Project : P6

Group members

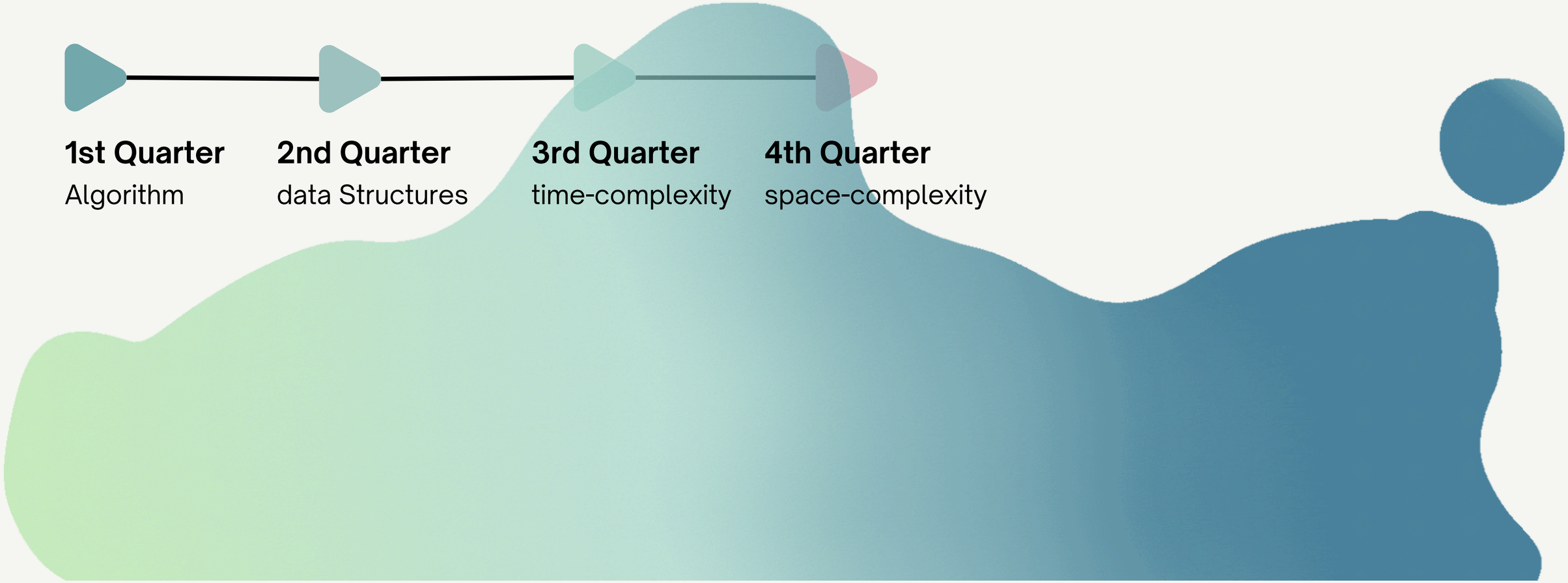
Ayush Mittal	:	202301145
Sri Sadana Dharavath	:	202301121
Zeel Parmar	:	202301171
Sanya Vaishnavi	:	202301117

Problem Statement:

You need to build a manager for all the DA-IICT clubs. The manager ensures that a club member can be looked up in minimum time. A member can either be a faculty or a student. One should be able to search by name, ID, specific club name, or club category (i.e., arts, science & technology, sports, culture). Note that the user of this manager may not be a DA-IICT-ian and, therefore, may not know the clubs' names.

Timeline

Briefly discuss the key dates for the project.



1st Quarter
Algorithm

2nd Quarter
data Structures

3rd Quarter
time-complexity

4th Quarter
space-complexity

Algorithm:

1. Define Club and Member Structures :

- Create club structures with details about the club's name, category, and members (students and faculty members).
- Create structures for members containing information such as name, ID, and club memberships.

2. Initialize Club Database :

- Create a database containing details on every DA-IICT club, such as names, groups, and membership.

3. Search Member Function (searchMember) :

- Searches for a member by name or ID using membersByName and membersById.
- Utilizes lowercase conversion for case-insensitive search.
- Time Complexity: O(1) on average for ID search, O(N) for name search where N is the number of members.
- Space Complexity: O(1)
- Display all clubs in that category along with their details.

Algorithm:

4. Handling Non-DA-IICT Users :

- If the user is not familiar with DA-IICT clubs' names.
- Provide a list of club categories as options for the user to choose from.
- Once a category is selected, display all clubs in that category for further selection.
- Finally, display the details of the selected club or its members.

5. Optimize Search Time :

- Implement data structures like hash tables or binary search trees to optimize search time for faster lookup of members or clubs.
- Time Complexity: O(N*M), where N is the number of members and M is the average number of clubs per member.
- Space Complexity: O(N+M), where N is the number of members and M is the number of unique clubs.
- Provide appropriate error messages to guide the user.

Algorithm:

7. Main Function :

- Handles user interaction and input processing.
- Executes search functions based on user choices.
- Time Complexity: Depends on the chosen operation.
- Space Complexity: Depends on the data retrieved from the search functions.

8. Time complexity part by part

-Reading the CSV file and parsing each line: O(n · m)

-Searching for a member (searchMember function): O(1).

-Searching for a club by name (searchClub function): O(K)

- Searching for clubs by category (searchClubByCategory function): O(1)

□ loading data: O(n.m)

□ search member (by name): O(n)

□ search club (by name): O(k)

□ search club by category: O(1)(category lookup)+ O(m) per club

Data Structures used:

1. Unordered Maps
2. Vectors
3. Enums
4. Struct

1 . Unordered map :

Use: For storing relationships between values (Member or Club objects) and keys (such as member names, IDs, and club names).

Why use this: With an average time complexity of O(1) for insertion, deletion, and search operations, unordered maps offer quick lookup speeds.

Space Complexity: An unordered map has a space complexity of O(n), where n is the total number of elements that the map stores. This is due to the fact that the map's space requirements increase linearly with the quantity of key-value pairs it stores.

Justification: Because unordered maps use constant-time lookup operations, they are appropriate for efficient searching because they enable speedy retrieval of members and clubs by their names and IDs.

2. Vectors :

Use: Lists of clubs within each category are stored for the purpose of enabling category-specific searches.

Why use this: Vectors are useful in situations where sequential access to items is necessary because they provide constant-time access to elements by index.

Space Complexity: A vector has a space complexity of O(n), where n is the number of components it contains. This is a result of vectors' dynamic memory allocation and resizing.

Justification: When looking for clubs inside particular categories, efficient iteration and access are made possible by storing clubs in vectors. Because there can be variations in the number of clubs in each category, vectors' dynamic scaling guarantees the best possible memory utilization.

3. Enums :

Useful for defining separate club types to improve code readability and organization.

Why use this: By mandating a limited number of values, enums improve code readability by offering a means to describe fixed sets of named constants.

Space Complexity: Enums usually don’t significantly increase the program’s space complexity. Typically, numerical values are used to represent them.

Justification: The code is less prone to errors and more self-explanatory when handling club categories thanks to the use of enums. Enumerations lessen the possibility of using wrong category names and aid in ensuring type safety.

4. Struct :

Use: To create unique data structures for clubs and members that combine related data into a single entity.

Why use this : Structures link relevant attributes together to make managing and manipulating member and club information easier.

Space Complexity: A structure’s space complexity is influenced by the magnitude of each of its constituent variables. In this case, the size of the strings (name) and integer variables (ID) inside the structures dictates the space complexity.

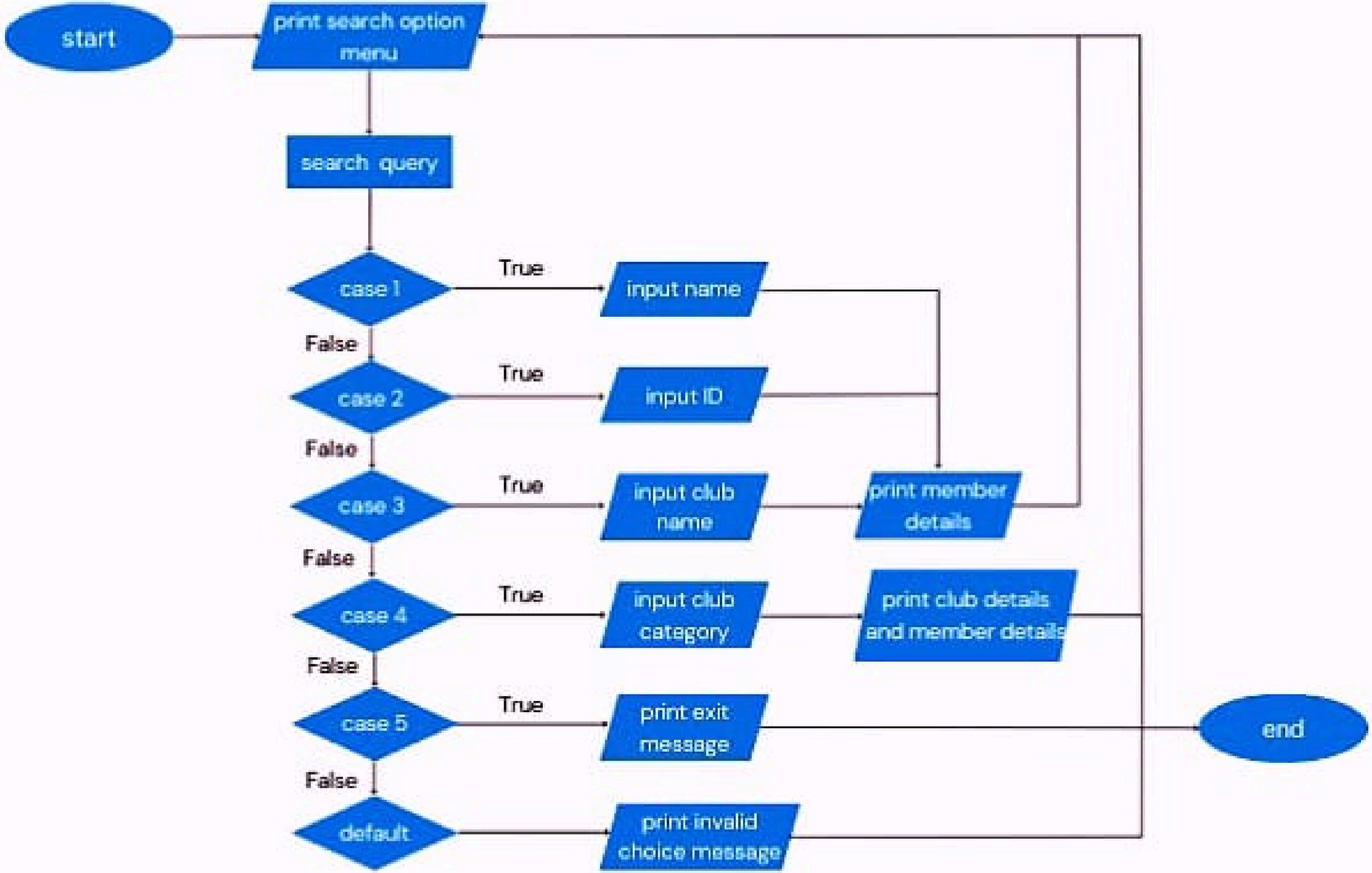
Justification: The code becomes more modular and organized by establishing structures for clubs and members. It also improves maintainability and readability of the code by logically grouping attributes related to clubs and members.

time-complexity

	time-complexity
reading the csv file	O(n.m)
search for a member by name	O(N)
search by club name	O(K)
search by category	O(1)

space-complexity

	space-complexity
the code	O(N+M)
search for a member by name	O(N)
search by club name	O(K)
search by category	O(1)


start
print search option menu
search query
case 1
True
input name
False
case 2
True
input ID
False
case 3
True
input club name
print member details
False
case 4
True
input club category
print club details and member details
False
case 5
True
print exit message
end
False
default
print invalid choice message

```cpp
#include <iostream>
#include <cstdio>
#include <fstream>
#include <sstream>
#include <vector>
#include <unordered_map>
#include <algorithm>
#include <string>

using namespace std;

// Define club categories
enum class ClubCategory {
    Arts,
    Science,
    Sports,
    Culture
};

// Define a structure for a club
struct Club {
    string name;
    ClubCategory category;

    Club(const string& n, ClubCategory c) {
        this->name = n;
        this->category = c;
    }
};

string toLower(string str) {
    transform(str.begin(), str.end(), str.begin(), ::tolower);
    return str;
}

// Define a structure for a member
struct Member {
    string name;
    int id;
    vector<Club*> clubs;
```

Thank You