Project Overview:

The objective of this project is to develop a robust, multi-threaded chat application in the C language, facilitating communication among multiple clients through a central server. The emphasis is on implementing synchronization mechanisms to mitigate potential issues such as deadlock, starvation, and security vulnerabilities like brute-force attacks. The project documentation aims to provide a comprehensive understanding of design decisions, implementation details, and synchronization techniques employed within the application.

Objectives:

1. Design and Implementation of Chat Application using Pipes:

- Develop a client-server communication system using named pipes (FIFOs) in C.
- Efficiently manage multiple clients through thread management techniques within the C programming paradigm.

2. Synchronization Mechanisms with Pipes:

- Utilize synchronization mechanisms, such as semaphores or monitors, to prevent deadlock and starvation.
- Employ appropriate C language concurrency constructs to safeguard shared resources.

3. Security Measures with Pipes:

- Implement security measures to protect against brute-force attacks and unauthorized access.
- Ensure secure communication within the C programming environment.

Requirements:

Server:

- Server Creation with Named Pipes:
  - Develop a C server that listens for client connections using named pipes.
- Threaded Communication Handling:
  - Establish separate threads to handle communication with individual clients.
  - Utilize C language threading mechanisms for efficient thread management.
- Message Handling System with Pipes:
  - Implement a message handling system to receive and broadcast messages among connected clients using named pipes.
  - Maintain a list of connected clients for message recipient identification.
- Graceful Disconnection:
  - Handle client disconnections gracefully.
  - Implement appropriate error handling techniques in C.

Client:

- Client Creation with Named Pipes:
    - Develop a C client that connects to the server using named pipes for communication.
- Message Input Mechanism:
    - Implement a C-based message input mechanism for user message entry.
- Display Mechanism:
    - Display received messages in the user interface using appropriate C output mechanisms.
- Graceful Disconnection:
    - Handle disconnections from the server gracefully.
    - Implement error handling techniques in C.

Synchronization and Security:

- Synchronization Mechanisms with Pipes:
    - Use semaphores or monitors to prevent deadlock and starvation.
    - Apply appropriate C concurrency constructs for shared resource management using named pipes.
- Implementation Guidelines:
    - Develop a multi-threaded server with a thread pool for efficient client request handling.
    - Ensure synchronization mechanisms for shared resources, including thread-safe data structures for message queues using named pipes.
- Client-Server Communication with Pipes:
    - Implement a robust communication protocol for messages, commands, and user interactions using named pipes.
    - Establish connections between clients and the server using named pipes.
    - Implement error handling mechanisms for graceful degradation.
- Mutual Exclusion:
    - Protect critical code sections from data corruption using locks or synchronization mechanisms with named pipes.
    - Implement strategies to avoid race conditions and ensure data integrity.
- Deadlock Prevention:
    - Analyze and prevent deadlock situations by using deadlock prevention techniques with named pipes.
    - Consider employing deadlock detection algorithms or designing the application to avoid circular waits.
- Starvation Prevention:
    - Implement a fair scheduling mechanism to prevent client or thread starvation.
    - Consider priority-based scheduling or other techniques to ensure equitable resource allocation.
- Security Measures with Named Pipes:
    - Integrate user authentication and authorization mechanisms to prevent brute-force attacks and unauthorized access.

Documentation:

The documentation will be provided in a structured format, including:

- Code Walkthrough:
    - Detailed explanations of key functions, data structures, and synchronization techniques with named pipes.
- Individual Submission:
    - Each student submits their project code, documentation, and presentation independently.
    - Demonstrates understanding of concepts and effective application in C programming with named pipes.