Namal University Mianwali

Department of Electrical Engineering

Data Structures and Algorithms

Course Title: CSC-200L

Project Report

Social Media Friend Recommendation Simulator

Name	Hassan Ali Ahmad	Abdul Rehman	Muqadas Bibi
Roll No.	NUM-BSEE-2023-03	NUM-BSEE-2023-36	NUM-BSEE-2022-18

Instructor: Dr. Farrukh Qureshi

Lab Engineer: Engr. Sana Perveen

Contents

1 Introduction

The Enhanced Friend Request Simulator System is an academic project designed to demonstrate the practical use of Data Structures and Algorithms (DSA) in a real-world inspired web application. The system simulates a social networking platform where users can create accounts, manage profiles, send and accept friend requests, search users, and communicate through messages.

The main goal of this project is not to build a commercial social network, but to show how theoretical concepts such as graphs, hash tables, sets, queues, and sorting algorithms can be applied to solve real problems.

2 Project Flow

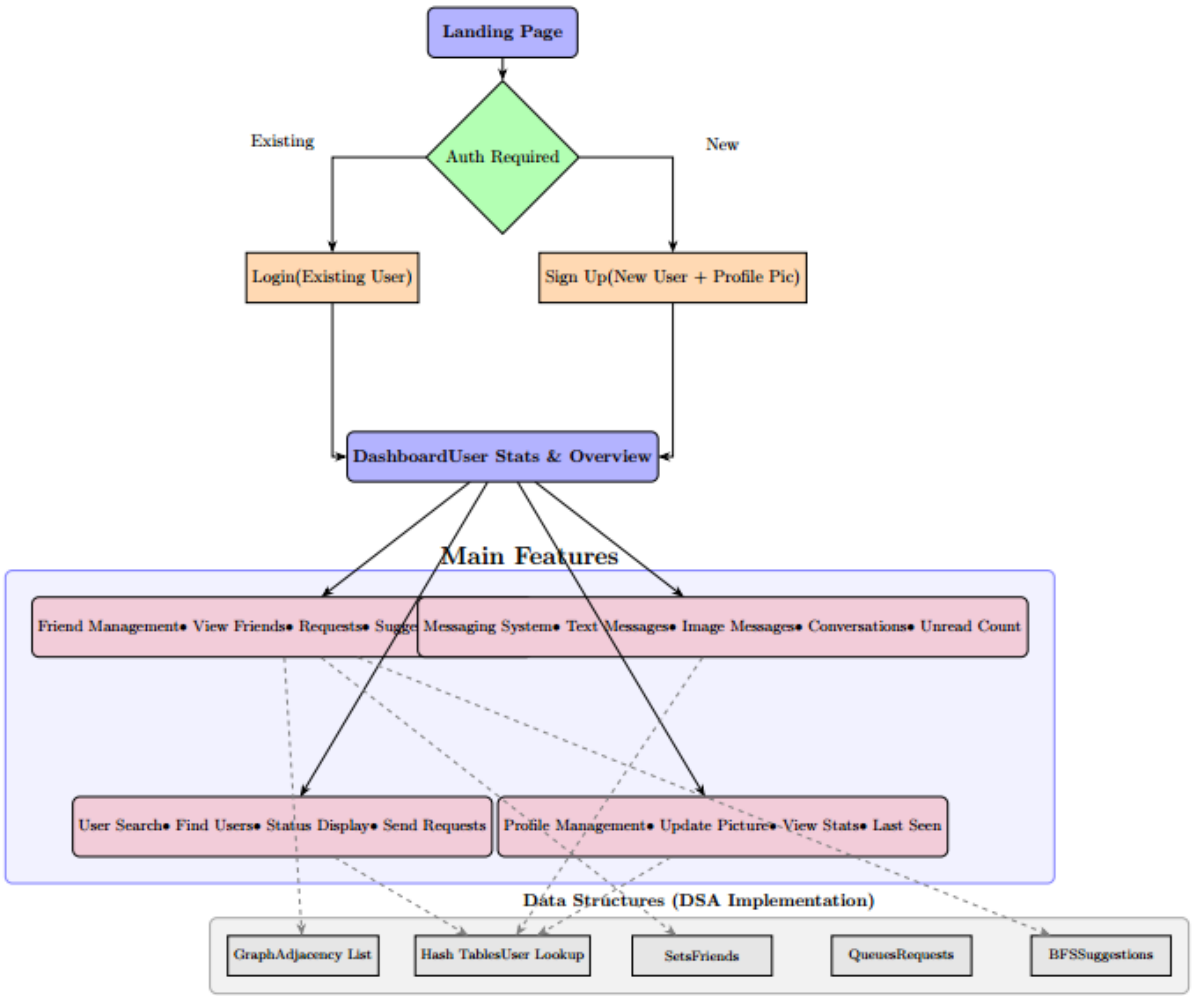

Figure 1: project flow

3 System Theory and Data Structure Foundations

3.1 Social Network as a Graph

Each user is a node and each friendship is an edge. An adjacency list is used for memory efficiency and easy traversal.

3.2 Hash Tables for User Management

Users are stored in a dictionary using username as key, giving O(1) access time.

3.3 Sets for Friend Lists

Sets avoid duplicate friends and allow fast lookup and mutual friend detection.

3.4 Queues and Lists

Friend requests follow queue behavior. Messages are stored in lists to maintain order.

3.5 Sorting and Searching Theory

Friend suggestions use TimSort. User search uses linear substring matching.

3.6 Security Theory

Passwords are hashed. Sessions manage login securely.

3.7 Data Persistence Theory

Data is stored in JSON using serialization and deserialization.

4 Algorithm Design and Implementation

4.1 User Management Algorithms

4.1.1 User Registration and Login Algorithm

Listing 1: User Registration Function

```
def register_user(username, password):
    if username in users:
        return "Username already exists"
    hashed = hash_password(password)
    users[username] = {
        "password": hashed,
        "friends": set(),
        "pending": [],
        "sent": []
    }
    graph[username] = []
    save_data()
    return "Registration successful"
```

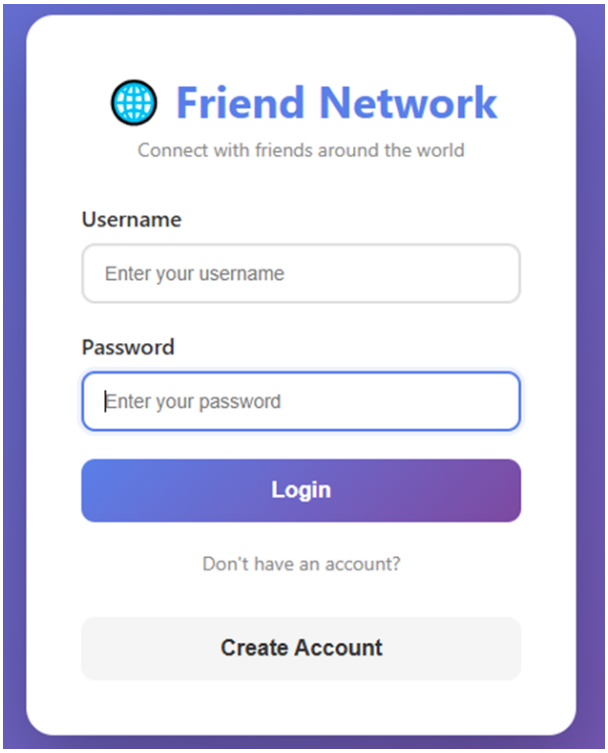

Figure 2: user Login

4.1.2 Registration Steps

1. Check username in hash table
2. Hash password
3. Store user object
4. Initialize graph node

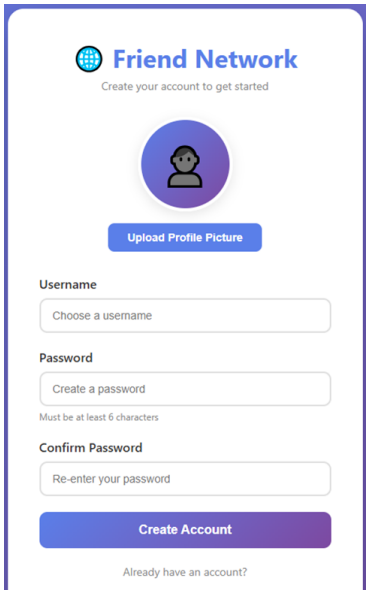

Figure 3: User Registration

4.2 Friend Request System

4.2.1 Friend Request Algorithm

Listing 2: Send Friend Request

```python
def send_request(sender, receiver):
    if receiver not in users:
        return "User not found"
    if receiver == sender:
        return "Cannot send request to yourself"
    if receiver in users[sender]["friends"]:
        return "Already friends"
    if sender in users[receiver]["pending"]:
        return "Request already sent"
    users[sender]["sent"].append(receiver)
    users[receiver]["pending"].append(sender)
    save_data()
    return "Request sent"
```

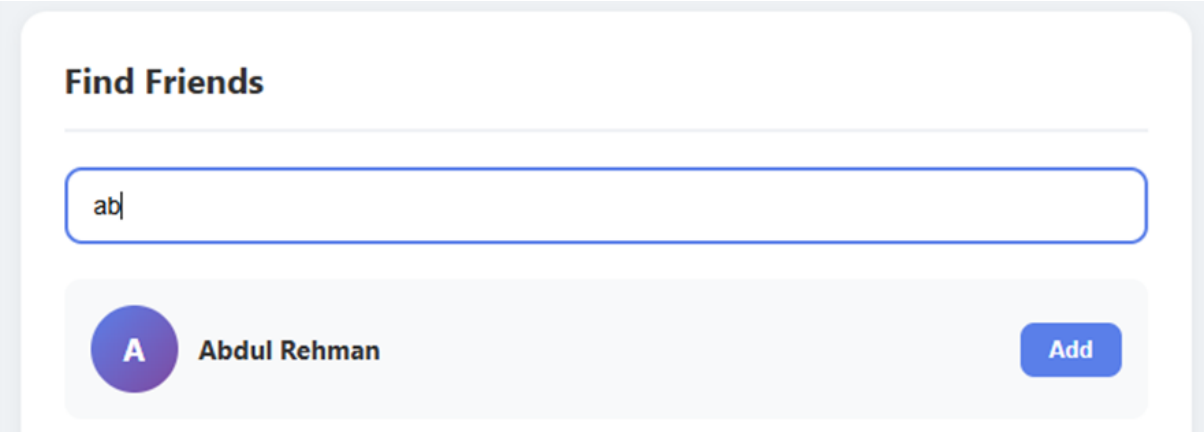

Figure 4: user Login

4.2.2 Friend Acceptance Algorithm

Listing 3: Accept Friend Request

```python
def accept_request(receiver, sender):
    users[receiver]["pending"].remove(sender)
    users[sender]["sent"].remove(receiver)
    users[receiver]["friends"].add(sender)
    users[sender]["friends"].add(receiver)
    graph[receiver].append(sender)
    graph[sender].append(receiver)
    save_data()
```

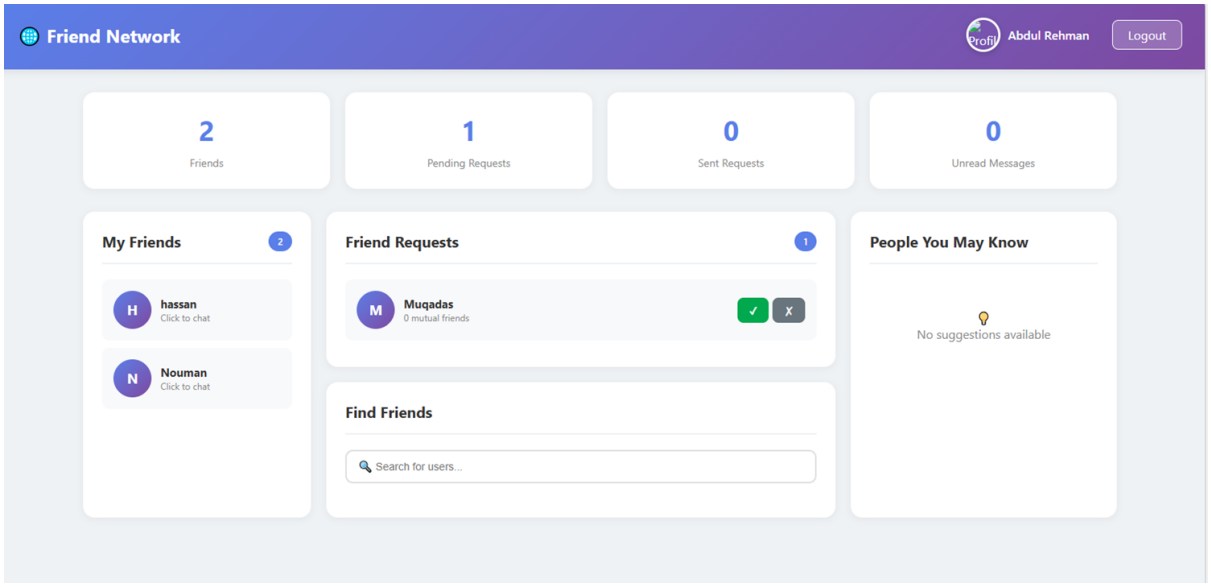

Figure 5: Friend Request Acceptance Algorithm

4.3 Friend Recommendation System

4.3.1 Friend Suggestion Code

Listing 4: Friend Suggestion Logic

```
def suggest_friends(user):
    suggestions = {}
    for friend in users[user]["friends"]:
        for fof in users[friend]["friends"]:
            if fof != user and fof not in users[user]["friends"]:
                suggestions[fof] = suggestions.get(fof, 0) + 1
    return sorted(suggestions.items(), key=lambda x: x[1], reverse=True
        )
```

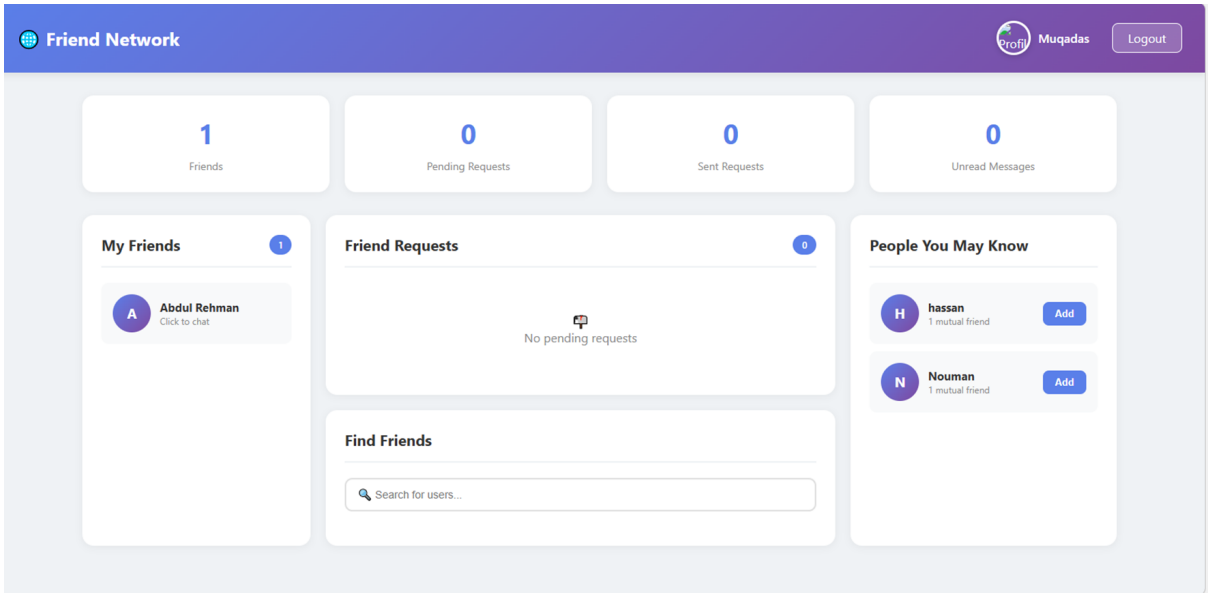

Figure 6: Friend Suggestion Algorithm

4.4 Search Mechanism

Linear scanning matches substring in usernames.

4.5 Messaging System

4.5.1 Messaging Code

Listing 5: Send Message Function

```
def send_message(sender, receiver, text):
    if receiver not in users or receiver not in users[sender]["friends"
        ]:
         return "Can only message friends"
    message = {
         "from": sender,
         "to": receiver,
         "text": text,
         "read": False
    }
    messages.append(message)
    save_data()
```

4.5.2 Unread Message Handling

Unread messages are counted by checking read status.

4.6 Profile Management

Profile pictures are validated, sanitized, and saved.

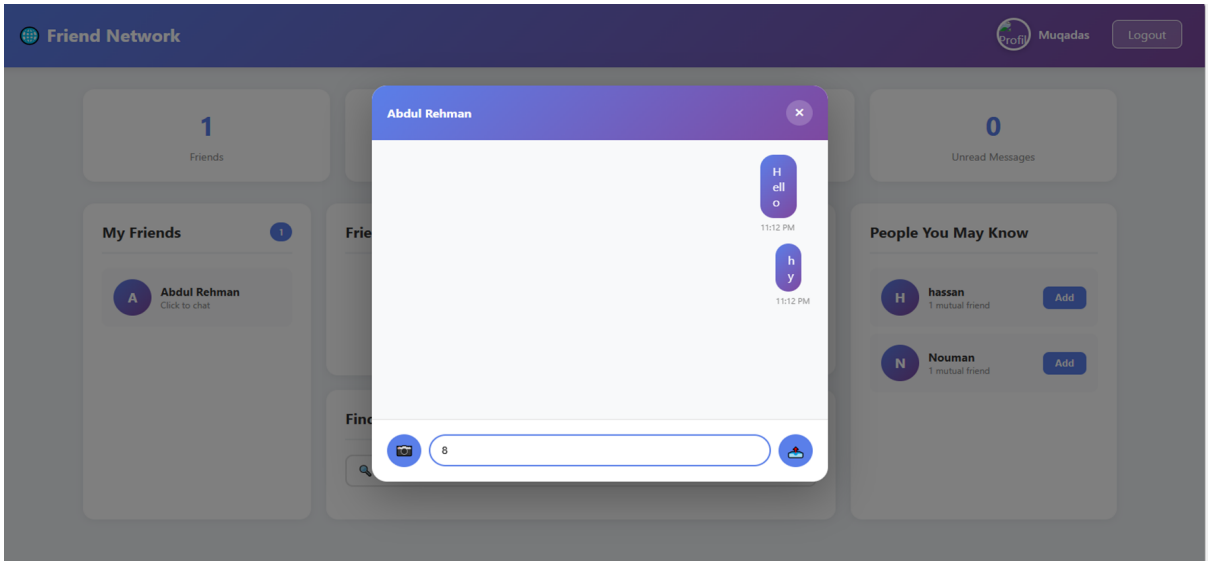

Figure 7: Message Algorithm

5 Results and Performance Analysis

5.1 System Behavior

The system correctly handles user creation, friendships, and messaging.

5.2 Feature Verification

All features were tested with multiple users.

5.3 Time Complexity Summary

- User lookup: O(1)
- Friend check: O(1)
- Search: O(n)
- Friend suggestion: O(V+E)
- Message retrieval: O(m)

6 Conclusion

This project demonstrates practical use of DSA concepts through a social network simulator. Graphs, hash tables, sets, queues, and sorting algorithms ensure efficiency and correctness while remaining suitable for academic learning.