



## **DSA Lab Project**

**Submitted by:**

Mah-Gul Khan     2024-CS-211

**Supervised by:**

Ma'am Rabeeya Saleem

**Course:**

CSC-200L Data Structures

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

## Table of Contents

|     |                               |    |
|-----|-------------------------------|----|
| 1.  | Project Title: _____          | 3  |
| 2.  | Project Description: _____    | 3  |
| 3.  | Objectives: _____             | 3  |
| 4.  | Data Structure Used: _____    | 3  |
| 5.  | Algorithms Used: _____        | 4  |
| 6.  | System Flow Diagram: _____    | 6  |
| 7.  | Implementation Details: _____ | 6  |
| 8.  | Case Study: _____             | 7  |
| 9.  | Wireframes: _____             | 9  |
| 10. | Performance Analysis: _____   | 13 |

## 1. Project Title:

The title for my DSA mid project is: **The Campus Chronicle.**

## 2. Project Description:

This project, **The Campus Chronicle**, is designed to bridge that gap by demonstrating how a simplified social media feed system can be built using fundamental data structures. The system allows university students to share campus happenings, interact with each other, view timelines, and express engagement through likes and comments, similar to real-world social platforms.

## 3. Objectives:

The primary objective of this project is:

- To design and implement a mini social media application, **Campus Chronicle**, where university students can post updates and interact with each other.
- To provide a practical demonstration of how real applications use core data structures such as linked lists, queues, stacks, and searching/sorting algorithms.
- To enhance understanding of data structure operations and their real-world relevance.
- To show how posts, interactions, and histories can be managed efficiently using structured storage and algorithmic processing.
- To give users a simple and intuitive interface while internally showcasing how modern applications manage user-generated content.

## 4. Data Structure Used:

In this project, three primary data structures were implemented to simulate the internal workings of a social media feed system: **Linked List, Stack, and Queue.**

### 1. **Linked List**

- **Purpose:** Store and manage user data

- **Implementation:** Singly linked list
- **Operations:**  $O(n)$  for search,  $O(1)$  for insertion

## 2. Stack

- **Purpose:** Track user navigation between forms
- **Implementation:** Linked list-based stack
- **Operations:**  $O(1)$  for push/pop operations

## 3. Queue

- **Purpose:** Manage posts in chronological order
- **Implementation:** Linked list-based queue
- **Operations:**  $O(1)$  for enqueue/dequeue

## 4. Array

- **Purpose:** Track user actions for undo functionality
- **Implementation:** Circular array with fixed size
- **Operations:**  $O(1)$  for most operations

## 5. Algorithms Used:

This project implements three core algorithms to support searching, sorting, and arranging content within the Campus Chronicle system. Each algorithm was chosen for simplicity, clarity, and suitability for small to medium data sizes.

### 5.1. Linear Search

Linear Search was used to perform **keyword-based search in posts**. When the user enters a keyword, the system scans each post sequentially to check if the keyword appears in the post content.

This approach is easy to implement and works well for moderate-sized datasets typical of student or campus applications.

### Why Linear Search?

- Simple and intuitive algorithm
- Does not require data to be sorted
- Works well for dynamic data structures like Linked Lists

## 5.2. Bubble Sort

Bubble Sort was used in two areas:

### ➤ **Sorting Usernames Alphabetically**

The system sorts usernames alphabetically for better organization and visual presentation. Bubble Sort repeatedly compares adjacent items and swaps them if they are in the wrong order.

### ➤ **Sorting Posts in Feed**

Posts can be sorted using Bubble Sort based on different criteria such as:

- Creation time
- Likes
- Alphabetical order

This allows students to understand how real feed ordering is handled internally.

### **Why Bubble Sort?**

- Very easy to understand and implement
- Ideal for demonstrating sorting logic in educational projects
- Works well for smaller datasets where performance is not critical

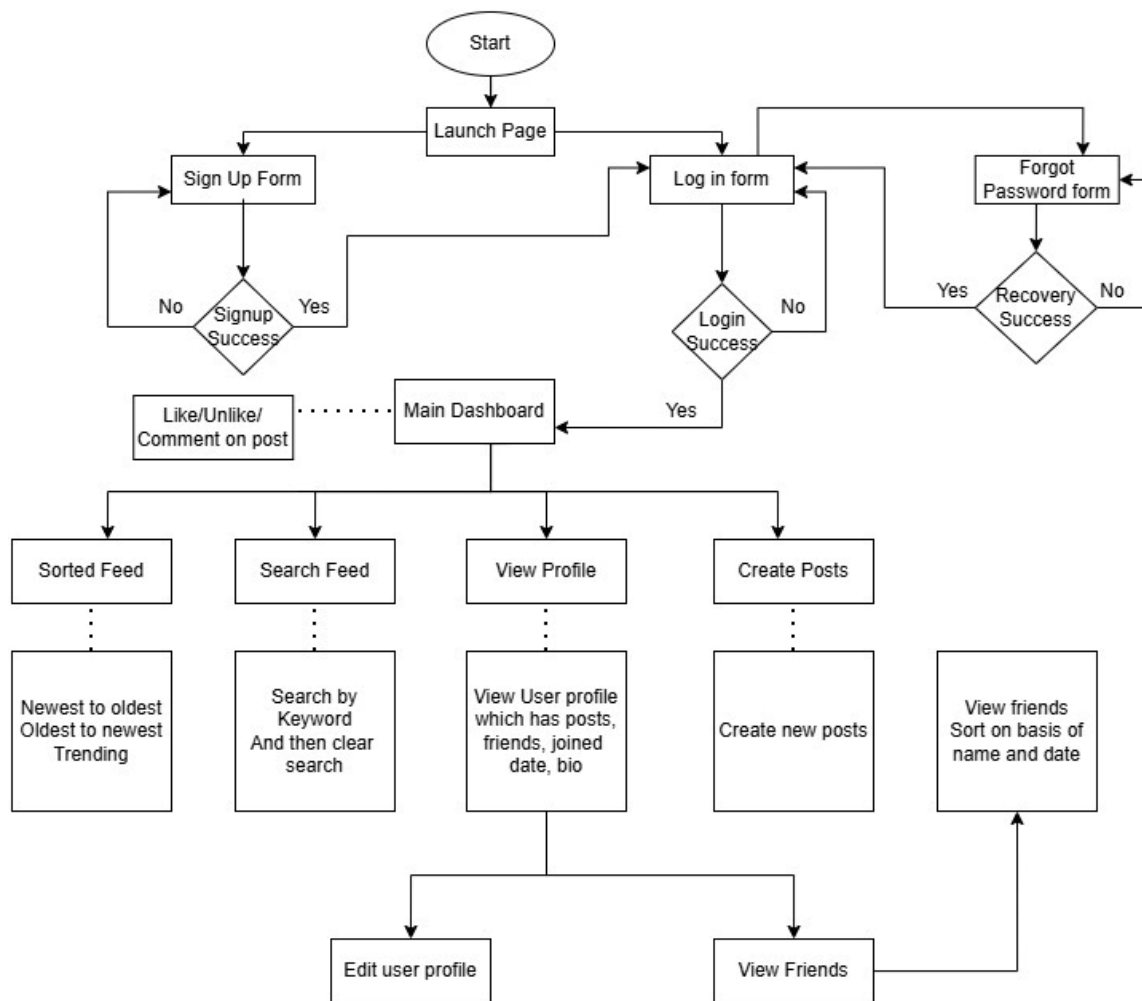
## 5.3. Feed Ordering and Queue Sorting

While the Queue ensures posts are displayed in the order they are added, sorting is applied when the user wants to:

- Sort by time
- Sort alphabetically
- Sort by engagement (likes, comments, etc.)

In such cases, Bubble Sort is applied to arrange the posts after extraction while still demonstrating real-world feed behaviour.

## 6. System Flow Diagram:



## 7. Implementation Details:

### 7.1. Authentication System

- User registration with validation
- Password hashing using SHA256
- Session management with static class

### 7.2. Post Management

- Queue-based post storage
- Like/comment functionality
- Sorting algorithms:
  - **Bubble Sort** for trending posts (by likes)
  - **Selection Sort** for recent posts

### **7.3. Search Functionality**

- Linear search for keyword matching
- User search by username
- Friend recommendation system

### **7.4. Navigation System**

- Stack-based navigation history
- Form tracking and backtracking
- State preservation between forms

### **7.5. Database Integration**

- MySQL database connectivity
- Singleton pattern for database helper
- CRUD operations for all entities

## **8. Case Study:**

### **8.1. User Registration and Authentication**

1. User Input: User enters credentials in Login form (UI Layer)
2. Validation: Input validated against business rules (BL Layer)
3. Verification: UserCRUD.Login() checks credentials (DL Layer)
4. Data Access: Database queried for user verification (DatabaseHelper)
5. Session Setup: SessionManager.Login() sets user properties (BL Layer)
6. Navigation: User redirected to Dashboard (UI Layer)

### **8.2. Post Creation**

1. Content Creation: User writes post in PostsForm (UI Layer)
2. Object Creation: Post object instantiated with metadata (BL Layer)
3. Queue Management: Post added to PostQueue (Data Structures)
4. Database Persistence: Post saved to MySQL database (DatabaseHelper)
5. Feed Update: Queue traversed to display posts (DL Layer → UI Layer)

### **8.3. Social Interaction**

1. Like Action: User clicks like button on post (UI Layer)
2. Validation: PostsCrud checks if user already liked post (BL Layer)
3. Count Update: Post like count incremented in PostQueue (Data Structures)
4. Database Sync: Like record inserted in database (DatabaseHelper)
5. History Tracking: Action saved in BrowsingHistory array for undo

## **8.4. Search and Discovery**

1. Search Input: User enters search keyword (UI Layer)
2. Linear Search: PostsCrud.SearchPosts() traverses PostQueue (BL Layer)
3. Result Filtering: Matching posts collected (Data Structures)
4. Sorting: Bubble sort for trending, Selection sort for recent (Algorithms)
5. Display: Sorted results displayed in DataGridView (UI Layer)

## **8.5. Navigation**

1. Form Transition: User navigates to new form (UI Layer)
2. State Capture: Current form data captured in FormDataBL (BL Layer)
3. Stack Push: Form data pushed to NavigationStack (Data Structures)
4. History Management: Stack maintains navigation trail
5. Back Navigation: Stack pop retrieves previous form state

## **8.6. Undo**

1. User Discovery: Search for users through Search form (UI Layer → DL Layer)
2. Profile Verification: UserCRUD.FindProfile() locates user (BL Layer)
3. List Update: Friend added to FriendsList linked list (Data Structures)
4. Database Sync: Friendship record inserted in database (DatabaseHelper)

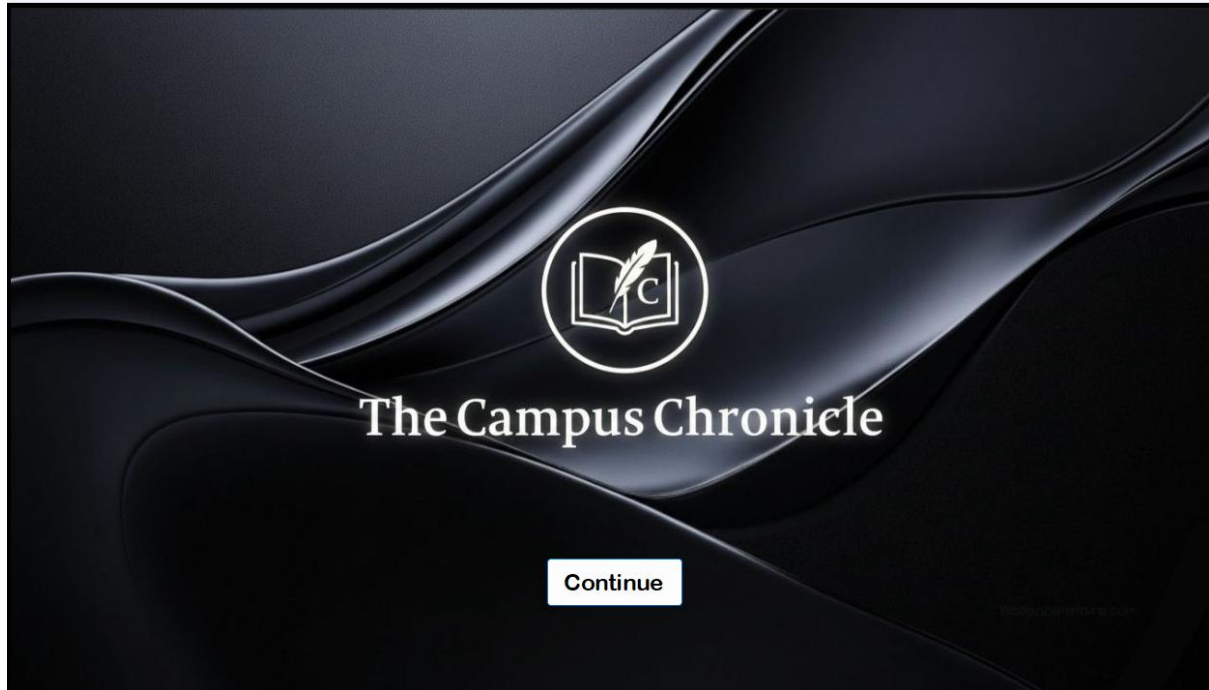
## **8.7. Comments**

1. Comment Input: User enters comment text (UI Layer)
2. Object Creation: Comment object created with post relation (BL Layer)
3. Queue Management: Comment added to CommentQueue (Data Structures)
4. Count Update: Post comment count incremented (DL Layer)
5. Database Sync: Comment saved to database (DatabaseHelper)

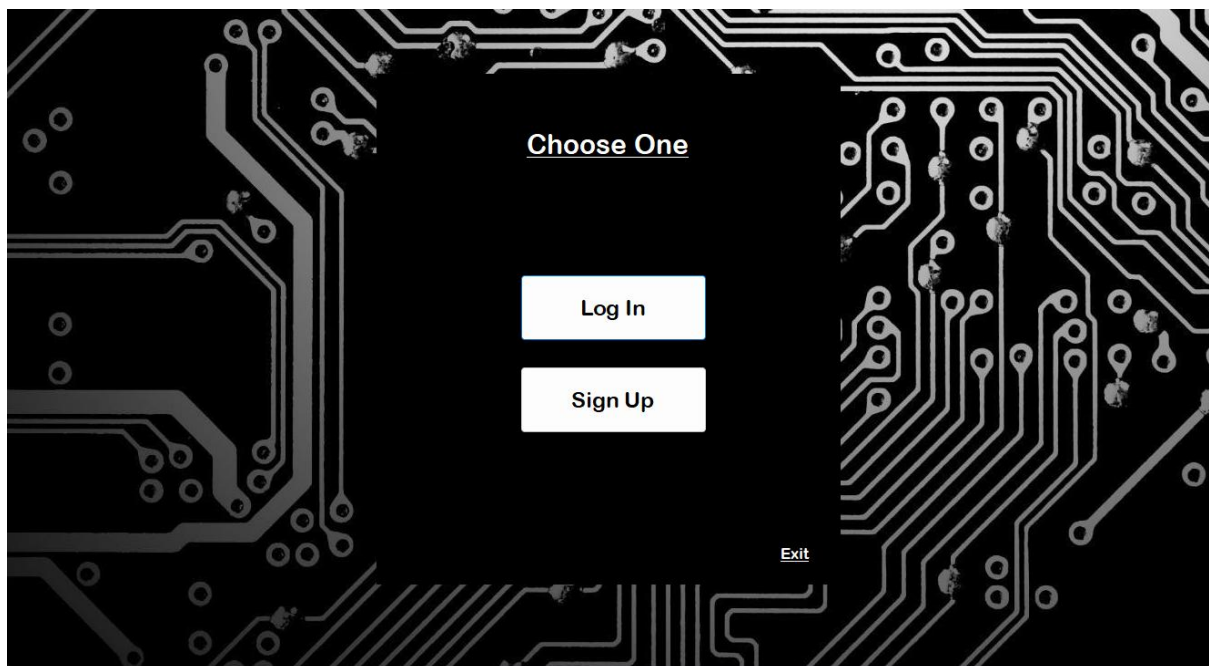


## 9. Wireframes:

### 9.1. Launch Page



### 9.2 . Login/Sign up Select



### 9.3. Main Dashboard

*The Campus Chronicle*

Profile

All Posts

| Sr.no | Username | Content                                    | Likes | Comment | CreatedA   |
|-------|----------|--|-------|---------|------------|
| 1     | mahgul   | Hiii Guysss!                               | 5     | 1       | 22/11/2... |
| 2     | ayesha   | Hey everyone                               | 3     | 2       | 22/11/2... |
| 3     | fatima   | So tired Lately                            | 4     | 2       | 22/11/2... |
| 4     | mahgul   | OUR DSA PROJECTT IS DUE ON MONDAYYY!!!!... | 3     | 2       | 22/11/2... |
| 6     | mahgul   | 32 convocation                             | 1     | 0       | 24/11/2... |
| 7     | ayesha   | lol  | 1     | 1       | 24/11/2... |
| 8     | ahmad    | hello people                               | 1     | 0       | 24/11/2... |
| 9     | ahmad    | wanna skip school                          | 1     | 0       | 24/11/2... |
| 12    | fatima   | nicee                                      | 1     | 0       | 24/11/2... |

Search

New Post

Undo

Refresh

Log Out

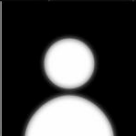
Exit



Sort

### 9.4. User Profile

*The Campus Chronicle*



User Profile

| UserID | Username | Email            |
|--------|----------|------------------|
| 1      | mahgul   | mahgul2310@gmail |
| *      |          |                  |

Edit Profile

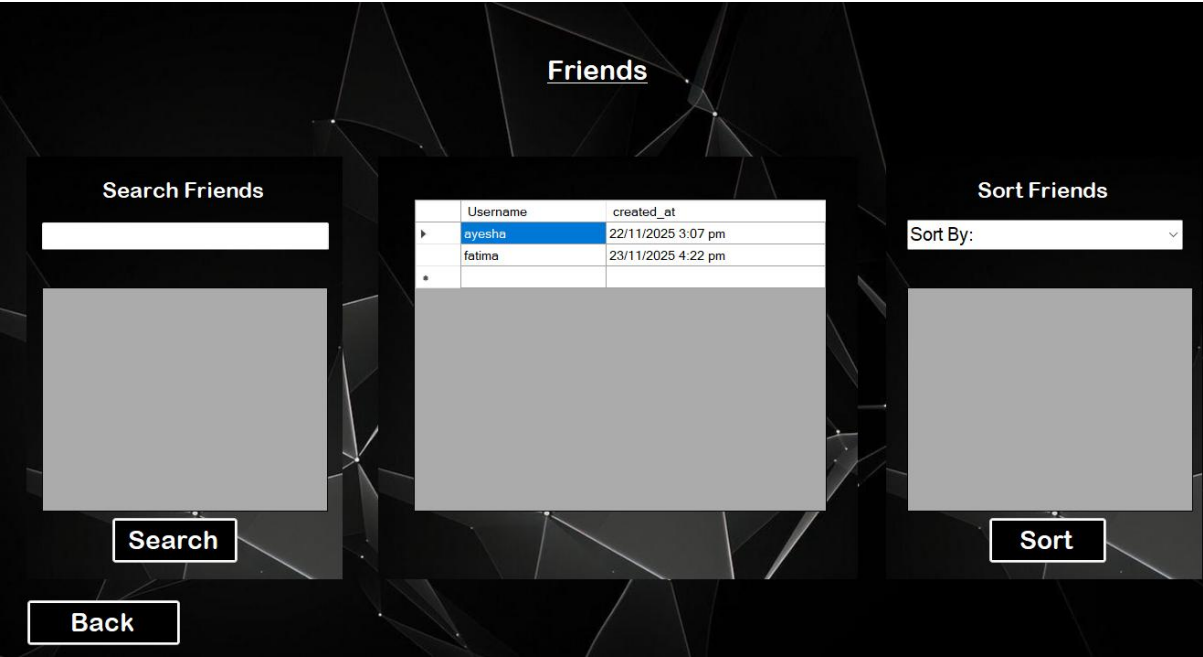
Back

Friends

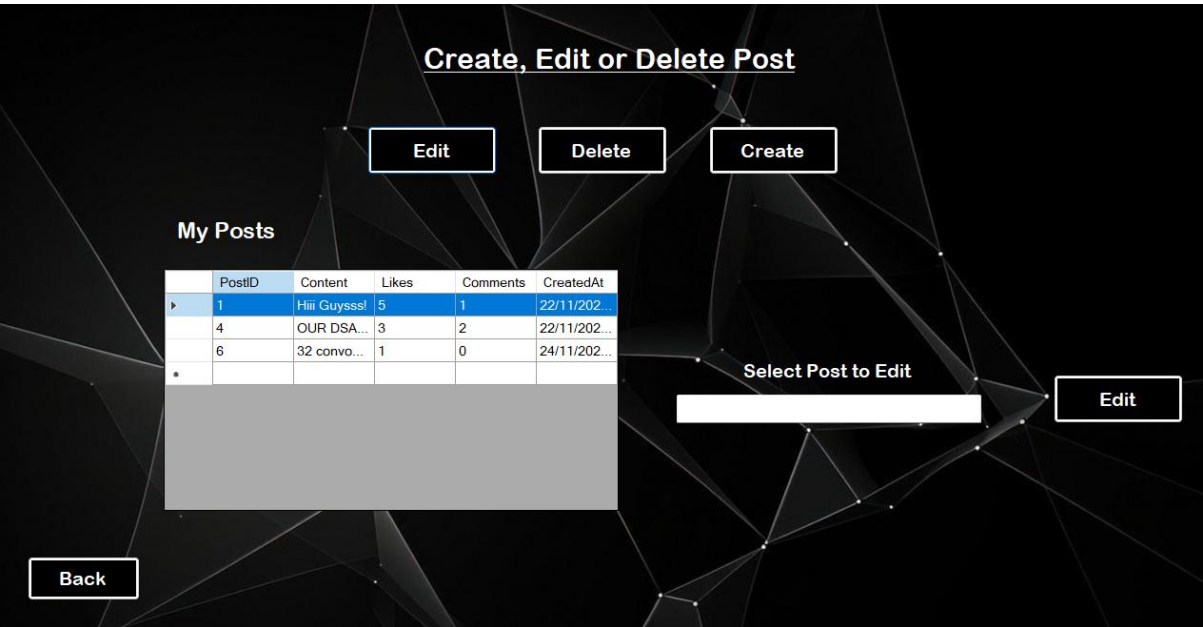
History

My Posts

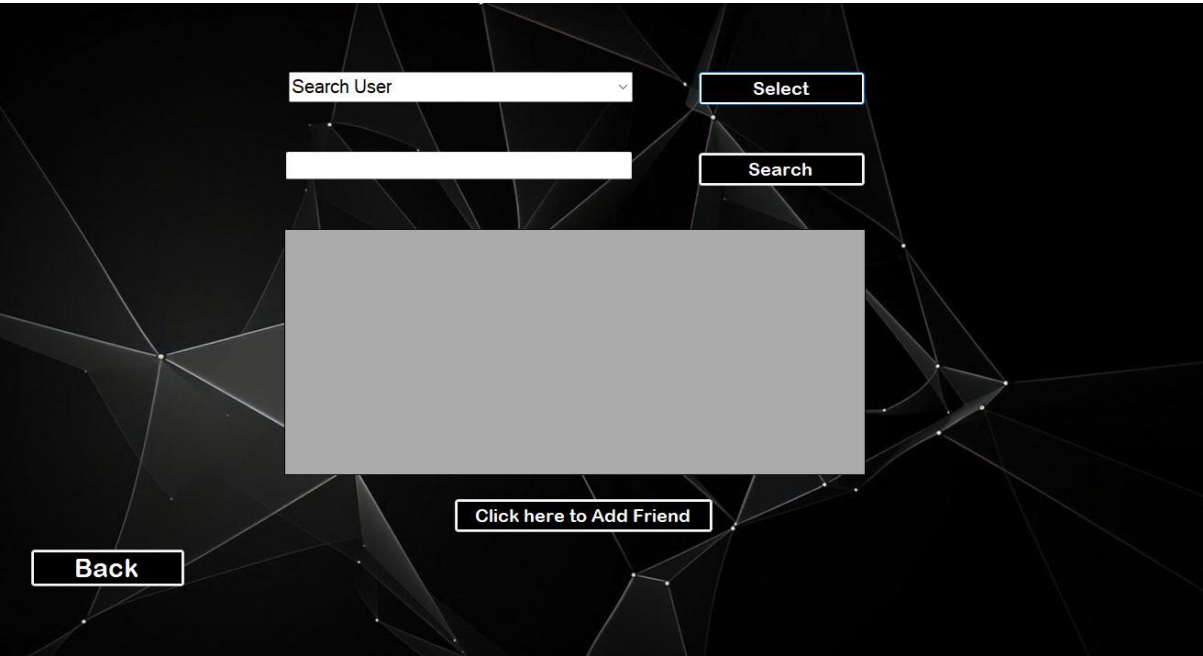
9.5. Friends Page



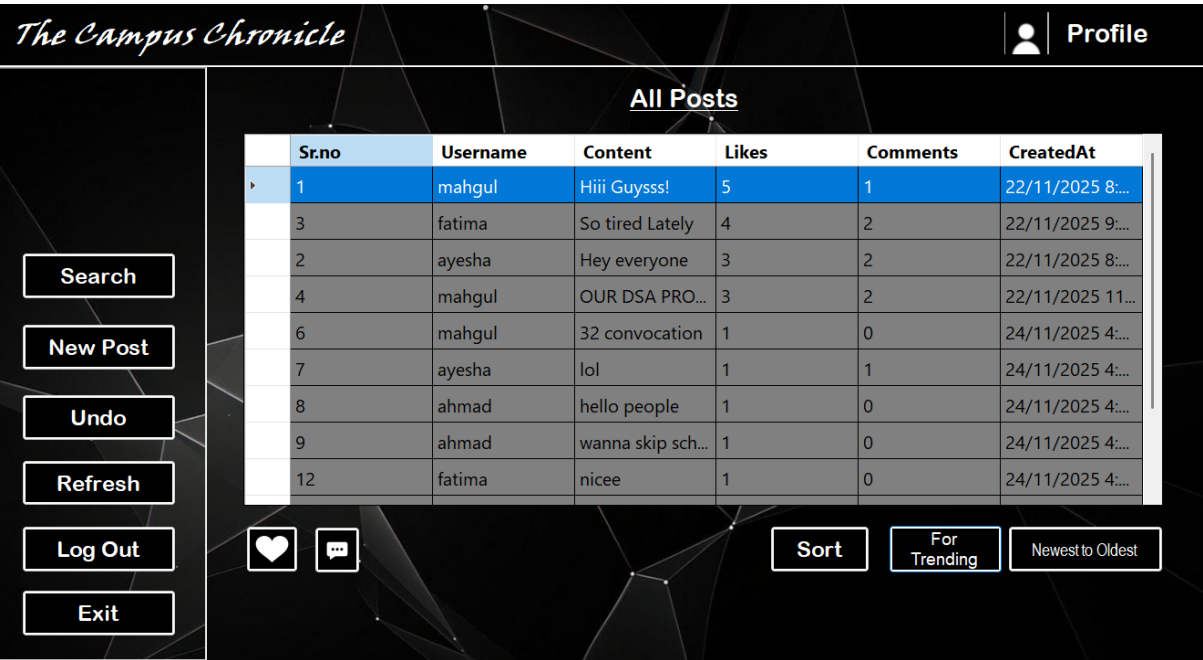
9.6. User Posts



9.7. Search Page



9.8. Sorting For Trending Posts



## 9.9. Commenting on Posts

**The Campus Chronicle** Profile

**All Posts**

| Sr.no | Username | Content           | Likes | Comments | CreatedAt        |
|-------|----------|-------------------|-------|----------|------------------|
| 1     | mahgul   | Hiii Guysss!      | 5     | 1        | 22/11/2025 8:... |
| 3     | fatima   | So tired Lately   | 4     | 2        | 22/11/2025 9:... |
| 2     | ayesha   | Hey everyone      | 3     | 2        | 22/11/2025 8:... |
| 4     | mahgul   | OUR DSA PRO...    | 3     | 2        | 22/11/2025 11... |
| 6     | mahgul   | 32 convocation    | 1     | 0        | 24/11/2025 4:... |
| 7     | ayesha   | lol               | 1     | 1        | 24/11/2025 4:... |
| 8     | ahmad    | hello people      | 1     | 0        | 24/11/2025 4:... |
| 9     | ahmad    | wanna skip sch... | 1     | 0        | 24/11/2025 4:... |
| 12    | fatima   | nicee             | 1     | 0        | 24/11/2025 4:... |

Search New Post Undo Refresh Log Out Exit

Heart Comment [Input Field] Add Sort For Trending Newest to Oldest

## 10. Performance Analysis:

### 9.1. Time Complexity

| Operation             | Data Structure        | Time Complexity | Description                       |
|-----------------------|-----------------------|-----------------|-----------------------------------|
| User Login            | Linked List           | $O(n)$          | Linear search by username         |
| Add Post              | Queue                 | $O(1)$          | Enqueue operation                 |
| Search Posts          | Queue + Linear Search | $O(n)$          | Iterate through all posts         |
| Sort Posts (Trending) | List + Bubble Sort    | $O(n^2)$        | Compare all elements by likes     |
| Sort Posts (Recent)   | List + Selection Sort | $O(n^2)$        | Find most recent posts repeatedly |
| Add Friend            | Linked List           | $O(1)$          | Insert at end                     |
| Navigation Push/Pop   | Stack                 | $O(1)$          | Constant time operations          |
| Undo Action           | Array                 | $O(1)$          | Access by index                   |

9.2. Space Complexity

| Component        | Space Complexity | Description            |
|------------------|------------------|------------------------|
| User List        | $O(n)$           | n users in linked list |
| Post Queue       | $O(m)$           | m posts in queue       |
| Comments         | $O(c)$           | c comments per post    |
| Friends List     | $O(f)$           | f friendships          |
| Navigation Stack | $O(s)$           | s navigation states    |
| Browsing History | $O(1)$           | Fixed-size array       |