King Saud University
College of Computer and Information Sciences

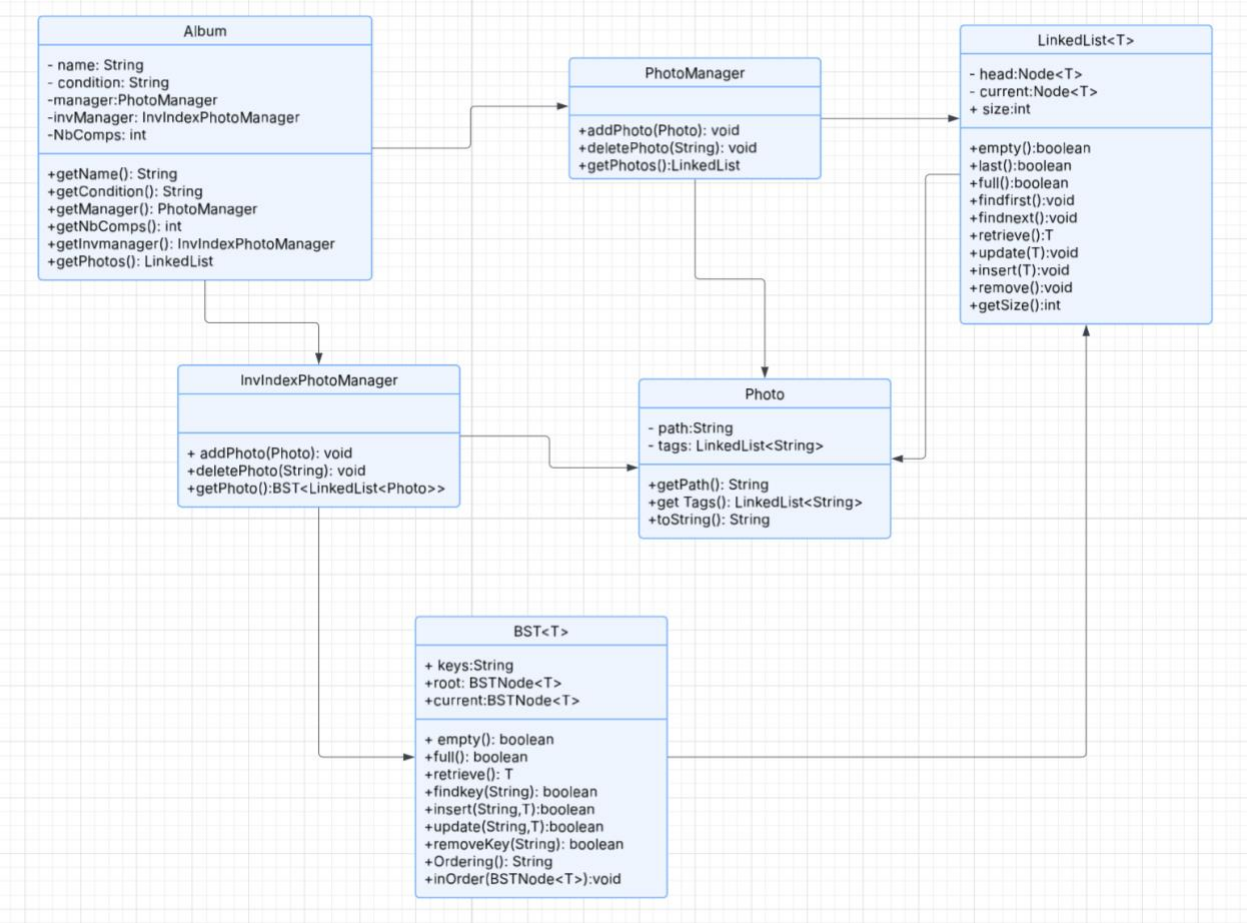
Department of Computer Science

CSC 212 Data Structures Project Report – 2nd Semester 2024-2025

Developing a Photo Management Application

Authors

Name	ID	List of all methods implemented by each student
Nameer Saad Almoqhim	**444200419**	**Photo-PhotoManager-invindexPhotoManager**
Marah basel	**444204752**	**Album-test-debug**
Rawasi Almutairi	**444200484**	**BST-LinkedList-UML DIGRAM**



1. Introduction

The project involves building a **Photo Management System** that allows users to organize their photos into albums. The photos are organized based on a set of tags provided by the user. Each photo can have one or more tags, and albums can be created based on specific tag conditions (e.g., `animal AND grass`). The project explores two different methods for searching and retrieving photos from the collection: using **LinkedLists** and using a more efficient **Inverted Index** implemented via a **Binary Search Tree (BST)**. This report will describe the specifications, design, implementation, and performance analysis of the system, as well as discuss the comparisons between the two methods of photo retrieval.

2. Specification

The project is implemented in Java, and the following classes and ADTs were used:

1. **Photo Class**
    - **Fields**:
        - `String path`: The file path of the photo (unique identifier).

        - `LinkedList<String> tags`: A linked list storing all tags associated with the photo.
    - **Methods**:
        - `String getPath()`: Returns the full file path.
        - `LinkedList<String> getTags()`: Returns a linked list of tags associated with the photo.
2. **PhotoManager Class**
    - **Fields**:
        - `LinkedList<Photo> photos`: A linked list storing all the photos.
    - **Methods**:
        - `void addPhoto(Photo p)`: Adds a new photo to the manager.
        - `void deletePhoto(String path)`: Deletes a photo based on its path.
        - `LinkedList<Photo> getPhotos()`: Returns the list of all photos.
3. **Album Class**
    - **Fields**:
        - `String name`: The name of the album.
        - `String condition`: The condition (tags) used to filter the photos in the album.
        - `PhotoManager manager`: The photo manager that stores all photos.
        - `InvIndexPhotoManager invmanager`: The inverted index manager used for efficient search (optional).
    - **Methods**:
        - `String getName()`: Returns the album name.
        - `String getCondition()`: Returns the album condition.
        - `LinkedList<Photo> getPhotos()`: Retrieves all photos that match the condition.
        - `int getNbComps()`: Returns the number of tag comparisons performed while filtering photos.
4. **InvIndexPhotoManager Class**
    - **Fields**:
        - `BST<LinkedList<Photo>> invertedIndex`: A binary search tree that stores tags as keys and lists of photos as values.
    - **Methods**:
        - `void addPhoto(Photo p)`: Adds a photo to the inverted index.
        - `void deletePhoto(String path)`: Deletes a photo from the inverted index.
        - `BST<LinkedList<Photo>> getPhotos()`: Returns the inverted index of all photos.
5. **BST Class**
    - **Fields**:
        - `BSTNode<T> root`: The root of the binary search tree.
    - **Methods**:
        - `boolean findKey(String key)`: Finds a key in the BST.
        - `boolean insert(String key, T value)`: Inserts a key-value pair into the BST.
        - `boolean removeKey(String key)`: Removes a key from the BST.

- `String inOrder()`: Returns the in-order traversal of the tree

3. Design

The system is designed to manage photos through two distinct methods of searching: **LinkedList** and **Binary Search Tree (BST)**. The **PhotoManager** handles the storage of photos in a linked list and provides basic operations like adding, deleting, and retrieving photos. The **Album** class allows users to create albums based on specific tag conditions and either use **LinkedList** or **BST** for filtering photos.

The **BST-based Inverted Index** is implemented to improve the search performance. It stores tags as keys and associates each tag with a list of photos that contain that tag. This structure allows faster retrieval of photos by leveraging the properties of the BST, specifically its logarithmic time complexity for searching.

The two approaches (LinkedList and BST) provide different performance characteristics:

- **LinkedList**: This approach sequentially scans all photos and compares their tags to the album's condition, resulting in a linear time complexity (O(n)).
- **BST**: By using an inverted index, this approach performs the search in logarithmic time (O(log n)), significantly improving the performance for large datasets.

The user can choose which method to use during runtime by selecting between the two options, offering flexibility in how photos are retrieved.

4. Implementation

The **Photo Class** was implemented to store the path and tags of a photo. A linked list was used for storing tags because it allows efficient insertion and retrieval of tag information.

The **PhotoManager Class** manages the collection of photos. It offers methods to add, delete, and retrieve photos. The deletion method also ensures that the inverted index is updated accordingly.

The **Album Class** allows users to create albums based on tag conditions. The condition can be a single tag or a combination of tags using logical AND (e.g., "animal AND grass"). The `getPhotos()` method retrieves photos that satisfy the album's condition by either using a simple **LinkedList** scan or by leveraging the **BST** if the inverted index is available.

The **InvIndexPhotoManager Class** is responsible for managing the inverted index. It builds the index by associating tags with lists of photos, enabling efficient search. The **BST Class** is used to implement the inverted index, providing fast lookups for tags and ensuring that the list of photos associated with each tag is ordered and easily retrievable.

5. Performance analysis

The performance of the Album.getPhotos() method was analyzed in terms of time complexity, both **before** and **after** using the inverted index (BST).

1. **Without Inverted Index (LinkedList)**:
    - **Time Complexity**: O(n * m)
        - Where n is the number of photos and m is the number of tags in each photo. For each photo, we compare each of its tags to the condition specified in the album.
2. **With Inverted Index (BST)**:
    - **Time Complexity**: O(log n * k)
        - Where n is the number of photos and k is the number of tags in the condition. The BST allows us to search for tags in logarithmic time, and for each tag, we retrieve the associated photos efficiently.

By using the inverted index, we significantly reduce the number of comparisons required, especially when dealing with large datasets. The performance improvement becomes more evident as the number of photos and tags increases.

6. Conclusion

In this project, a **Photo Management System** was developed, utilizing two methods for photo retrieval: **LinkedList** and **BST**. The system allows users to organize photos into albums based on tag conditions and offers a choice of search method at runtime.

- **LinkedList** provides a simple but less efficient way of searching, with a linear time complexity.
- **BST** improves performance by using an inverted index, offering logarithmic search times and reducing the number of comparisons.

The performance analysis showed a significant improvement with the **BST-based inverted index**, making it suitable for large datasets.

Future Work:

- **Scalability**: Implement additional features like bulk photo uploads and batch tag editing.
- **User Interface**: Develop a graphical user interface (GUI) for easier interaction with the system.
- **Advanced Search**: Implement more complex search queries, such as OR conditions or fuzzy tag matching.