CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

# UNIVERSITY INSTITUTE OF COMPUTING

# PROJECT REPORT
# ON
# Social Media Management System

Program Name: BCA

Subject Name/Code: Data Structures(23CAT-201)


**Submitted by:**

Name: Yashaswi Prashar

UID: **23BCA10575**

Section: BCA – **10** "A"

**Submitted to:**

Name: Miss Preeti

Designation: Ass. Professor

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

# ABSTRACT

## Introduction:

The Social Media Management System is a C++ application designed to simulate core social media functionalities, such as creating user profiles, posting content, liking posts, adding comments, and following users. It leverages data structures including arrays, binary search trees, and structs to efficiently manage and store user and post data, while providing quick access to these resources.

Users can register their profiles, create posts, and interact with each other by following users, liking posts, and adding comments. A menu-driven interface allows users to navigate through these options in a straightforward and interactive manner.

## Technique:

This application is built using C++ with an emphasis on procedural programming principles and data structure utilization to optimize data storage and retrieval. The following data structures are used:

1. Binary Search Tree (BST):
    - Purpose: To store and manage user profiles.
    - Benefits: The BST allows for efficient searching, insertion, and deletion of profiles based on unique userID. This structure is particularly beneficial as the number of users grows, as it ensures quick access to profiles.
    - Implementation: Each Profile is a node in the BST, containing user information like userID, username, email, and follower/following counts.
2. Array of Structures:
    - Purpose: To store posts and comments.
    - Benefits: Using an array provides constant-time access to each post and enables straightforward management of posts and comments up to a defined limit.

- Implementation: Posts are stored in a fixed-size array with each Post containing a postID, userID, content, likes count, and a sub-array for comments with a limit of 10 comments per post.

3. Structs:
    - Purpose: To encapsulate data for profiles and posts.
    - Benefits: Structs provide an organized way to store related data fields in single units, which helps in modularity and readability of the code.
    - Implementation: The Profile struct holds user details, while the Post struct holds post information and an array for comments.
4. Error Handling and Input Validation:
    - Each menu input is validated to avoid errors and guide the user to correct inputs, enhancing overall reliability and user experience.

# System Configuration:

- OS: Windows 10 or Linux
- Processor: Intel Core i3 (minimum); Core i5 or higher recommended
- RAM: 4 GB (minimum); 8 GB recommended
- Development Environment: Any C++ IDE (e.g., Visual Studio, Code::Blocks) or Visual Studio Code with a C++ compiler (GCC or Microsoft C++ Compiler)

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

# SUMMARY

## Input:

Users interact with a main menu, selecting from various options:

1. Create Profile: Allows new users to register by entering a unique userID, username, email, and password.
2. Display Profile: Displays an existing user's profile based on userID.
3. Create Post: Allows users to create a new post by entering userID, a unique postID, and post content.
4. Like Post: Enables users to like a post by specifying its postID.
5. Add Comment to Post: Allows users to add comments on a post by entering the postID and comment text.
6. Display User Posts: Lists all posts by a user, including comments and likes count.
7. Exit: Ends the session.

# CODE:

```
#include <iostream>
#include <string>
#include<algorithm>
using namespace std;
const int MAX_COMMENTS = 10; // Maximum comments per post

// Define a structure for User Profile in a Binary Search Tree (BST) node
struct Profile {
    int userID;
    string username;
    string email;
    string password;
    int followersCount;
    int followingCount;
    Profile* left;
    Profile* right;

    Profile(int id, string user, string mail, string pass) :
        userID(id), username(user), email(mail), password(pass),
        followersCount(0), followingCount(0), left(nullptr), right(nullptr) {}
};

// Define a structure for Posts
struct Post {
```

```
    int postID;
    int userID; // ID of the user who created the post
    string content;
    int likesCount;
    string comments[MAX_COMMENTS]; // Array for comments
    int commentCount;
};

// Root of the Profile BST
Profile* root = nullptr;

// Array for posts
Post posts[100];
int postCount = 0;

// Function to insert a profile in BST
Profile* insertProfile(Profile* root, Profile* newProfile) {
    if (!root) return newProfile;
    if (newProfile->userID < root->userID)
        root->left = insertProfile(root->left, newProfile);
    else if (newProfile->userID > root->userID)
        root->right = insertProfile(root->right, newProfile);
    return root;
}

// Function to search for a profile by userID
```

```
Profile* searchProfile(Profile* root, int userID) {
    if (!root || root->userID == userID) return root;
    if (userID < root->userID) return searchProfile(root->left, userID);
    return searchProfile(root->right, userID);
}

// Function to create a user profile
void createProfile() {
    int userID;
    string username, email, password;
    cout << "Enter a unique User ID: ";
    cin >> userID;
    cin.ignore();

    if (searchProfile(root, userID)) {
        cout << "Error: User ID already exists. Please try again." << endl;
        return;
    }

    cout << "Enter Username: ";
    getline(cin, username);
    cout << "Enter Email: ";
    getline(cin, email);
    cout << "Enter Password: ";
    getline(cin, password);
```

```
    Profile* newProfile = new Profile(userID, username, email, password);
    root = insertProfile(root, newProfile);
    cout << "Profile created for user: " << username << endl;
}

// Function to display profile details
void displayProfile() {
    int userID;
    cout << "Enter User ID to view profile: ";
    cin >> userID;

    Profile* profile = searchProfile(root, userID);
    if (profile) {
        cout << "User ID: " << profile->userID << endl;
        cout << "Username: " << profile->username << endl;
        cout << "Email: " << profile->email << endl;
        cout << "Followers: " << profile->followersCount << endl;
        cout << "Following: " << profile->followingCount << endl;
    } else {
        cout << "Profile not found." << endl;
    }
}

// Function to create a post
void createPost() {
    if (postCount >= 100) {
```

```
    cout << "Maximum post limit reached." << endl;
    return;
}

int userID, postID;
string content;
cout << "Enter User ID to create a post: ";
cin >> userID;
cin.ignore();

if (!searchProfile(root, userID)) {
    cout << "Error: User ID not found. Please create a profile first." << endl;
    return;
}

cout << "Enter a unique Post ID: ";
cin >> postID;
cin.ignore();

for (int i = 0; i < postCount; i++) {
    if (posts[i].postID == postID) {
        cout << "Error: Post ID already exists. Please try again." << endl;
        return;
    }
}
```

```
    cout << "Enter Post Content: ";
    getline(cin, content);

    posts[postCount++] = { postID, userID, content, 0, {}, 0 };
    cout << "Post created by user " << userID << ": " << content << endl;
}

// Function to like a post
void likePost() {
    int postID;
    cout << "Enter Post ID to like: ";
    cin >> postID;
    for (int i = 0; i < postCount; i++) {
        if (posts[i].postID == postID) {
            posts[i].likesCount++;
            cout << "Post " << postID << " now has " << posts[i].likesCount << " likes." << endl;
            return;
        }
    }
    cout << "Post not found." << endl;
}

// Function to add a comment to a post
void addComment() {
    int postID;
    string comment;
```

```
    cout << "Enter Post ID to comment on: ";
    cin >> postID;
    cin.ignore();

    for (int i = 0; i < postCount; i++) {
        if (posts[i].postID == postID) {
            if (posts[i].commentCount < MAX_COMMENTS) {
                cout << "Enter Comment: ";
                getline(cin, comment);
                posts[i].comments[posts[i].commentCount++] = comment;
                cout << "Comment added to post " << postID << ": " << comment << endl;
            } else {
                cout << "Maximum comment limit reached for this post." << endl;
            }
            return;
        }
    }
    cout << "Post not found." << endl;
}

// Function to display all posts by a user
void displayUserPosts() {
    int userID;
    cout << "Enter User ID to view posts: ";
    cin >> userID;
```

```
    cout << "Posts by User " << userID << ":" << endl;
    bool found = false;
    for (int i = 0; i < postCount; i++) {
        if (posts[i].userID == userID) {
            found = true;
            cout << "Post ID: " << posts[i].postID << endl;
            cout << "Content: " << posts[i].content << endl;
            cout << "Likes: " << posts[i].likesCount << endl;
            cout << "Comments:" << endl;
            for (int j = 0; j < posts[i].commentCount; j++) {
                cout << "- " << posts[i].comments[j] << endl;
            }
            cout << "-----------------------------" << endl;
        }
    }
    if (!found) cout << "No posts found for this user." << endl;
}

int main() {
    int choice;
    do {
        cout << "\n----- Social Media Management System -----" << endl;
        cout << "1. Create Profile" << endl;
        cout << "2. Display Profile" << endl;
        cout << "3. Create Post" << endl;
        cout << "4. Like Post" << endl;
```

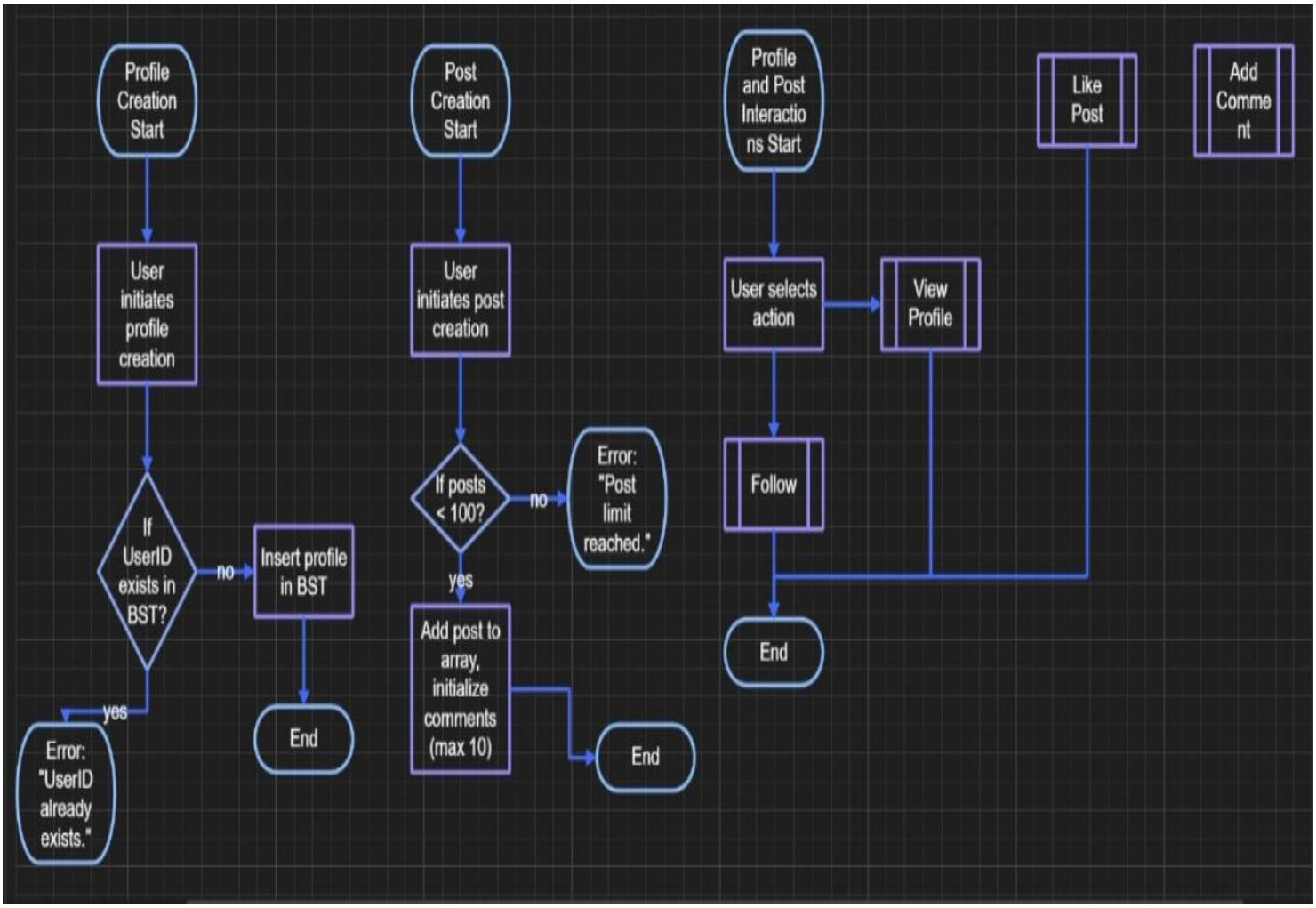
```
        cout << "5. Add Comment to Post" << endl;
        cout << "6. Display User Posts" << endl;
        cout << "7. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        if (cin.fail()) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Invalid choice. Please enter a number from 1 to 7." << endl;
            continue;
        }

        switch (choice) {
            case 1: createProfile(); break;
            case 2: displayProfile(); break;
            case 3: createPost(); break;
            case 4: likePost(); break;
            case 5: addComment(); break;
            case 6: displayUserPosts(); break;
            case 7: cout << "Exiting..." << endl; break;
            default: cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 7);
    return 0;
}
```

# Process:

Here is the flowchart of one some of the features:



# Output:

## Main Menu:

```
----- Social Media Management System -----
1. Create Profile
2. Display Profile
3. Create Post
4. Like Post
5. Add Comment to Post
6. Follow User
7. Display User Posts
8. Exit
```

## Creating New ID:

```
----- Social Media Management System -----
1. Create Profile
2. Display Profile
3. Create Post
4. Like Post
5. Add Comment to Post
6. Follow User
7. Display User Posts
8. Exit
Enter your choice: 1
Enter User ID: 123
Enter Username: Yashi
Enter Email: Yashi2004@gmail
Enter Password: 1234
Profile created for user: Yashi
```

## Follow User:

```
----- Social Media Management System -----
1. Create Profile
2. Display Profile
3. Create Post
4. Like Post
5. Add Comment to Post
6. Follow User
7. Display User Posts
8. Exit
Enter your choice: 6
Enter User ID to follow: 123
Enter Your User ID: 1234
User 1234 followed User 123
```

## Display Profile:

```
----- Social Media Management System -----
1. Create Profile
2. Display Profile
3. Create Post
4. Like Post
5. Add Comment to Post
6. Follow User
7. Display User Posts
8. Exit
Enter your choice: 2
Enter User ID to view profile: 1234
User ID: 1234
Username: Harshit
Email: Harshit2004
Followers: 0
Following: 1
```

## Create a Post:

```
----- Social Media Management System -----
1. Create Profile
2. Display Profile
3. Create Post
4. Like Post
5. Add Comment to Post
6. Follow User
7. Display User Posts
8. Exit
Enter your choice: 3
Enter User ID to create a post: 1234
Enter Post ID: 1
Enter Post Content: Hello Guys Chai Peelo
Post created by user 1234: Hello Guys Chai Peelo
```

# CONCLUSION:

The Social Media Management System is an interactive, efficient application designed to simulate social media functionalities using C++ and essential data structures. The incorporation of a Binary Search Tree for managing user profiles enables quick insertion, searching, and deletion of users. An array structure for posts and comments ensures easy access to each post's details, including likes and comments.

The project demonstrates effective use of C++ data structures for practical applications and provides a scalable framework for implementing additional social media features, such as direct messaging or multimedia support, in future development.