# 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 |
|---|---|---|
| Jana alobaidi | **444200842** | **Linkedlist,photomanager ,test,report** |
| Lana alhumaidan | **444200238** | **Photo, InvIndexPhotoManager,test,report** |
| Reema albadah | **444203044** | **Bst,album,test,report** |

# 1. Introduction

This project involves developing a Photo Management Application that organizes photos into albums based on user-defined tags.

The photo manager organizes the photos into albums created by the user. An album is identified by a unique name and regroups photos that satisfy certain conditions. For the purpose of this assignment, the conditions used to create albums consist in a sequence of tags separated by ”AND”: Tag1 AND Tag2 AND Tag3

Photos that contain all specified tags will appear in the album. An empty tag list matches all photos.

# 2. Specification

**1] ADT Photo**

**Elements:**

- path: String representing the file path (unique identifier).
- tags: Collection of descriptive words (LinkedList<String>).

**Structure:**

- Each photo has a unique identifier path and associated with a list of tags.

**Domain:**

- All possible photo paths and their associated tags.

**Operations:**

***1-*** **Constructor** Photo(String path, LinkedList<String> tags)
**Requires:** path is non-null and unique; tags is non-null.
**Input:** path, tags.
**Results:** Initializes a new photo with the given path and tags.
**Output:** New Photo object.

***2-*** **Method** getPath()
**Requires:** None.
**Input:** None.
**Results:** Returns the photo’s file path.
**Output:** String path.

***3-*** **Method** getTags ()
**Requires:** None.
**Input:** None.
**Results:** Returns the list of tags associated with the photo.
**Output:** LinkedList<String> tags.

## 2] ADT PhotoManager

**Elements:**

- A collection of Photo objects.

**Structure:**

- Maintains a list of photos (LinkedList<Photo>).

**Domain:**

- All photos added with a unique identified path.

**Operations:**

***1-*** **Constructor** PhotoManager()
**Requires:** None.
**Input:** None.
**Results:** Initializes an empty photo manager.
**Output:** New PhotoManager object.

***2-*** **Method** getPhotos ()
**Requires:** None.
**Input:** None.
**Results:** Returns all managed photos.
**Output:** LinkedList<Photo> photos.

***3-*** **Method** addPhoto(Photo p)
**Requires:** p is non-null and has a unique path.
**Input:** Photo p.
**Results:** Adds p to the manager.
**Output:** None.

***4-*** **Method** deletePhoto(String path)
**Requires:** A photo with path exists in the manager.
**Input:** String path.
**Results:** Removes the photo with the specified path.
**Output:** None.

## 3] ADT Album

**Elements:**

- name: String identifier for the album.
- condition: String of tags separated by "AND" (e.g., "animal AND grass").
- manager: Reference to a PhotoManager.
- `invmanager`: Reference to InvIndexPhotoManager.
-

**Structure:**

- Dynamically evaluates photos matching condition when getPhotos() is called.

**Domain:**

- All possible combinations of album names, conditions, and photo sets.

**Operations:**

**Constructor:** Album(String name, String condition, PhotoManager manager, InvIndexPhotoManager invmanager)

**Requires:** name and condition are non-null; manager is initialized.
**Input:** name, condition, manager.
**Results:** Initializes an album with the given parameters.
**Output:** New Album object.

***1-*** **Method** getName ()

**Requires:** None.
**Input:** None.
**Results:** Returns the album's name.
**Output:** String name.

***2-*** **Method** getCondition ()

**Requires:** None.
**Input:** None.
**Results:** Returns the album's condition.
**Output:** String condition.

***3-*** **Method** getManager ()

**Requires:** None.
**Input:** None.
**Results:** Returns the album's photo manager.
**Output:** PhotoManager manager.

***4-*** **Method** getPhotos ()

**Requires:** None.
**Input:** None.
**Results:** Returns all photos in the manager that match condition.
**Output:** LinkedList<Photo> list.

***5-*** **Method** getNbComps ()

**Requires:** None.
**Input:** None.
**Results:** Returns the number of tag comparisons made during getPhotos().
**Output:** int count.

**4] ADT InvIndexPhotoManager**

**Elements:**

- photos: A collection of Photo objects.
- index: BST where each node maps a tag (String) to a LinkedList<Photo>.

**Structure:**

- use BST for O(log n) tag searches.

**Domain:**

- All tags and their associated photos in the manager.

**Operations:**

***1-*** **Constructor** InvIndexPhotoManager ()
**Requires:** None.
**Input:** None.
**Results:** Initializes an empty photo manager and an empty inverted index (BST).
**Output:** New InvIndexPhotoManager object.

***2-*** **Method** addPhoto(Photo p)
**Requires:** p is non-null and has a unique path.
**Input:** Photo p.
**Results:** Adds p to the PhotoManage and For each tag in p.getTags(), updates the BST: inserts the tag (if new) or appends p to the existing tag's list.
**Output:** None.

***3-*** **Method** deletePhoto(String path)
**Requires:** A photo with path exists.
**Input:** String path.
**Results:** Removes the photo from the PhotoManager And For each tag in the photo's tags, removes path from the BST; deletes the tag node if its photo list becomes empty.
**Output:** None.

***4-*** **Method** getPhotos ()
**Requires:** None.
**Input:** None.
**Results:** Returns the inverted index (BST).
**Output:** BST<LinkedList<Photo>> index.

# 3. Design

This project involves developing a Photo Management Application that organizes photos into albums based on defined tags. The application allows to:

- Assign tags to photos (e.g., "animal," "grass").
- Create albums with conditions like "animal AND grass".
- Efficiently retrieve photos matching these conditions using an inverted index for optimization.

**System Workflow**

***1-*** Add Photo:

- adds a photo with tags (e.g., "animal, grass").
- The PhotoManager stores the photo and updates the inverted index.

***2-*** Album Creation:

- Creates an album with a condition (e.g., "animal AND grass").
- The Album class uses the inverted index to retrieve matching photos.

- **Photo Retrieval:**
- During runtime, the system prompts the user to select the search method:
  - 1: Linked List (Linear Search)
  - 2: BST (Optimized Search)

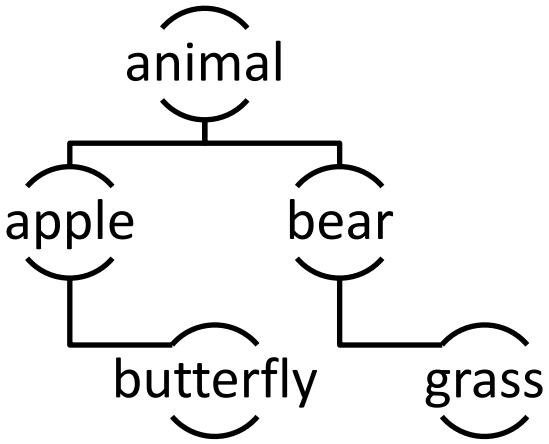
  - Photos matching all tags are retrieved based on the chosen method.

- **Photo Deletion:**
  - Deletes photos and updates both PhotoManager and the inverted index.

**Data Structures**

| Component | Data Structure | Purpose |
|---|---|---|
| **Photo Storage** | LinkedList<Photo> | Store all photos in the manager. |
| **Inverted Index** | BST<String, LinkedList<Photo>> | Map tags to photos for O(log n) search. |

**Inverted Index (BST) Diagram**



# 4. Implementation

**Key Features**

**1. Inverted Index Construction**

- BST Nodes: Each node stores a tag and a LinkedList<Photo> of photos with that tag.
- Insertion/Deletion:
  - When a photo is added, its tags are inserted into the BST (or appended to existing lists).
  - When a photo is deleted, its entries are removed from the BST; empty tag lists are deleted.

**2. Album Photo Retrieval**

```
public LinkedList<Photo> getPhotos() {
		if (condition == null || condition.trim().equals(""))
			return manager.getPhotos();
		else {
			LinkedList<Photo> list = new LinkedList<Photo>();
			LinkedList<Photo> photos = manager.getPhotos();
			String conditions[] = condition.trim().split("AND");
			photos.findFirst();
			while (!photos.last()) {
				Photo photo = photos.retrieve();
				boolean allConditions = true;
				LinkedList<String> tags = photo.getTags();
				for (String condition : conditions) {
					if (tags.find(condition.trim()) == null) {
						allConditions = false;
						break;
					}
				}
				if (allConditions) {
					if (list.find(photo) == null)
						list.insert(photo);
				}
				photos.findNext();
			}
			return list;
		}
	}
```

**Important Note:**
The getPhotos() method shown in the implementation section represents the version used in **Phase 1**, where the system searches through all photos and tags linearly using the PhotoManager and LinkedList. However, in **Phase 2**, this method is optimized by retrieving photos directly from the **Inverted Index** (BST) via the InvIndexPhotoManager, where each tag maps to a list of photos, and matching photos are found by intersecting those lists.

# 5. Performance analysis

| Operation | Before Inverted Index (Linear) | After Inverted Index (BST) |
|---|---|---|
| **getPhotos()** | O(n × m) | O(k × log n) |
| **addPhoto()** | O(1) | O(m × log n) |
| **deletePhoto()** | O(n) | O(m × log n) |

**Variables:**

- n: Total photos.
- m: Tags per photo.
- k: Tags in the album condition.

# 6. Conclusion

The photo management application successfully organizes photos into albums using tag-based conditions. The implementation of an inverted index with a BST improved photos retrieval efficiency, by reducing the time complexity from O(n * m) to O(k log n). Future enhancements could include support for more complex conditions (e.g., "OR" operations) and a graphical user interface (GUI).