

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018



**Mini Project Report
on**

SOCIAL NETWORKING APPLICATION

Submitted in partial fulfillment of the requirements as part of Data Structures Laboratory [BCSL305] for the III Semester degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belagavi

Submitted by

NAME-USN

CHETAN.N. JOSHI-1RN22IS029

HEMANTH T-1RN22IS050

MOKSH A N-1RN22IS191

Under the Guidance of

Aishwarya G

Assistant Professor

Department of ISE



ESTD:2001

An Institute with a Difference

Department of Information Science and Engineering

RNS Institute of Technology

Dr. Vishnuvardhana Road, Rajarajeshwari Nagar post,

Channasandra, Bengaluru-560098

2023-24

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvardhana Road, Rajarajeshwari Nagar post,

Channasandra, Bengaluru - 560098

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING



CERTIFICATE

Certified that the project work titled **“Social Networking Application”** has been successfully completed by **CHETAN.N.JOSHI (s) (1RN22IS029),HEMANTH.T(s) (1RNN22IS050),MOKSH.A.N (s) (1RN22IS191)** bonafide students of **RNS Institute of Technology, Bengaluru** in partial fulfillment of the requirements as part of **Data Structures Laboratory [BCSL305]** for the award of degree in **Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University, Belagavi** during the academic year **2023-2024**.

ACKNOWLEDGMENT

The fulfilment and joy associated with the successful completion of any assignment would be incomplete without acknowledging the individuals who made it possible. Their consistent guidance and support have crowned our efforts with success.

We would like to express my profound gratitude to the **Management of RNS Institute of Technology** for providing a conducive environment to conduct the activities of the SCR course.

We would like to express our gratitude to our Director **Dr. M K Venkatesha**, and our Principal **Dr. Ramesh Babu H S** for their support and inspiration towards the pursuit of knowledge.

We wish to express our heartfelt gratitude to **Dr. Suresh L**, Professor and Head of the Department of Information Science and Engineering, for being the driving force and mastermind behind our project.

We extend our heartfelt thanks to **Aishwarya G**, Assistant Professor in the Department of Information Science and Engineering, for guiding and evaluating the project.

We would like to thank all teaching and non-teaching staff of Information Science & Engineering department, who have directly or indirectly helped us to carry out the project.

Place: Bengaluru

CHETAN.N. JOSHI

1RN22IS029

Date:

HEMANTH.T

1RN22IS050

MOKSH.A. N

1RN22IS191

Table of Contents

ABSTRACT.....	i
CHAPTER 1: INTRODUCTION	1
1. Problem Definition.....	1
2. Motivation	1
3. Objectives.....	2
CHAPTER 2: DATA STRUCTURE USED	2
2.1 Introduction	
2.2 Operations	
2.3 Advantages and Limitations	
2.4 Applications	
CHAPTER 3: SYSTEM REQUIREMENT SPECIFICATIONS.....	4
CHAPTER 4: DESIGN	5
1. Flowchart	5
2. Use-Case Diagrams	5
3. Algorithms	
CHAPTER 5: IMPLEMENTATION.....	6
CHAPTER 6: RESULTS	7
CHAPTER 7: CONCLUSION.....	9
CHAPTER 8: REFERENCES	

ABSTRACT

The Social Networking project implemented in c leverages the power of linked list data structure to provide the connection of users.

In this project, we address to build a connection between the user and also sharing the post and finding mutual friends.

A linked list is a fundamental data structure used in computer science to store and organize data. Unlike arrays, linked lists do not require contiguous memory allocation, allowing for dynamic memory allocation and efficient insertion and deletion operations.

CHAPTER 1: INTRODUCTION

In today's interconnected world, social media platforms play a pivotal role in facilitating communication, networking, and information sharing among individuals. With the increasing popularity and reliance on social media, there arises a need for efficient and scalable networking solutions. This project endeavors to address this need by developing a social media networking application using the C programming language and implementing a single linked list data structure.

Problem Definition:

Existing social media platforms often rely on complex data structures and algorithms to manage user connections, posts, and interactions. However, implementing such systems can be challenging, especially for individuals seeking to understand fundamental data structures and their practical applications. The project aims to simplify this process by utilizing a single linked list data structure to manage user connections and posts within the social media network.

Motivation:

The motivation behind this project stems from the desire to bridge the gap between theoretical knowledge of data structures and their practical implementation in real-world applications. By developing a social media networking application using a single linked list, the project aims to provide a hands-on learning experience for individuals interested in understanding data structures and their role in software development. Additionally, the project aims to create a simplified social media platform that demonstrates the core functionalities of larger-scale networks while maintaining ease of comprehension and implementation.

Objectives:

The primary objectives of the project include:

Implementation of a social media networking application in C programming language.

Utilization of a single linked list data structure to manage user connections and posts.

Development of basic functionalities such as adding friends, posting updates, and viewing posts. Demonstration of the efficiency and scalability of the single linked list data structure in managing social media interactions.

Provision of a platform for learning and experimentation with data structures in a practical software development context.

By achieving these objectives, the project endeavors to provide a foundational understanding of data structures while offering insights into the development of social media networking applications.

Data Structure Used:

A single linked list is a fundamental linear data structure used in computer science and programming. It consists of a sequence of elements called nodes, where each node contains both data and a reference (or link) to the next node in the sequence. The first node is typically referred to as the head of the list, and the last node points to NULL, indicating the end of the list. Each node contains both data and a reference to the next node in the sequence, forming a unidirectional chain. The first node, often called the head, serves as the starting point for traversing the list. Traversal involves following the next pointers from one node to the next until reaching the end, where the last node's pointer typically points to NULL. Single linked lists offer dynamic memory allocation, allowing for efficient insertion and deletion operations without the need for contiguous memory allocation. Overall, single linked lists provide a versatile and efficient means of organizing and managing data in computer programs, offering flexibility and scalability for a wide range of applications and scenarios. Understanding the principles and operations of single linked lists is crucial for building a strong foundation in data structures and algorithms.

Operations on Single Linked List:

1. Insertion: Nodes can be inserted at the beginning, middle, or end of the linked list by adjusting the references appropriately.
2. Deletion: Nodes can be removed from the list by updating the references to skip over the deleted node.
3. Traversal: The linked list can be traversed sequentially, allowing access to each node for reading or modification.
4. Search: Nodes can be searched based on their data values to locate specific elements within the list.

Advantages of Single Linked List:

- Dynamic Size: Single linked lists can grow or shrink dynamically during program execution, allowing for efficient memory allocation.
- Insertion and Deletion: Insertion and deletion operations can be performed quickly at the beginning or end of the list without requiring elements to be shifted.
- Memory Efficiency: Single linked lists only require memory for storing data and the reference to the next node, making them memory-efficient compared to arrays or other data structures.
- Flexibility: Linked lists provide flexibility in terms of element insertion, deletion, and rearrangement, making them suitable for various applications.

Disadvantages of Single Linked List:

- **No Random Access:** Unlike arrays, accessing elements in a linked list requires traversal from the head, which can result in slower access times for random access operations.
- **Extra Memory Overhead:** Linked lists require additional memory to store the references (pointers) to the next node, which can increase memory overhead compared to contiguous storage structures.
- **Traversal Overhead:** Traversing a linked list can be less efficient than accessing elements in an array, especially for large lists, due to the need to follow pointers sequentially.

Applications of Single Linked List:

- **Dynamic Memory Allocation:** Single linked lists are commonly used in dynamic memory allocation scenarios where the size of the data structure is not known beforehand.
- **Implementation of Stacks and Queues:** Linked lists serve as the underlying data structure for implementing stack and queue data structures due to their dynamic nature.
- **Symbol Tables and Hash Tables:** Linked lists are used in symbol tables and hash tables to handle collisions and store key-value pairs efficiently.
- **Polynomial Manipulation:** Single linked lists can represent polynomials efficiently, making them suitable for polynomial addition, subtraction, and multiplication algorithms.

In summary, single linked lists offer flexibility and efficiency for dynamic data storage and manipulation, making them suitable for a wide range of applications in computer science and programming. However, they also come with certain limitations, particularly in terms of access time and memory overhead, which should be considered when selecting the appropriate data structure for a given problem.

System Requirements:

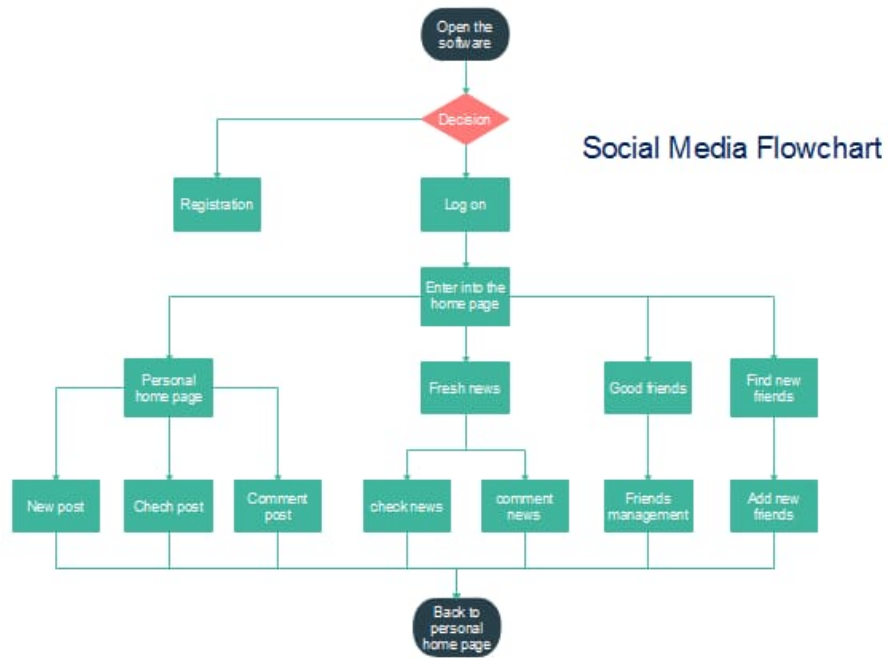
Windows 8 or above

RAM: 8 GB

Processor: i3 12th gen or higher

Software: Dev C++

Design:



IMPLEMENTATION

Implement User Management Functions: Develop functions to handle user registration, login, logout, profile editing, and deletion. Ensure that user passwords are securely stored and authenticated.

Implement Friendship Management Functions: Create functions to add friends, remove friends, and display a user's friend list. These functions will help users connect with each other within the social network.

Implement Posting Functions: Write functions to allow users to create posts, view posts from friends, delete their posts, and comment on others' posts.

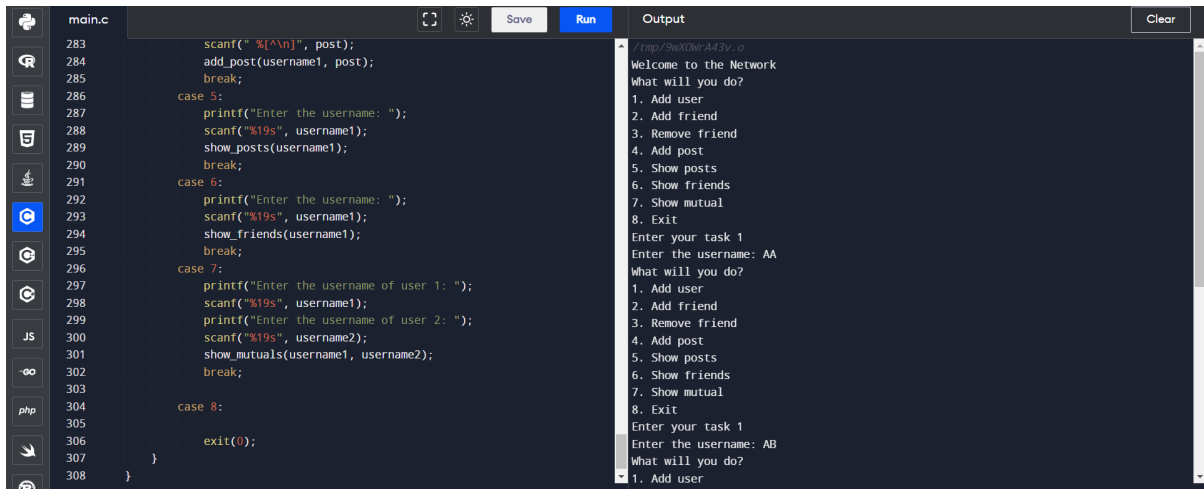
Implement User Interface : Depending on your project's requirements, you can create a command-line interface (CLI) or a simple graphical user interface (GUI) to interact with the social networking application.

Error Handling and Memory Management: Implement error handling mechanisms to deal with invalid input, memory allocation failures, and other runtime errors. Ensure that memory is managed efficiently to avoid memory leaks.

Testing and Debugging: Test each functionality thoroughly to ensure that it works as expected. Debug any errors or issues encountered during testing to improve the reliability and stability of the application.

Documentation: Document your code effectively, including comments and documentation ,strings to help other developers understand the purpose and functionality of each function and data structure.

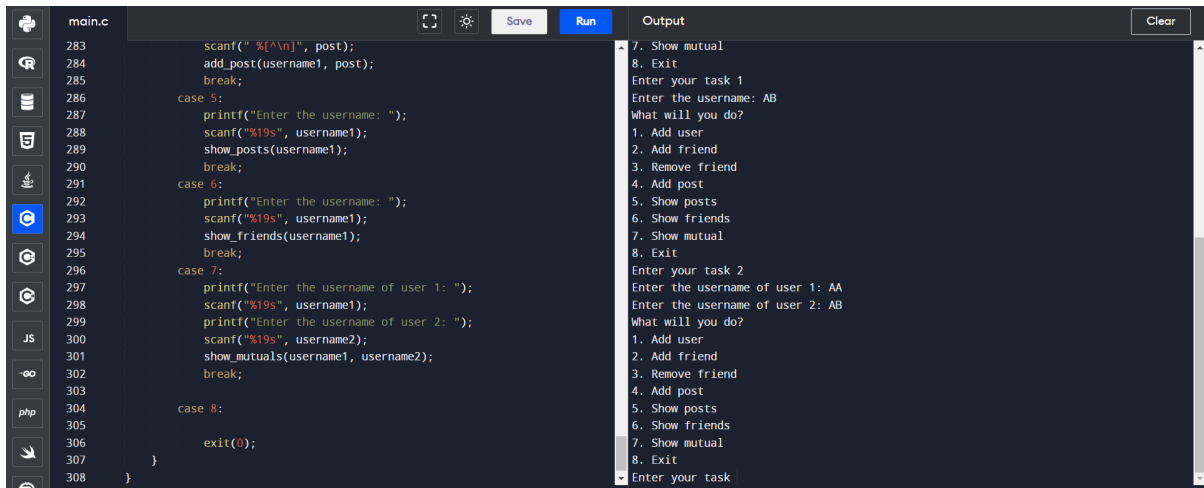
Result



```
main.c
283 scanf("%i\n", post);
284 add_post(username1, post);
285 break;
286 case 5:
287     printf("Enter the username: ");
288     scanf("%i9s", username1);
289     show_posts(username1);
290     break;
291 case 6:
292     printf("Enter the username: ");
293     scanf("%i9s", username1);
294     show_friends(username1);
295     break;
296 case 7:
297     printf("Enter the username of user 1: ");
298     scanf("%i9s", username1);
299     printf("Enter the username of user 2: ");
300     scanf("%i9s", username2);
301     show_mutuals(username1, username2);
302     break;
303
304 case 8:
305
306     exit(0);
307 }
308 }
```

Output

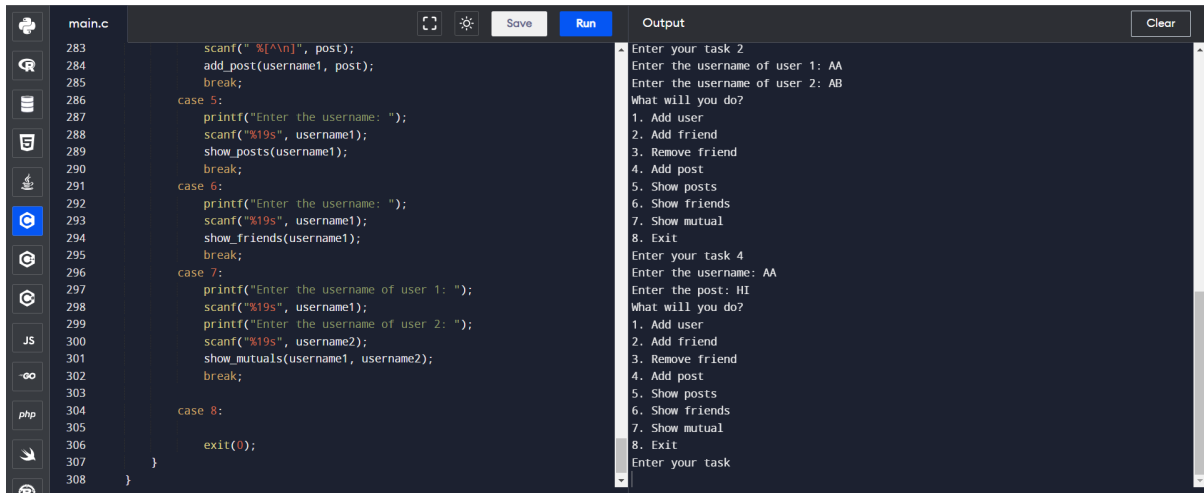
```
/tmp/3nX0NrA43v.o
Welcome to the Network
What will you do?
1. Add user
2. Add friend
3. Remove friend
4. Add post
5. Show posts
6. Show friends
7. Show mutual
8. Exit
Enter your task 1
Enter the username: AA
What will you do?
1. Add user
2. Add friend
3. Remove friend
4. Add post
5. Show posts
6. Show friends
7. Show mutual
8. Exit
Enter your task 1
Enter the username: AB
What will you do?
1. Add user
```



```
main.c
283 scanf("%i\n", post);
284 add_post(username1, post);
285 break;
286 case 5:
287     printf("Enter the username: ");
288     scanf("%i9s", username1);
289     show_posts(username1);
290     break;
291 case 6:
292     printf("Enter the username: ");
293     scanf("%i9s", username1);
294     show_friends(username1);
295     break;
296 case 7:
297     printf("Enter the username of user 1: ");
298     scanf("%i9s", username1);
299     printf("Enter the username of user 2: ");
300     scanf("%i9s", username2);
301     show_mutuals(username1, username2);
302     break;
303
304 case 8:
305
306     exit(0);
307 }
308 }
```

Output

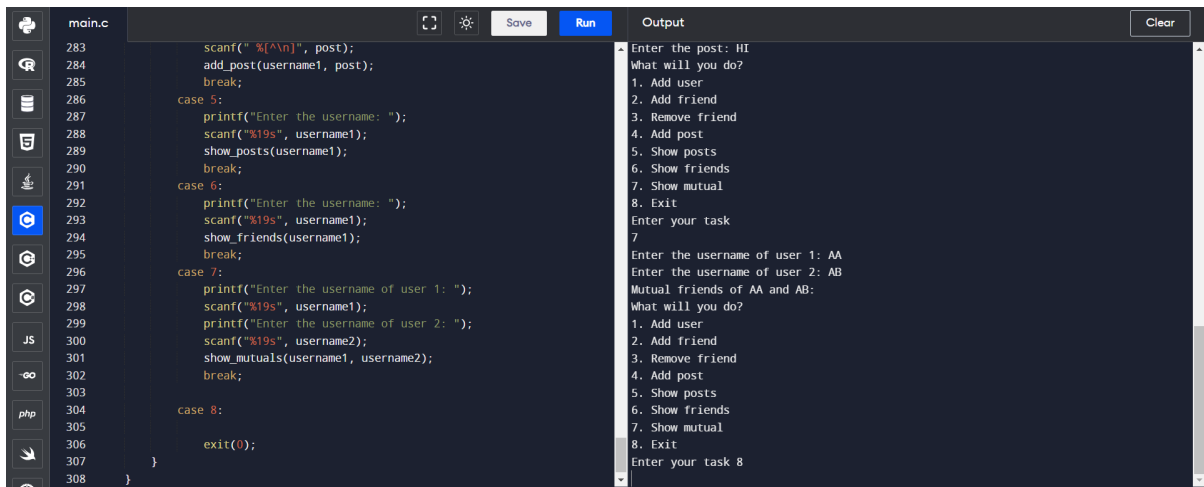
```
7. Show mutual
8. Exit
Enter your task 1
Enter the username: AB
What will you do?
1. Add user
2. Add friend
3. Remove friend
4. Add post
5. Show posts
6. Show friends
7. Show mutual
8. Exit
Enter your task 2
Enter the username of user 1: AA
Enter the username of user 2: AB
What will you do?
1. Add user
2. Add friend
3. Remove friend
4. Add post
5. Show posts
6. Show friends
7. Show mutual
8. Exit
Enter your task
```



```
main.c
283     scanf("%i\n", &post);
284     add_post(username1, post);
285     break;
286 case 5:
287     printf("Enter the username: ");
288     scanf("%i\n", &username1);
289     show_posts(username1);
290     break;
291 case 6:
292     printf("Enter the username: ");
293     scanf("%i\n", &username1);
294     show_friends(username1);
295     break;
296 case 7:
297     printf("Enter the username of user 1: ");
298     scanf("%i\n", &username1);
299     printf("Enter the username of user 2: ");
300     scanf("%i\n", &username2);
301     show_mutuals(username1, username2);
302     break;
303
304 case 8:
305
306     exit(0);
307 }
308 }
```

Output

Enter your task 2
Enter the username of user 1: AA
Enter the username of user 2: AB
What will you do?
1. Add user
2. Add friend
3. Remove friend
4. Add post
5. Show posts
6. Show friends
7. Show mutual
8. Exit
Enter your task 4
Enter the username: AA
Enter the post: HI
What will you do?
1. Add user
2. Add friend
3. Remove friend
4. Add post
5. Show posts
6. Show friends
7. Show mutual
8. Exit
Enter your task



```
main.c
283     scanf("%i\n", &post);
284     add_post(username1, post);
285     break;
286 case 5:
287     printf("Enter the username: ");
288     scanf("%i\n", &username1);
289     show_posts(username1);
290     break;
291 case 6:
292     printf("Enter the username: ");
293     scanf("%i\n", &username1);
294     show_friends(username1);
295     break;
296 case 7:
297     printf("Enter the username of user 1: ");
298     scanf("%i\n", &username1);
299     printf("Enter the username of user 2: ");
300     scanf("%i\n", &username2);
301     show_mutuals(username1, username2);
302     break;
303
304 case 8:
305
306     exit(0);
307 }
308 }
```

Output

Enter the post: HI
What will you do?
1. Add user
2. Add friend
3. Remove friend
4. Add post
5. Show posts
6. Show friends
7. Show mutual
8. Exit
Enter your task 7
Enter the username of user 1: AA
Enter the username of user 2: AB
Mutual friends of AA and AB:
What will you do?
1. Add user
2. Add friend
3. Remove friend
4. Add post
5. Show posts
6. Show friends
7. Show mutual
8. Exit
Enter your task 8

CONCLUSION:

In conclusion, the implementation of a mini-project social networking application using a single linked list data structure offers valuable insights into both data structures and practical software development. Throughout the project, several key aspects have been highlighted:

Understanding of Data Structures: Developing a social networking application using a single linked list deepens understanding of fundamental data structures. It provides hands-on experience in managing dynamic data, pointers, and memory allocation, which are essential concepts in computer science.

Efficiency and Scalability: Despite the simplicity of the single linked list, the social networking application demonstrates efficiency and scalability in managing user connections and posts. Insertion and deletion operations are streamlined, offering flexibility in adding or removing users and posts from the network.

Practical Applications: The project showcases the practical applications of single linked lists beyond theoretical concepts. By implementing real-world functionalities such as adding friends, posting updates, and viewing posts, users gain a tangible understanding of how data structures can be utilized in software development.

Limitations and Considerations: While single linked lists offer advantages in terms of dynamic memory allocation and simplicity, they also come with limitations. Lack of random access and potential inefficiencies in traversal may impact performance, especially in larger-scale applications. As such, careful consideration of data structure choice is crucial in designing robust and scalable systems.

Room for Improvement: As with any project, there are opportunities for enhancement and further development. Integrating additional features such as user authentication, messaging, and content recommendation algorithms could enrich the user experience and broaden the scope of the application.

In essence, the mini-project social networking application underscores the importance of bridging theoretical knowledge with practical implementation. By leveraging single linked lists as the underlying data structure, the project not only demonstrates the versatility and applicability of data structures but also fosters a deeper understanding of software design principles and development methodologies.

REFERENCE:

- <https://www.geeksforgeeks.org/>
- <https://www.github.com>
- <https://code-projects.org/>