



# **Namal University Mianwali**

**Department of Electrical Engineering**

**Data Structures and Algorithms**

**Course Title: CSC-200L**

**Project Report**

## **Social Media Friend Recommendation Simulator**

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

**Instructor: Dr. Farrukh Qureshi**

**Lab Engineer: Engr. Sana Perveen**

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Project Flow</b>	<b>2</b>
<b>3</b>	<b>System Theory and Data Structure Foundations</b>	<b>3</b>
3.1	Social Network as a Graph . . . . .	3
3.2	Hash Tables for User Management . . . . .	3
3.3	Sets for Friend Lists . . . . .	3
3.4	Queues and Lists . . . . .	3
3.5	Sorting and Searching Theory . . . . .	3
3.6	Security Theory . . . . .	3
3.7	Data Persistence Theory . . . . .	3
<b>4</b>	<b>Algorithm Design and Implementation</b>	<b>3</b>
4.1	User Management Algorithms . . . . .	3
4.1.1	User Registration and Login Algorithm . . . . .	3
4.1.2	Registration Steps . . . . .	4
4.2	Friend Request System . . . . .	5
4.2.1	Friend Request Algorithm . . . . .	5
4.2.2	Friend Acceptance Algorithm . . . . .	5
4.3	Friend Recommendation System . . . . .	6
4.3.1	Friend Suggestion Code . . . . .	6
4.4	Search Mechanism . . . . .	7
4.5	Messaging System . . . . .	7
4.5.1	Messaging Code . . . . .	7
4.5.2	Unread Message Handling . . . . .	7
4.6	Profile Management . . . . .	7
<b>5</b>	<b>Results and Performance Analysis</b>	<b>8</b>
5.1	System Behavior . . . . .	8
5.2	Feature Verification . . . . .	8
5.3	Time Complexity Summary . . . . .	8
<b>6</b>	<b>Conclusion</b>	<b>8</b>

# 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

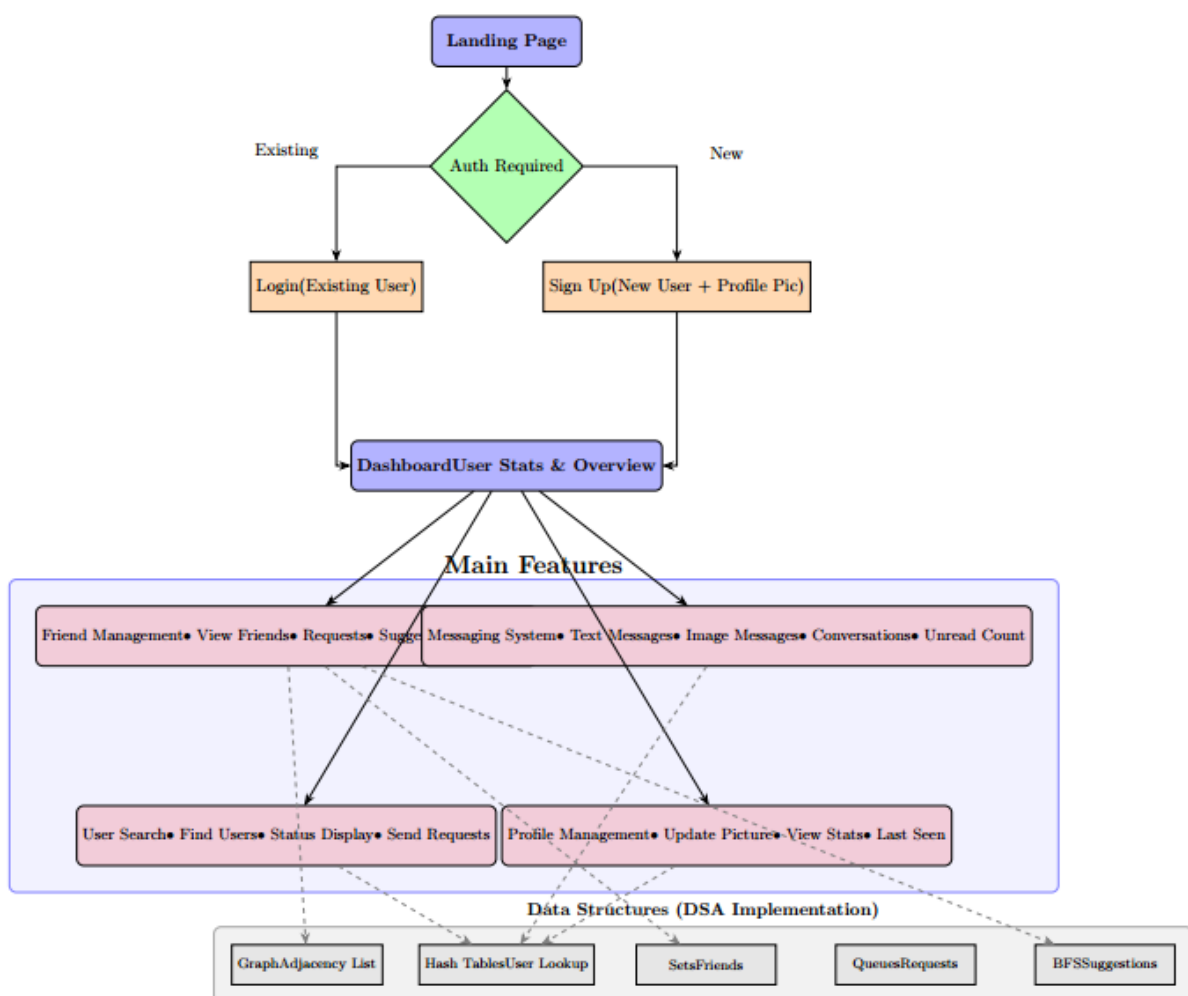


Figure 1: project flow

## 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.

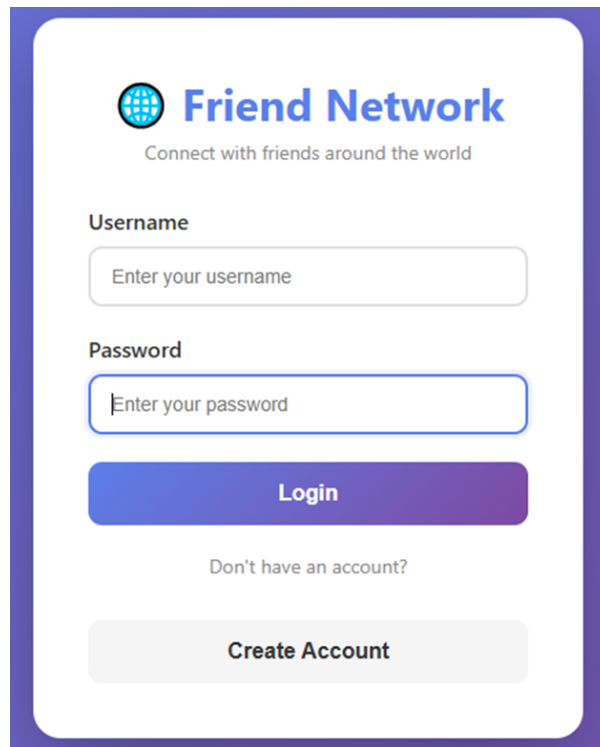
## 4 Algorithm Design and Implementation

### 4.1 User Management Algorithms

#### 4.1.1 User Registration and Login Algorithm

Listing 1: User Registration Function

```
1 def register_user(username, password):
2     if username in users:
3         return "Username already exists"
4     hashed = hash_password(password)
5     users[username] = {
6         "password": hashed,
7         "friends": set(),
8         "pending": [],
9         "sent": []
10    }
11    graph[username] = []
12    save_data()
13    return "Registration successful"
```



The login form for Friend Network features a blue header with the logo and tagline. It includes input fields for Username and Password, a blue Login button, a link for users without an account, and a grey Create Account button.

**Friend Network**  
Connect with friends around the world

**Username**  
Enter your username

**Password**  
Enter your password

**Login**

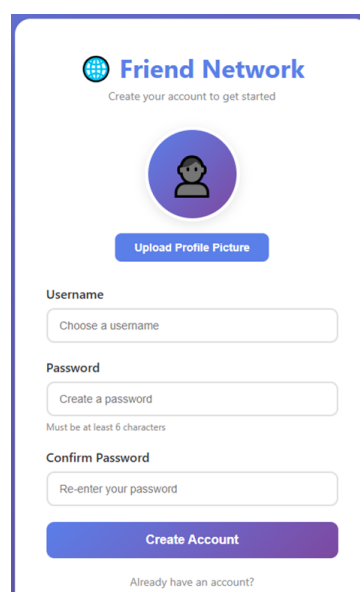
Don't have an account?

**Create Account**

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



The registration form for Friend Network includes a header with the logo and tagline. It features a profile picture upload section, followed by input fields for Username, Password, and Confirm Password. A blue Create Account button is at the bottom, with a link for existing users below it.

**Friend Network**  
Create your account to get started

Upload Profile Picture

**Username**  
Choose a username

**Password**  
Create a password  
Must be at least 6 characters

**Confirm Password**  
Re-enter your password

**Create Account**

Already have an account?

Figure 3: User Registration

## 4.2 Friend Request System

### 4.2.1 Friend Request Algorithm

Listing 2: Send Friend Request

```
1 def send_request(sender, receiver):
2     if receiver not in users:
3         return "User not found"
4     if receiver == sender:
5         return "Cannot send request to yourself"
6     if receiver in users[sender]["friends"]:
7         return "Already friends"
8     if sender in users[receiver]["pending"]:
9         return "Request already sent"
10    users[sender]["sent"].append(receiver)
11    users[receiver]["pending"].append(sender)
12    save_data()
13    return "Request sent"
```

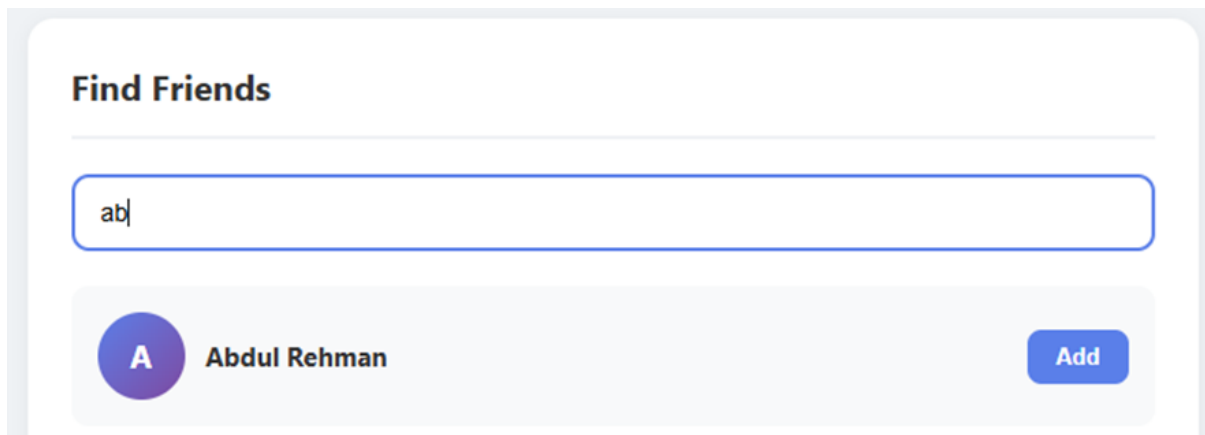


Figure 4: user Login

### 4.2.2 Friend Acceptance Algorithm

Listing 3: Accept Friend Request

```
1 def accept_request(receiver, sender):
2     users[receiver]["pending"].remove(sender)
3     users[sender]["sent"].remove(receiver)
4     users[receiver]["friends"].add(sender)
5     users[sender]["friends"].add(receiver)
6     graph[receiver].append(sender)
7     graph[sender].append(receiver)
8     save_data()
```

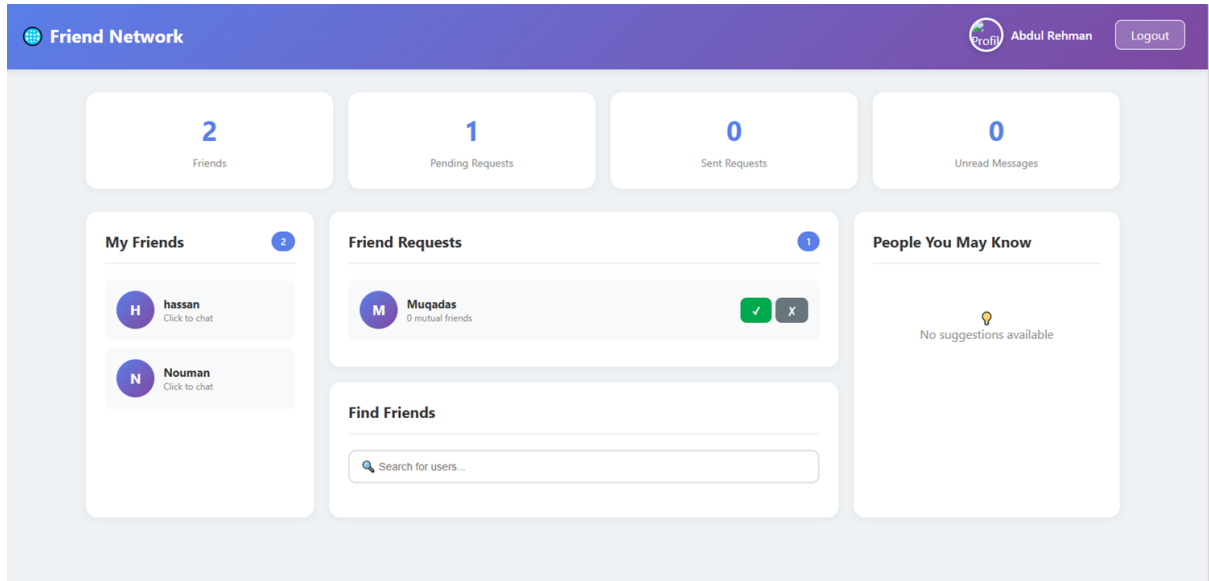


Figure 5: Friend Request Acceptance Algorithm

## 4.3 Friend Recommendation System

### 4.3.1 Friend Suggestion Code

Listing 4: Friend Suggestion Logic

```

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

```

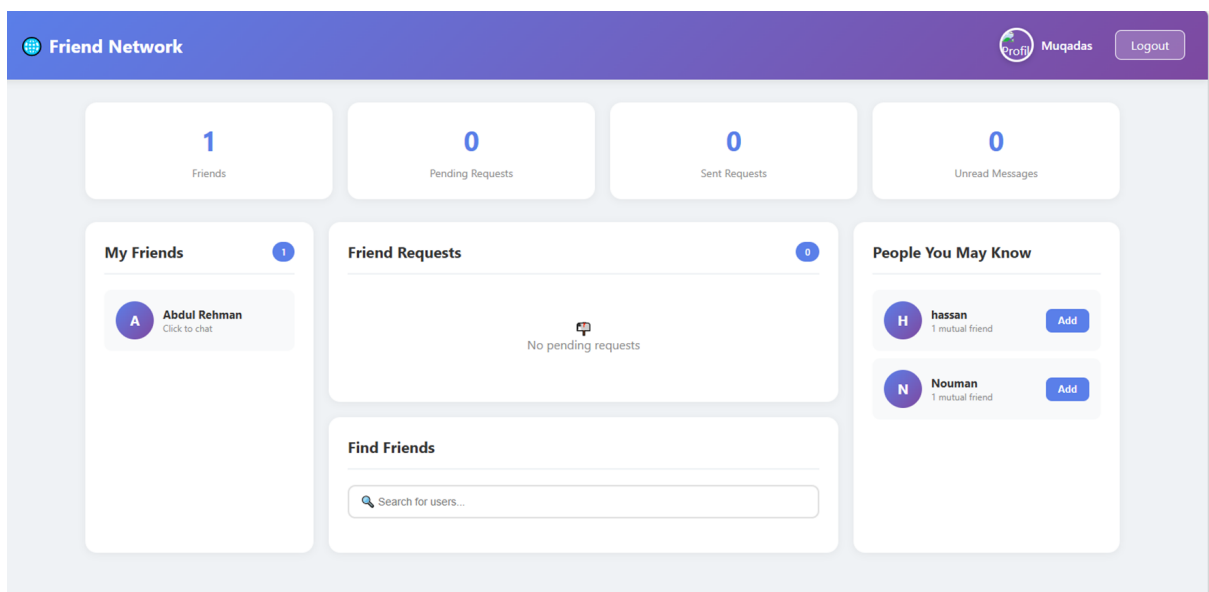


Figure 6: Friend Suggestion Algorithm

## 4.4 Search Mechanism

Linear scanning matches substring in usernames.

## 4.5 Messaging System

### 4.5.1 Messaging Code

Listing 5: Send Message Function

```
1 def send_message(sender, receiver, text):
2     if receiver not in users or receiver not in users[sender]["friends"]
3         return "Can only message friends"
4     message = {
5         "from": sender,
6         "to": receiver,
7         "text": text,
8         "read": False
9     }
10    messages.append(message)
11    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.

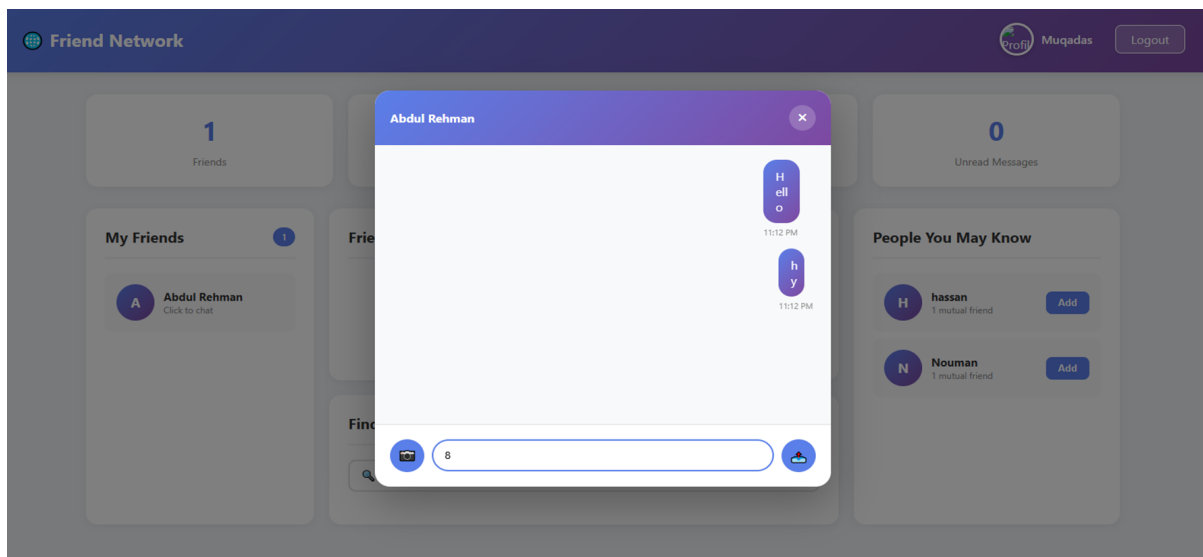


Figure 7: Message Algorithm



## 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.