Information Technology University (ITU)
Lahore.

Facebook with DSA Concepts

BS-Computer Engineering

Session: 2022-2026

Course Instructor: Dr. Tahira Mahboob

Submitted By:

Syed Kifayat Ur Rahman	BSCE22026
Muhammad Haseeb	BSCE22048

Department of Computer and Software Engineering,
Information Technology University (ITU), Lahore

Statement of Submission

This is to certify that **Syed Kifayat Ur Rahman** Roll No. **BSCE22026** and **Muhammad Haseeb** Roll No. **BSCE22026** have completed the DSA project named: **Social Media application like Facebook** to fulfill the partial requirement of the DSA**.**

_____

Course Instructor:

Name: Dr. Tahira Mahboob

Acknowledgment

We express our total submission and eternal gratitude to Almighty **ALLAH.** The Most Merciful, without whose mercy and grace, this endeavor could not be possible even its slightest. We bow before our compassionate endowments. Peace be upon the **Holy Prophet MUHAMMAD (S.A.W)** who is ever a torch of guidance and knowledge for humanity as a whole.

Our advisor **Dr. Tahira Mehboob**, is a lecturer of **DSA** at **ITU Lahore**, is someone we consider to be an honor and we sincerely thank her for her assistance and collaboration. She has been a constant source of guidance throughout this project.

Team Members:

1-Syed Kifayat Ur Rahman

2- Muhammad Haseeb

Date:

Abstract

This documentation takes you through the development of our C++ application, the "Social Media Application like Facebook". Our goal with this project is to create a project in which we use the core concept of DSA like graphs, link list etc.

Our application is your go-to solution for understanding the Facebook algorithm with ease. We've taken away the complexities involved in connecting friends with graph, posting posts and many other, so you can focus on what matters most.

Index

Table of Contents

Table of Contents

Chapter-1

"Introduction of Project"

1.1 Purpose

The primary aim of this documentation is to serve as a comprehensive guide for the development and functionality of a C++-based social media console application. It is designed to provide in-depth insights into the core concepts of data structures and algorithms used in creating a social media platform reminiscent of Facebook. This documentation is intended to benefit both the developers actively involved in its creation and those seeking to understand and maintain the application.

1.2 Scope

This documentation covers an extensive scope, encompassing critical phases of the social media console application's development lifecycle. It begins from the conceptualization of the project, delves into the implementation of data structures and algorithms pertinent to a Facebook-like algorithm, and extends through the iterative stages of coding, testing, and refining the application. Each phase is meticulously detailed to provide a comprehensive understanding of how these core concepts are integrated into the application, fostering a deeper insight into its functionality and structure.

1.3 Audience

This documentation is crafted to cater to a diverse audience, including but not limited to:

1.3.1 Development Team:

This document serves as a valuable resource for the development team responsible for building and maintaining the website. It offers insights into the project's requirements, design, and technical details, aiding them in their tasks.

1.3.2 Maintenance and Support Teams:

Individuals entrusted with the continual maintenance and support of the social media console application can rely on this documentation for troubleshooting procedures, executing updates, and ensuring the seamless operation of the application.

1.3.3 **End Users**:

Although primarily tailored for development and maintenance stakeholders, this documentation also serves as a helpful reference for end users seeking insights into the functionalities and capabilities embedded within the social media console application. It offers a glimpse into the underlying mechanisms driving their user experience.

Chapter-2

"Requirement Gathering"

2.1 Functional Requirements:

2.1.1 Creating Account:

Users can register for a new account by providing essential information such as username, email, and password. Upon successful registration, the system securely stores user credentials for subsequent logins.

2.1.2 Show Feeds:

Upon logging in, users are presented with a personalized feed displaying posts, updates, and activities from their friends or followed accounts. The feed is generated based on user interactions, preferences, and algorithmic recommendations.

2.1.3 Add Friends:

Users can send friend requests to other users, allowing them to connect and view each other's posts and updates. The recipient can either accept or decline the friend request.

2.1.4 Seen Friend Request:

Users are notified when their friend request has been seen by the recipient. This feature ensures transparency in the friend request process, indicating whether the recipient has viewed the request.

2.1.5 Add Posts:
Users can create and share posts consisting of text, images, videos, or links. They can customize the audience for each post, choosing to share it publicly, with friends, or specific groups.

2.1.6 Friend Suggestion:
The platform provides friend suggestions based on mutual friends, shared interests, or similar user interactions. These suggestions assist users in expanding their network within the platform.

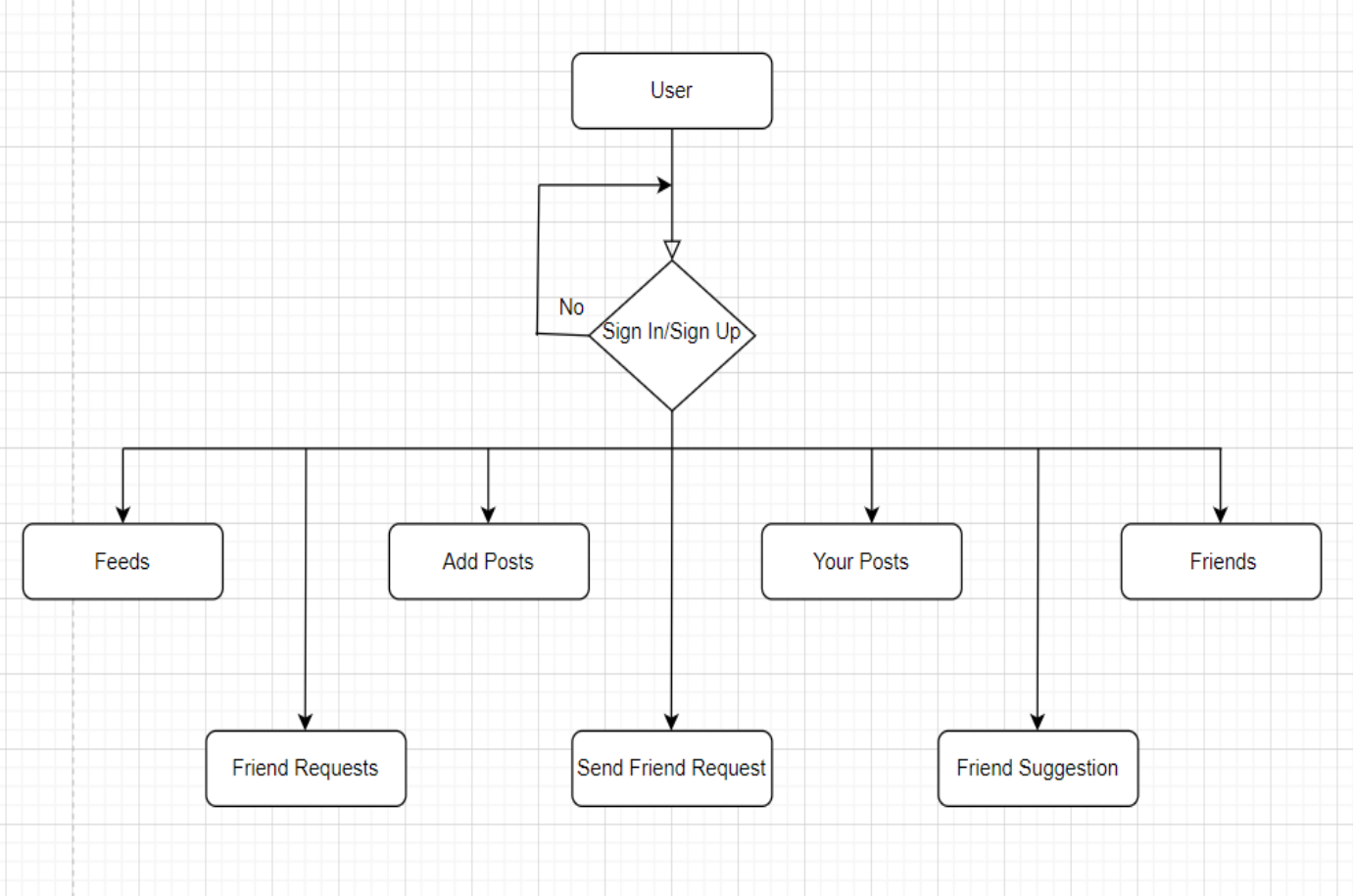
2.1.7 View Self Posts:
Users have access to a profile section where they can view all their own posts, allowing them to review their activities and interactions within the platform.

2.2 Non-functional Requirements:

2.2.1 Authentication:

User authentication is a pivotal aspect of the application's security. It employs strong authentication mechanisms, including password hashing and encryption, to verify user identities during login.

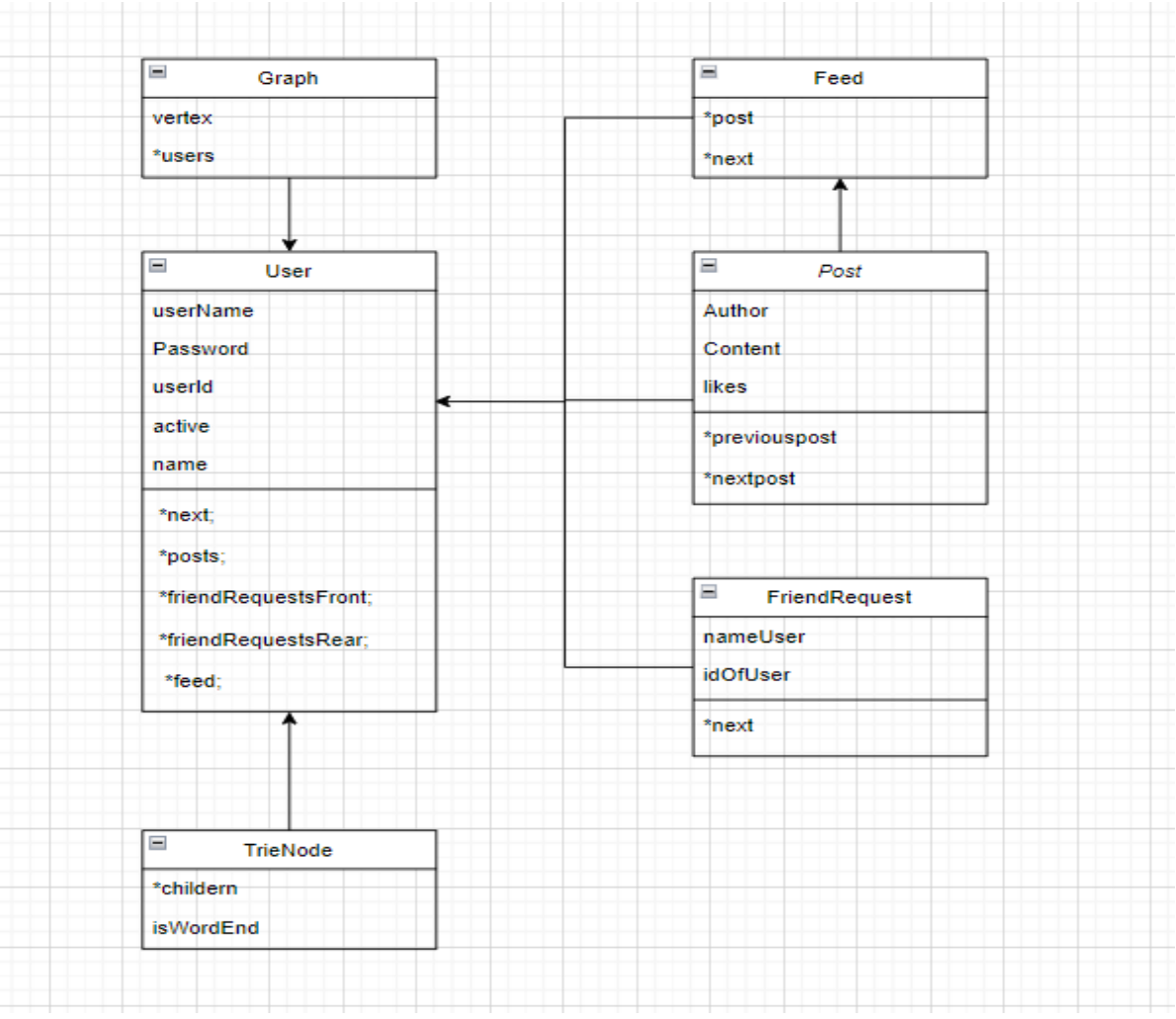
2.3 Use-Case Diagram



Chapter-3
"Design of Facebook"

3.1 Information Architecture

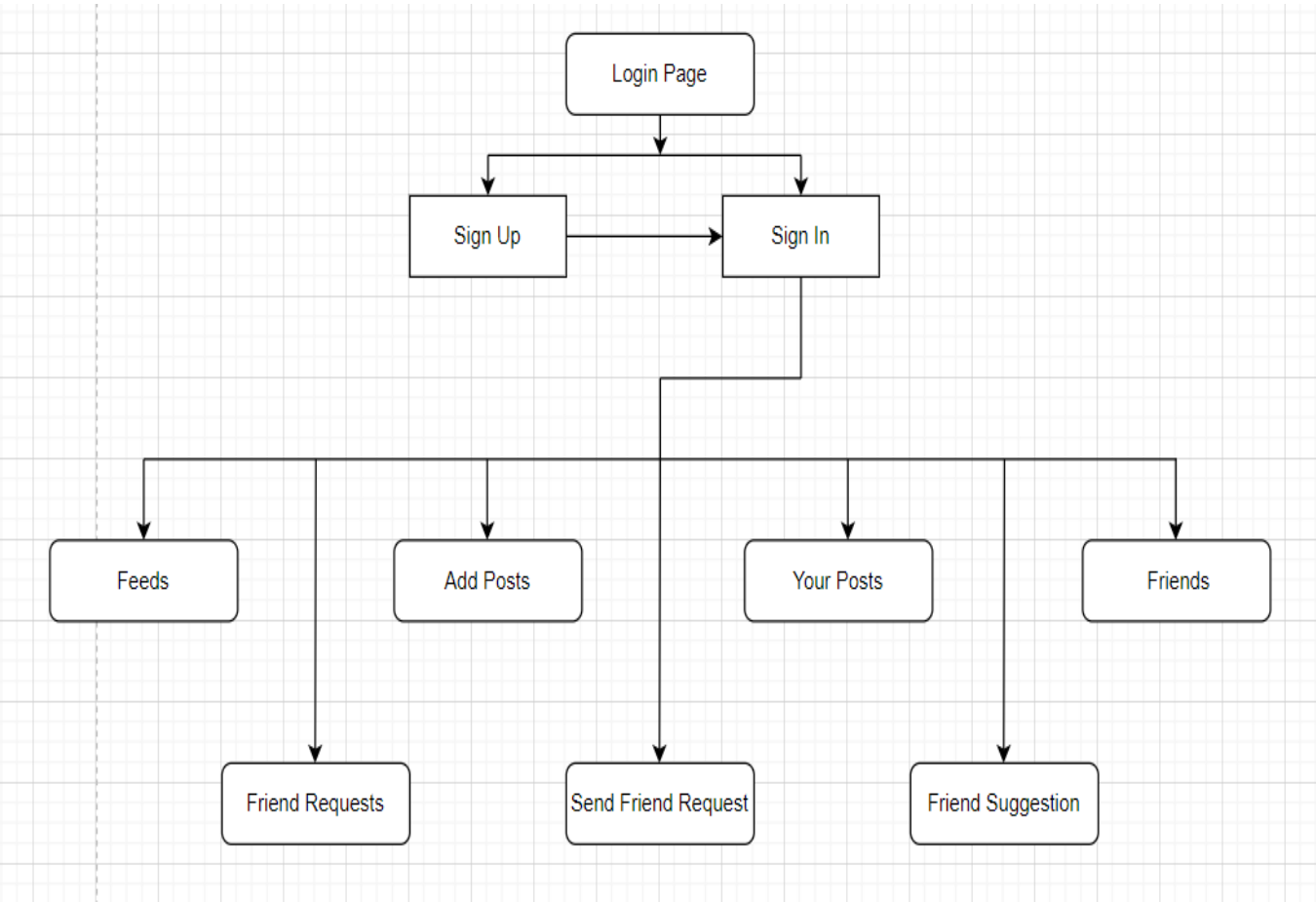
3.1.1 **Class Diagram:**



3.1.2 **Architecture Diagram**:

An architecture diagram is a graphical representation of a set of concepts that are part of architecture, including their principles, elements and components.

1. User:



3.2 Wireframes and Mockups

3.2.1 **Wireframes:** The wireframes provided serve as foundational visual guides detailing the layout and initial content placement within the application's interface. These wireframes outline the structural framework of each screen, aiding developers and designers in comprehending the intended arrangement of elements and the flow of user interactions.

3.2.2 **Mockups:** Attached mockups offer a more comprehensive representation of the application's pages, encompassing design elements such as colors, typography, and visual aesthetics. These detailed mockups provide a tangible visualization of the application's intended user interface, offering a clear depiction of its overall look and feel. They serve as a reference for creating the final design, ensuring consistency and coherence in the application's visual presentation.

3.4 Filing for Data Storage

3.4.1 **Entity-Relationship Diagram (ERD):** Attached is an Entity-Relationship Diagram (ERD) illustrating the database's structure. The ERD showcases the interconnectedness of key entities within the application, encompassing users, posts, comments, friendships, and other relevant data entities. This visual representation delineates the entities, attributes associated with each entity, and the relationships between them, aiding in a comprehensive understanding of the database's architecture.

3.4.2 **Schema:** The documentation encapsulates a detailed schema elucidating the database structure. It provides comprehensive information on the various tables, fields contained within each table, specified data types, and any applied constraints. This detailed schema documentation is vital for developers and database administrators, furnishing them with a clear blueprint of the database's organization. It facilitates efficient data management, optimization, and maintenance strategies, ensuring the integrity and optimal performance of the application's data storage.

Chapter-4

"Technology Stack"

4.1 Code Editor

4.1.1 **VS Code:** We are using vs code as code editor. It's a versatile and lightweight code editor that's widely used for a variety of programming languages and development tasks.

4.2 Coding Language

4.2.1 **C++:** For the implementation of Data Structures and Algorithms (DSA), C++ stands as the primary coding language. Known for its efficiency and versatility, C++ serves as the backbone for handling complex data structures and algorithmic implementations. Its robustness and low-level functionalities empower developers to create optimized solutions for a wide array of DSA-related challenges.

Utilizing C++ allows for the creation of efficient algorithms and data structures, leveraging its powerful features such as pointers, memory management, and rich standard libraries. With its focus on performance and flexibility, C++

becomes an excellent choice for executing intricate DSA concepts, ensuring optimal resource utilization and computational efficiency.

Chapter-5

"Development"

5.1 Project Structure

Our social media console application project employs a structured organization methodology, promoting clarity and ease of collaboration among team members. The core elements of our project structure encompass:

5.1.1 **Directory Hierarchy:** The project is structured with well-defined directories for front-end and back-end components, ensuring a clear segregation of functionalities and responsibilities within the codebase. This separation enhances readability and facilitates efficient development and maintenance.

5.1.2 **Templates and Pages:** A well-organized layout for templates and pages is maintained, enabling streamlined content updates and simplified creation of new pages. This structured approach ensures consistency and ease in managing the application's visual components.

5.1.3 **Configurations:** Configuration files and environment settings are thoughtfully organized, ensuring accessibility and ease of modification. This structured arrangement allows for efficient configuration management, simplifying the handling of various settings within the application.

5.2 Coding Standards

To ensure code quality, readability, and maintainability within our C++ implementation of Data Structures and Algorithms for a social media console application, we adhere to a set of coding standards:

5.2.1 **Code Formatting:** We strictly adhere to specific code formatting guidelines, encompassing indentation, line length limitations, and consistent code structure. This standardization enhances code readability and maintains a uniform coding style across the project.

5.2.2 **Naming Conventions:** All variables, functions, and classes follow a consistent naming convention that accurately reflects their purpose and functionality. Descriptive and meaningful names are chosen to improve code comprehension and maintain consistency throughout the codebase.

5.2.3 **Commenting and Documentation:** Comprehensive comments and documentation are embedded within the codebase, explaining the rationale behind algorithms, complex data structures, and crucial sections of the code. This documentation aids developers in understanding the code's logic and functionality, enhancing maintainability and future development efforts.

5.2.4 **Meaningful Naming:** Variable and function names are carefully chosen to be self-explanatory, fostering code readability and reducing the need for additional comments to explain their purpose. Intuitive naming conventions enhance code clarity and comprehension.

5.2.5 **Modularity and Reusability:** The codebase is structured with a focus on modularity and reusability. Algorithms, data structures, and functionalities are designed to be modular, allowing for easy integration,

maintenance, and reuse across different parts of the application. This approach promotes efficiency in development and ensures scalability of the project.

5.3 Implementation Details:

The "Implementation Details" section delves deeply into the execution of various functionalities within our C++ implementation of Facebook-like features. It encompasses:

5.3.1 **Feature Implementation:** Comprehensive explanations detailing the implementation of core features such as user authentication, post creation, news feed algorithms, friend/follow functionalities, messaging systems, and content moderation.

5.3.2 **Code Snippets:** Included code snippets and examples elucidate specific implementation details, offering clear insights into the application of data structures (e.g., graphs, trees) and algorithms (e.g., sorting, searching) pertinent to each feature's implementation.

5.3.3 **Dependencies and Libraries:** Documentation outlining the external libraries, if any, and dependencies utilized in the C++ environment for implementing the Facebook-like functionalities. This includes frameworks or libraries leveraged for optimized data structure handling and algorithmic implementations.

5.4 Version Control

Version control plays a pivotal role in maintaining stability and facilitating collaboration within our C++ implementation of Facebook-like features. We employ a version control system, such as Git, to manage our codebase. This encompasses:

5.4.1 **Branching Strategy:** Our branching strategy delineates the management of feature branches, bug fixes, and releases within the C++ implementation. It outlines procedures for creating, merging, and handling different branches, ensuring an organized development workflow.

5.4.2 **Commit Messages:** Adhering to best practices, we emphasize the importance of writing descriptive and informative commit messages. These messages succinctly describe the purpose and scope of each code change, aiding in comprehending the evolution of the codebase over time.

5.4.3 **Merge and Pull Request Guidelines:** We have established guidelines for code reviews, merges, and pull requests, ensuring a systematic and effective review process. These guidelines facilitate collaborative code evaluation, ensuring quality assurance before integrating changes into the main codebase.

5.4.4 **Continuous Integration and Deployment (CI/CD):** Integration of Continuous Integration and Deployment practices automates testing and deployment procedures. This integration ensures consistent code quality through automated testing and enables rapid updates and deployment cycles for efficient development.

5.4.5 **Repository Access and Permissions:** Access to our repository and corresponding permissions are clearly defined and managed. This delineation ensures that only authorized personnel have access to specific sections of the codebase, maintaining security and control over the project's integrity.

Chapter-6

"Features and Functionality"

6.1 Login System

The login system serves as the gateway for user access to the application and encompasses both the sign-up and sign-in options:

6.1.1 **Sign-Up Option:** Allows new users to create accounts by providing essential information such as username, email, password, and any additional required details. This feature facilitates the registration process, enabling new users to join the platform.

6.1.2 **Sign-In Option:** Enables existing users to securely log into their accounts using their credentials, typically username or email and password. This functionality grants access to the user's personalized content, social interactions, and other application features.

The login system ensures a secure and streamlined process for both new users seeking to join the platform and existing users aiming to access their accounts, fostering user engagement and interaction within the social media console application.

6.2 Home Page

Upon successful login, users are directed to the home page, tailored to their preferences and interactions within the application:

6.2.1 **Show Feeds:** The home page displays a dynamic feed, presenting a stream of posts, updates, and activities from users the individual is connected with. This feed algorithmically populates content based on user interactions, preferences, and relationships.

6.2.2 **Add Friends:** A dedicated section allows users to discover and connect with new friends. It enables users to expand their social network.

6.2.3 **Seen Friend Request:** Notifications or a designated section informs users about pending friend requests that they have seen but not yet accepted or declined, ensuring transparency in the friend request process.

6.2.4 **Add Post:** Users can create and share posts directly from the home page. This feature allows for the composition and publication of text or links, engaging with their social circle.

6.2.5 **View Self Posts:** A dedicated section or profile snapshot allows users to quickly view their own posts and interactions within the platform, providing easy access to their personal content and activities.

6.2.6 **Friend Suggestions:** Continuous suggestions of potential friends or connections based on mutual friends, common interests, or similar interactions within the platform, aiding users in expanding their social network.

Chapter-7

"Algorithm Overview"

7.1 User Management

7.1.1 User Class: Define a class to store user details (name, ID, password, etc.).

7.1.2 Graph Data Structure: Use a graph to manage user relationships.

- Each node represents a user.
- Edges denote friendships between users.
- Adjacency lists can store friends' information.

7.2 Authentication and Authorization

7.2.1 Implement user authentication:

- Compare entered login details with stored user credentials.

7.2.2 Authorization: Define user roles and permissions if needed.

7.3 Friend Requests

- Store friend requests using queues or arrays.
- Implement functions to send, receive, and manage friend requests.
- Manage pending requests and allow users to accept or reject them.

7.4 Posts and Feeds

- Each user has a list of posts they've created.
- Create a feed for each user showing their friends' recent posts.
- Implement functions to add, delete, and view posts.
- Update feeds when new posts are added.

7.5 Suggestions

- Implement a friend suggestion algorithm based on mutual connections.

7.6 Implementation Using Data Structures

7.6.1 Graph Representation

- Use an adjacency list or adjacency matrix to represent friendships.
- Store users in an array or hash map for quick access.

7.6.2 Trie Data Structure for Auto-Complete (if needed)

- Use a Trie to implement auto-complete functionalities like searching for users, names, etc.

7.6.3 Queue/LinkedList for Friend Requests

- Store pending friend requests using a queue or linked list.

7.6.4 Post Management

- Doubly linked lists or arrays to manage user posts.
- Stacks or linked lists for managing feed updates.

7.7 Algorithm Details

7.7.1 User Registration and Login

- Read user data from files or databases during initialization.
- Authenticate users during login using stored credentials.

7.7.2 Friend Requests

- Allow users to send and accept/reject friend requests.
- Update the graph structure accordingly.

7.7.3 Post Creation and Feeds

- Enable users to create posts, manage their posts, and view their feeds.
- Update feeds based on friends' activities.

7.7.4 Friend Suggestions

- Implement a suggestion algorithm based on mutual friends or common interests.

7.7.5 Auto-Complete for Searching

- Utilize a Trie data structure for efficient auto-completion during user search.

7.7.6 File Handling

- Read and write data to files for user details, posts, friend requests, etc.

7.8 Conclusion

This algorithm provides a general framework for implementing a basic Facebook-like system using C++ and DSA concepts. Actual implementation details may vary based on specific requirements, optimizations, and additional features.

Conclusion

- Summary of Achievements

Successful Development and Deployment: Implementation of core functionalities resembling a social media platform using C++ and DSA concepts, achieving a robust and functional application.

Positive User Engagement: Received positive feedback and substantial user engagement, reflecting user satisfaction and interest in the application's features and interactions.

These achievements underscore the successful development and deployment of a social media console application resembling Facebook, utilizing C++ and leveraging Data Structures and Algorithms to create a functional, compliant, and engaging platform.

- Challenges Faced

1. Technical Challenges during Development:

Issue: Implementation complexities arose while integrating complex Data Structures and Algorithms (DSA) within the application's functionalities.

Resolution: Addressed technical hurdles by conducting thorough research, refactoring code structures, and optimizing algorithms for better performance. Collaboration among the development team facilitated effective problem-solving and implementation strategies.

These challenges were met with proactive strategies, collaborative efforts, and a structured approach to problem-solving, enabling the project to navigate technical complexities, compliance changes, and user-centric development effectively.

- Future Enhancements

1. Expanding Features and Functionalities:

Vision: Enrich the application's features and functionalities to enhance user engagement and interactions.

Approach: Introduce new features such as real-time chat, multimedia sharing enhancements (videos, GIFs), event creation and management, or enhanced privacy settings to provide users with more diverse and engaging experiences.

2. Integration of Advanced Algorithms:

Vision: Implement advanced algorithms to improve content personalization and recommendation systems.

Approach: Leverage sophisticated algorithms like machine learning-based content recommendation, sentiment analysis for personalized content feeds, or graph algorithms for friend suggestions to enhance user engagement and satisfaction.

3. Continuous Performance Optimization:

Vision: Maintain and improve the application's performance for a seamless user experience.

Approach: Implement performance optimization techniques, such as optimizing data structures and algorithms for faster execution, caching mechanisms for quicker data retrieval, and minimizing server load for improved responsiveness.

These future enhancements aim to elevate the Facebook-like social media console application by expanding features, optimizing performance, leveraging advanced algorithms, and exploring innovative technologies to ensure a rich and engaging user experience while utilizing C++ and DSA concepts.