog Interface

- **Task**:

  Implement the GUI to display search options (by annotation, date, file type, name) and file details. The interface should also collect user input for file management actions (view, move, delete, add).

  - Implement the GUI components using **Java Swing** (JFrame, JPanel, JTextField, JButtons).
  - Validate user input on the client side to improve UX.
  - Use **ActionListeners** for buttons and input fields to trigger actions in the catalog logic.
  - Display search results using components like **JTable**.

- **Implementation Details**:
  - Java Swing with **MVC** pattern.
  - Swing components for user interactions (e.g., **JTable** to list search results).
  - Pass validated data from UI to logic via event listeners.

2. Curation Interface

- **Task**:

  Implement the GUI for functions to search files on disk, add files to the catalog, delete files, compare files, and validate the catalog.

  - Design the interface using **Swing** for buttons and dialog boxes to handle file comparison, deletion, and validation.
  - Display comparison results using a text panel or a more advanced diff visualization component.
  - Provide visual feedback when files are added or deleted.

- **Implementation Details**:
  - Java Swing GUI for file management actions.
  - Use **JFileChooser** for file selection when adding files from disk.

3. File Comparison Viewer

- **Task**:

  Create the visual component to display differences between files, highlighting changes.

  - Implement file comparison UI using **JTextArea** or similar components to show file differences.
  - Use color highlighting to show added, removed, or modified lines.
- **Implementation Details**:
  - Swing-based diff viewer using **JTextArea** or custom painting on a **JPanel**.
  - External libraries for diff highlighting could be used, like **JDiff**.

Logic Implementation Sunidhi Pruthikosit

4. Catalog Logic

- **Task**:

  Control the search logic for files in the catalog based on user input (annotation, date, file type, or file name). Supports editing annotations and moving files.

  - Develop methods to perform searches in the catalog using different criteria.
  - Implement file-moving logic to update metadata when a file's location is changed.
  - Add functionality for editing annotations.
- **Implementation Details**:
  - Use **Java Collections** or an **SQL database** for storing and searching through file metadata.
  - Use **Observer pattern** to communicate changes back to the UI.

5. Curation Logic

- **Task**:

  Handle the logic for searching source files on disk, adding them to the catalog, deleting them, comparing files, and validating the catalog.

  - Develop logic to browse the file system and retrieve files.
  - Implement methods to curate (add, delete) files in the catalog based on user input.
  - Link with the **Disk Comparison Module** and **File Validation Module** to ensure catalog accuracy.
- **Implementation Details**:
  - Use **Java File I/O** for browsing files on disk.
  - Implement curation logic using a **Service Layer** pattern to separate disk operations and catalog updates.

Backend Implementation Niracha Janavatara

6. File Catalog

- **Task**:

  Store metadata about files (annotations, modification date, etc.) and facilitate data transfer between catalog logic and curation logic.

  - Implement a database or collection that stores file metadata.
  - Ensure data transfer between catalog logic and curation logic.
  - Implement CRUD operations for the catalog.

- **Implementation Details**:
  - Use **SQLite** or **H2 Database** for storing catalog data.
  - Alternatively, use **Java Collections** (e.g., **HashMap** or **ArrayList**) for an in-memory catalog.
  - Provide methods for adding, deleting, and updating metadata entries.

7. Catalog Comparison Module

- **Task**:

  Compare the content of files based on user selection, showing differences.

  - Implement the file comparison logic (e.g., compare lines between two files).
  - Send the comparison data to the UI for display.

- **Implementation Details**:
  - Use **Java BufferedReader** or **Apache Commons Text** library for line-by-line file comparison.
  - Create methods that return the comparison result as a structured format (e.g., list of changes).

8. Disk Comparison Module

- **Task**:

  Compare files in the catalog with actual files on disk to detect differences or missing files.

  - Develop logic to compare file contents and metadata on disk against catalog entries.
  - Report any discrepancies between disk files and catalog entries.

- **Implementation Details**:
  - Use **Java NIO** package for file access and comparison.
  - Implement comparison algorithms that check for changes in file size, modification date, or content.

9. File Validation Module

- **Task**:

  Ensure catalog data aligns with actual files on disk (check for file changes, missing files).

  - Implement validation logic to verify that catalog data matches the actual state of files on disk.
  - Report any missing files or discrepancies.

- **Implementation Details**:
  - Use **Java File I/O** to access and validate files on disk.
  - Build a service that periodically validates the catalog against the file system.

Final Task Summary by Role

(UI): Lalida Krairit

1. Catalog Interface Development
2. Curation Interface Development
3. File Comparison Viewer

(Logic): Sunidhi Pruthikosit

1. Catalog Logic
2. Curation Logic

(Backend): Niracha Janavatara

1. File Catalog
2. Catalog Comparison Module
3. Disk Comparison Module
4. File Validation Module

Implementation Notes

- **Java Swing** will be used for all GUI components, ensuring a modular MVC approach.
- **Java File I/O** and **NIO** will handle disk operations.
- **SQLite** will be used as our database and integrated with Java through JDBC
- **BufferedReader** and third-party libraries like **Apache Commons Text** will assist in file comparison tasks.

By separating the tasks between UI, logic, and backend, each person can focus on their specific area of expertise while ensuring smooth integration between components.