



NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCE

Mini Instagram

GROUP MEMBERS

Name	Roll No
Talha Abdullah Bangyal	22F- 3194
Mian Muhammad Ammar	22F- 3102

Table of Contents

Codes.....	3
------------	---

menu.h.....	3
menu.cpp	3
graph.h.....	4
graph.cpp	4
credential.h.....	6
credential.cpp	7
data.h	7
data.cpp	9
main.cpp.....	11
Screenshot	14
Signup	14
For Talha	14
For Abdullah	15
Login	15
For Talha	15
For Abdullah	16
Logout	16
For Talha	16
For Abdullah	17
Send Friend Request	17
For Talha	17
For Abdullah	18
View Notifications	18
For Talha	18
For Abdullah	19
Post Content	19
For Talha	19
For Abdullah	20
View Posts	20
For Talha	20
For Abdullah	21
Send Message.....	21
For Talha	21

For Abdullah	22
View Messages.....	22
For Talha	22
For Abdullah	23
View Friends	23
For Talha	23
For Abdullah	24
View Friend Request	24
For Talha	24
For Abdullah	25
Accept Friend Request	25
For Talha	25
For Abdullah	26
Reject Friend Request	26
For Talha	26
For Abdullah	27
Exit.....	27

Codes

menu.h

```
#ifndef MENU_H
#define MENU_H
#pragma once
#include <iostream>
#include <string>
#include <ctime>
using namespace std;

void displayMenu();
string getCurrentTimestamp();
#endif
```

menu.cpp

```
#include "menu.h"

// Main Menu
void displayMenu() {
    cout << "--- Mini Instagram ---" << endl;
    cout << "1. Signup" << endl;
    cout << "2. Login" << endl;
    cout << "3. Logout" << endl;
    cout << "4. Send Friend Request" << endl;
}
```

```

    cout << "5. View Notifications" << endl;
    cout << "6. Post Content" << endl;
    cout << "7. View Posts" << endl;
    cout << "8. Send Message" << endl;
    cout << "9. View Messages" << endl;
    cout << "10. View Friends" << endl;
    cout << "11. View Friend Request" << endl;
    cout << "12. Accept Friend Request" << endl;
    cout << "13. Reject Friend Request" << endl;
    cout << "0. Exit" << endl;
    cout << "Enter your choice: ";
}

// Helper Function to Get Current Timestamp
string getCurrentTimestamp() {
    time_t now = time(0); // Get current time
    char buffer[26];      // Buffer to hold the formatted time
    ctime_s(buffer, sizeof(buffer), &now); // Safe version of ctime
    return string(buffer); // Return as a string
}

```

graph.h

```

#ifndef GRAPH_H
#define GRAPH_H
#pragma once
#include "menu.h"
#include "data.h"

class UserGraph {
    UserFriend* head;
    UserFriend* findUser(string username) {
        UserFriend* temp = head;
        while (temp && temp->username != username) {
            temp = temp->next;
        }
        return temp;
    }
public:
    UserGraph();
    void addUser(string username);
    void addFriendRequest(string sender, string receiver);
    void displayFriendRequests(string username);
    void acceptFriendRequest(string username, string sender);
    void rejectFriendRequest(string username, string sender);
    void displayNotifications(string username);
    void displayFriends(string username);
    void postContent(string username, string content);
    void viewPosts(string username);
    void sendMessage(string sender, string receiver, string content);
    void viewMessages(string username);
};
#endif

```

graph.cpp

```

#include "graph.h"

UserGraph::UserGraph() {
    head = nullptr;
}

void UserGraph::addUser(string username) {

```

```

        if (!findUser(username)) {
            UserFriend* newUser = new UserFriend(username);
            newUser->next = head;
            head = newUser;
        }
    }

    // Function to add a friend request
    void UserGraph::addFriendRequest(string sender, string receiver) {
        UserFriend* user = findUser(receiver);
        if (user) {
            user->friendRequests.enqueue(sender + " sent you a friend request.");
        }
    }

    // Function to display incoming friend requests for a user
    void UserGraph::displayFriendRequests(string username) {
        UserFriend* user = findUser(username);
        if (user) {
            cout << "Friend Requests for " << username << ":" << endl;
            user->friendRequests.display();
            cout << "Do you want to accept or reject any request? (Enter 'accept' or 'reject') " << endl;
        }
    }

    // Function to accept a friend request
    void UserGraph::acceptFriendRequest(string username, string sender) {
        UserFriend* user = findUser(username);
        if (user) {
            // Add the sender to the user's friend list
            FriendNode* newFriend = new FriendNode(sender);
            newFriend->next = user->friendsHead;
            user->friendsHead = newFriend;

            // Remove the notification
            string notificationMessage = sender + " sent you a friend request.";
            if (user->friendRequests.removeMessage(notificationMessage)) {
                cout << "Friend request accepted!" << endl;
            }
            else {
                cout << "Friend request not found." << endl;
            }
        }
    }

    // Function to reject a friend request
    void UserGraph::rejectFriendRequest(string username, string sender) {
        UserFriend* user = findUser(username);
        if (user) {
            // Remove the notification
            string notificationMessage = sender + " sent you a friend request.";
            if (user->friendRequests.removeMessage(notificationMessage)) {
                cout << "Friend request rejected!" << endl;
            }
            else {
                cout << "Friend request not found." << endl;
            }
        }
    }

    void UserGraph::displayNotifications(string username) {

```

```

    UserFriend* user = findUser(username);
    if (user) {
        user->notifications.display();
    }
}

void UserGraph::displayFriends(string username) {
    UserFriend* user = findUser(username);
    if (user) {
        FriendNode* temp = user->friendsHead;
        cout << "Friends of " << username << ": ";
        while (temp) {
            cout << temp->username << " ";
            temp = temp->next;
        }
        cout << endl;
    }
}

void UserGraph::postContent(string username, string content) {
    UserFriend* user = findUser(username);
    if (user) {
        user->posts.push(content);
        user->notifications.enqueue("New post added: " + content);
    }
}

void UserGraph::viewPosts(string username) {
    UserFriend* user = findUser(username);
    if (user) {
        user->posts.display();
    }
}

void UserGraph::sendMessage(string sender, string receiver, string content) {
    UserFriend* user = findUser(receiver);
    if (user) {
        user->messages.push(sender, content);
        user->notifications.enqueue(sender + " sent you a message: " + content);
    }
}

void UserGraph::viewMessages(string username) {
    UserFriend* user = findUser(username);
    if (user) {
        user->messages.display();
    }
}

```

credential.h

```

#ifndef CREDENTIAL_H
#define CREDENTIAL_H
#pragma once
#include "menu.h"
#include "data.h"

class CredentialManager {
    CredentialNode* head;

public:
    CredentialManager();

```

```

    void addUser(string username, string password);
    bool authenticate(string username, string password);
    bool findUser(string username);
};
#endif

```

credential.cpp

```

#include "credential.h"

CredentialManager::CredentialManager() {
    head = nullptr;
}

void CredentialManager::addUser(string username, string password) {
    if (!findUser(username)) {
        CredentialNode* newNode = new CredentialNode(username, password);
        newNode->next = head;
        head = newNode;
    }
}

bool CredentialManager::authenticate(string username, string password) {
    CredentialNode* temp = head;
    while (temp) {
        if (temp->username == username && temp->password == password)
            return true;
        temp = temp->next;
    }
    return false;
}

bool CredentialManager::findUser(string username) {
    CredentialNode* temp = head;
    while (temp) {
        if (temp->username == username)
            return true;
        temp = temp->next;
    }
    return false;
}

```

data.h

```

#ifndef DATA_H
#define DATA_H
#pragma once
#include "data.h"
#include "menu.h"

// Linked List for User Credentials
struct CredentialNode {
    string username;
    string password;
    CredentialNode* next;
    CredentialNode(string uname, string pwd);
};

// Linked List Node for Posts
struct Post {
    string timestamp;
    string content;
};

```

```
    Post* next;
    Post(string content);
};

// Stack for Posts
struct PostStack {
    Post* top;
    PostStack();
    void push(string content);
    void display();
};

// Queue for Friend Requests or Notifications
struct QueueNode {
    string message;
    QueueNode* next;
    QueueNode(string msg);
};

struct Queue {
    QueueNode* front;
    QueueNode* rear;
    Queue();
    void enqueue(string msg);
    string dequeue();
    bool isEmpty();
    void display();
    bool removeMessage(const string& msg);
};

// Stack for Messages
struct MessageNode {
    string sender;
    string content;
    MessageNode* next;
    MessageNode(string sndr, string msg);
};

struct MessageStack {
    MessageNode* top;
    MessageStack();
    void push(string sender, string content);
    void display();
};

// Graph for User Relationships
struct FriendNode {
    string username;
    FriendNode* next;
    FriendNode(string uname);
};

struct UserFriend {
    string username;
    FriendNode* friendsHead;
    UserFriend* next;
    Queue friendRequests;
    Queue notifications;
    PostStack posts;
    MessageStack messages;

    UserFriend(string uname);
```



```
};  
#endif
```

data.cpp

```
#include "menu.h"  
#include "data.h"  
  
// Linked List for User Credentials  
CredentialNode::CredentialNode(string uname, string pwd) {  
    username = uname;  
    password = pwd;  
    next = nullptr;  
}  
  
// Linked List Node for Posts  
Post::Post(string content) {  
    timestamp = getCurrentTimestamp();  
    this->content = content;  
    next = nullptr;  
}  
  
// Stack for Posts  
PostStack::PostStack() {  
    top = nullptr;  
}  
void PostStack::push(string content) {  
    Post* newPost = new Post(content);  
    newPost->next = top;  
    top = newPost;  
}  
  
void PostStack::display() {  
    Post* temp = top;  
    if (!temp) {  
        cout << "No posts available." << endl;  
        return;  
    }  
    while (temp) {  
        cout << temp->timestamp << ": " << temp->content << endl;  
        temp = temp->next;  
    }  
}  
  
// Queue for Friend Requests or Notifications  
QueueNode::QueueNode(string msg) {  
    message = msg;  
    next = nullptr;  
}  
  
Queue::Queue() {  
    front = nullptr;  
    rear = nullptr;  
}  
  
void Queue::enqueue(string msg) {  
    QueueNode* newNode = new QueueNode(msg);  
    if (rear) {  
        rear->next = newNode;  
    }  
    else {  
        front = newNode;  
    }  
}
```

```

    }
    rear = newNode;
}

string Queue::dequeue() {
    if (!front) return "Queue is empty.";
    QueueNode* temp = front;
    string msg = front->message;
    front = front->next;
    if (!front) rear = nullptr;
    delete temp;
    return msg;
}

bool Queue::isEmpty() {
    return front == nullptr;
}

void Queue::display() {
    QueueNode* temp = front;
    if (!temp) {
        cout << "Queue is empty." << endl;
        return;
    }
    while (temp) {
        cout << temp->message << endl;
        temp = temp->next;
    }
}

bool Queue::removeMessage(const string& msg) {
    if (!front) return false; // Queue is empty

    // If the message is at the front
    if (front->message == msg) {
        QueueNode* temp = front;
        front = front->next;
        if (!front) rear = nullptr;
        delete temp;
        return true;
    }

    // Search for the message in the rest of the queue
    QueueNode* temp = front;
    while (temp->next) {
        if (temp->next->message == msg) {
            QueueNode* toDelete = temp->next;
            temp->next = temp->next->next;
            if (temp->next == nullptr) rear = temp; // Update rear if
necessary
            delete toDelete;
            return true;
        }
        temp = temp->next;
    }

    return false; // Message not found
}

// Stack for Messages
MessageNode::MessageNode(string sndr, string msg) {
    sender = sndr;
    content = msg;
}

```

```

        next = nullptr;
    }

    MessageStack::MessageStack() {
        top = nullptr;
    }

    void MessageStack::push(string sender, string content) {
        MessageNode* newNode = new MessageNode(sender, content);
        newNode->next = top;
        top = newNode;
    }

    void MessageStack::display() {
        MessageNode* temp = top;
        if (!temp) {
            cout << "No messages available." << endl;
            return;
        }
        while (temp) {
            cout << temp->sender << ": " << temp->content << endl;
            temp = temp->next;
        }
    }

    // Graph for User Relationships
    FriendNode::FriendNode(string uname) {
        username = uname;
        next = nullptr;
    }

    UserFriend::UserFriend(string uname) {
        username = uname;
        friendsHead = nullptr;
        next = nullptr;
    }

```

main.cpp

```

#include "menu.h"
#include "graph.h"
#include "credential.h"
#include "data.h"

int main() {
    UserGraph userGraph;
    CredentialManager credentials;
    string currentLoggedInUser;

    while (true) {
        displayMenu();
        int choice;
        cin >> choice;
        if (choice == 0) break;

        string username, password, friendName, content, sender, receiver;
        switch (choice) {
            case 1: // Signup
                cout << "Enter username: ";
                cin >> username;
                cout << "Enter password: ";
                cin >> password;
                if (!credentials.findUser(username)) {

```

```

        credentials.addUser(username, password);
        userGraph.addUser(username);
        cout << "Signup successful!" << endl;
    }
    else {
        cout << "Username already exists." << endl;
    }
    break;

case 2: // Login
    if (!currentLoggedInUser.empty()) {
        cout << "You are already logged in as " << currentLoggedInUser <<
". Logout first." << endl;
        break;
    }
    cout << "Enter username: ";
    cin >> username;
    cout << "Enter password: ";
    cin >> password;
    if (credentials.authenticate(username, password)) {
        currentLoggedInUser = username;
        cout << "Login successful! Welcome, " << username << "." << endl;
    }
    else {
        cout << "Invalid username or password." << endl;
    }
    break;

case 3: // Logout
    if (!currentLoggedInUser.empty()) {
        cout << "Goodbye, " << currentLoggedInUser << "." << endl;
        currentLoggedInUser.clear();
    }
    else {
        cout << "No user is currently logged in." << endl;
    }
    break;

case 4: // Send Friend Request
    if (currentLoggedInUser.empty()) {
        cout << "Please login first." << endl;
        break;
    }
    cout << "Enter friend's username: ";
    cin >> friendName;
    userGraph.addFriendRequest(currentLoggedInUser, friendName);
    cout << "Friend request sent!" << endl;
    break;

case 5: // View Notifications
    if (currentLoggedInUser.empty()) {
        cout << "Please login first." << endl;
        break;
    }
    userGraph.displayNotifications(currentLoggedInUser);
    break;

case 6: // Post Content
    if (currentLoggedInUser.empty()) {
        cout << "Please login first." << endl;
        break;
    }

```

```
        cout << "Enter post content: ";
        cin.ignore();
        getline(cin, content);
        userGraph.postContent(currentLoggedInUser, content);
        cout << "Post added!" << endl;
        break;

    case 7: // View Posts
        if (currentLoggedInUser.empty()) {
            cout << "Please login first." << endl;
            break;
        }
        userGraph.viewPosts(currentLoggedInUser);
        break;

    case 8: // Send Message
        if (currentLoggedInUser.empty()) {
            cout << "Please login first." << endl;
            break;
        }
        cout << "Enter receiver's username: ";
        cin >> receiver;
        cout << "Enter message content: ";
        cin.ignore();
        getline(cin, content);
        userGraph.sendMessage(currentLoggedInUser, receiver, content);
        cout << "Message sent!" << endl;
        break;

    case 9: // View Messages
        if (currentLoggedInUser.empty()) {
            cout << "Please login first." << endl;
            break;
        }
        userGraph.viewMessages(currentLoggedInUser);
        break;

    case 10: // View Friends
        if (currentLoggedInUser.empty()) {
            cout << "Please login first." << endl;
            break;
        }
        userGraph.displayFriends(currentLoggedInUser);
        break;

    case 11: // View Friend Requests
        if (currentLoggedInUser.empty()) {
            cout << "Please login first." << endl;
            break;
        }
        userGraph.displayFriendRequests(currentLoggedInUser);
        break;

    case 12: // Accept Friend Request
        if (currentLoggedInUser.empty()) {
            cout << "Please login first." << endl;
            break;
        }
        cout << "Enter sender's username to accept the friend request: ";
        cin >> sender;
        userGraph.acceptFriendRequest(currentLoggedInUser, sender);
        break;
```

```
        case 13: // Reject Friend Request
            if (currentLoggedInUser.empty()) {
                cout << "Please login first." << endl;
                break;
            }
            cout << "Enter sender's username to reject the friend request: ";
            cin >> sender;
            userGraph.rejectFriendRequest(currentLoggedInUser, sender);
            break;

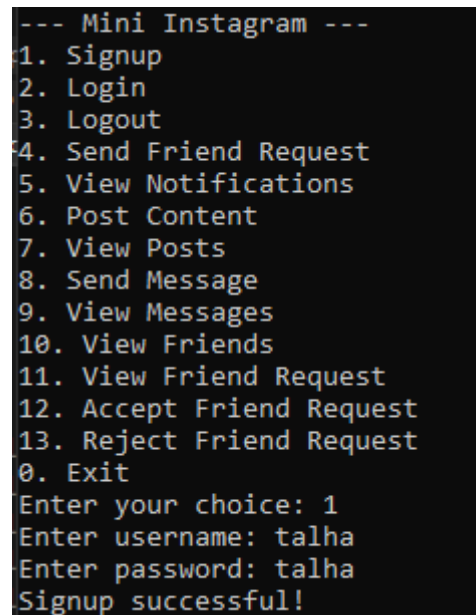
        default:
            cout << "Invalid choice." << endl;
    }
}

return 0;
}
```

Screenshot

Signup

For Talha



```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 1
Enter username: talha
Enter password: talha
Signup successful!
```

For Abdullah

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 1
Enter username: abdullah
Enter password: abdullah
Signup successful!
```

Login

For Talha

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 2
Enter username: talha
Enter password: talha
Login successful! Welcome, talha.
```

For Abdullah

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 2
Enter username: abdullah
Enter password: abdullah
Login successful! Welcome, abdullah.
```

Logout

For Talha

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 3
Goodbye, talha.
```


For Abdullah

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 3
Goodbye, abdullah.
```

Send Friend Request**For Talha**

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 4
Enter friend's username: abdullah
Friend request sent!
```

For Abdullah

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 4
Enter friend's username: ammar
Friend request sent!
```

View Notifications**For Talha**

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 5
Queue is empty.
```

For Abdullah

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 5
talha sent you a message: hello abdullah
```

Post Content

For Talha

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 6
Enter post content: test post
Post added!
```

For Abdullah

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 6
Enter post content: test post from abdullah
Post added!
```

View Posts

For Talha

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 7
Fri Dec 13 02:04:17 2024
: test post
```

For Abdullah

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 7
Fri Dec 13 02:13:12 2024
: test post from abdullah
```

Send Message**For Talha**

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 8
Enter receiver's username: abdullah
Enter message content: hello abdullah
Message sent!
```

For Abdullah

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 8
Enter receiver's username: talha
Enter message content: hello talha
Message sent!
```

View Messages

For Talha

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 9
No messages available.
```

For Abdullah

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 9
talha: hello abduallah
```

View Friends

For Talha

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 10
Friends of talha:
```

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 10
Friends of talha: abduallah
```

For Abdullah

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 10
Friends of abdullah: talha
```

View Friend Request**For Talha**

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 11
Friend Requests for talha:
Queue is empty.
```


For Abdullah

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 11
Friend Requests for abdullah:
talha sent you a friend request.
```

Accept Friend Request

For Talha

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 12
Enter sender's username to accept the friend request: abdullah
Friend request not found.
```

For Abdullah

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 12
Enter sender's username to accept the friend request: talha
Friend request accepted!
```

Reject Friend Request

For Talha

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 13
Enter sender's username to reject the friend request: abdullah
Friend request not found.
```

For Abdullah

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 13
Enter sender's username to reject the friend request: talha
Friend request not found.
```

Exit

```
--- Mini Instagram ---
1. Signup
2. Login
3. Logout
4. Send Friend Request
5. View Notifications
6. Post Content
7. View Posts
8. Send Message
9. View Messages
10. View Friends
11. View Friend Request
12. Accept Friend Request
13. Reject Friend Request
0. Exit
Enter your choice: 00
```