CSE110: Object-Oriented Programming
Spring 24 Semester

Project Report
Java Chat Application

Course Code:	CSE110
Title :	Object Oriented Programming
Section :	11

Submitted by:

Shimul Kumar Shoishob
ID: 2023-3-60-157
Std-mail: 2023-3-60-157@std.ewuubd.edu
E-mail: shimulkumarshoishob@gmail.com
Department of Computer Science & Engineering
East West University

Submitted To:

Jesan Ahammed Ovi
Senior Lecturer
Department of Computer Science & Engineering

Table of Contents

Introduction

This project report provides a comprehensive overview of the development process, objectives, and outcomes of our Java chat application. Developed as a semester project for the CSE110 Object-Oriented Programming course, the Java chat application is a social communication tool built upon the principles of Object-Oriented Programming. The Chat Box Application project aims to create a real-time messaging platform for users to communicate with each other. The report outlines the **problem design**, **implementation strategies**, and reflections on individual contributions and teamwork.
Additionally, it discusses the limitations of the current application and proposes potential avenues for future enhancements.

Objective

The main objective of the Java chat application is to provide users with **a platform to communicate through each other**.

The core features are:

- Sign up ,
- Log in,
- Chat with every connected people,
- View Profile,
- View Online Users,
- One to one chat, (Working)
- Encrypted Chatting,(Working)

This project aims to demonstrate proficiency in object-oriented programming principles by implementing various classes, encapsulating data, and defining interactions between objects.

Methodology

The development process started with a systematic approach, beginning with problem analysis and design, followed by implementation.

Object-oriented principles such as:

- Encapsulation,
- Abstraction,
- Polymorphism,
- Association,
- Model-View-Controller (MVC)

were applied throughout the development process to ensure code modularity, reusability, and maintainability.

Java topics used in this project:

- Socket Programming
- Java Swing
- Multithreading
- File as database

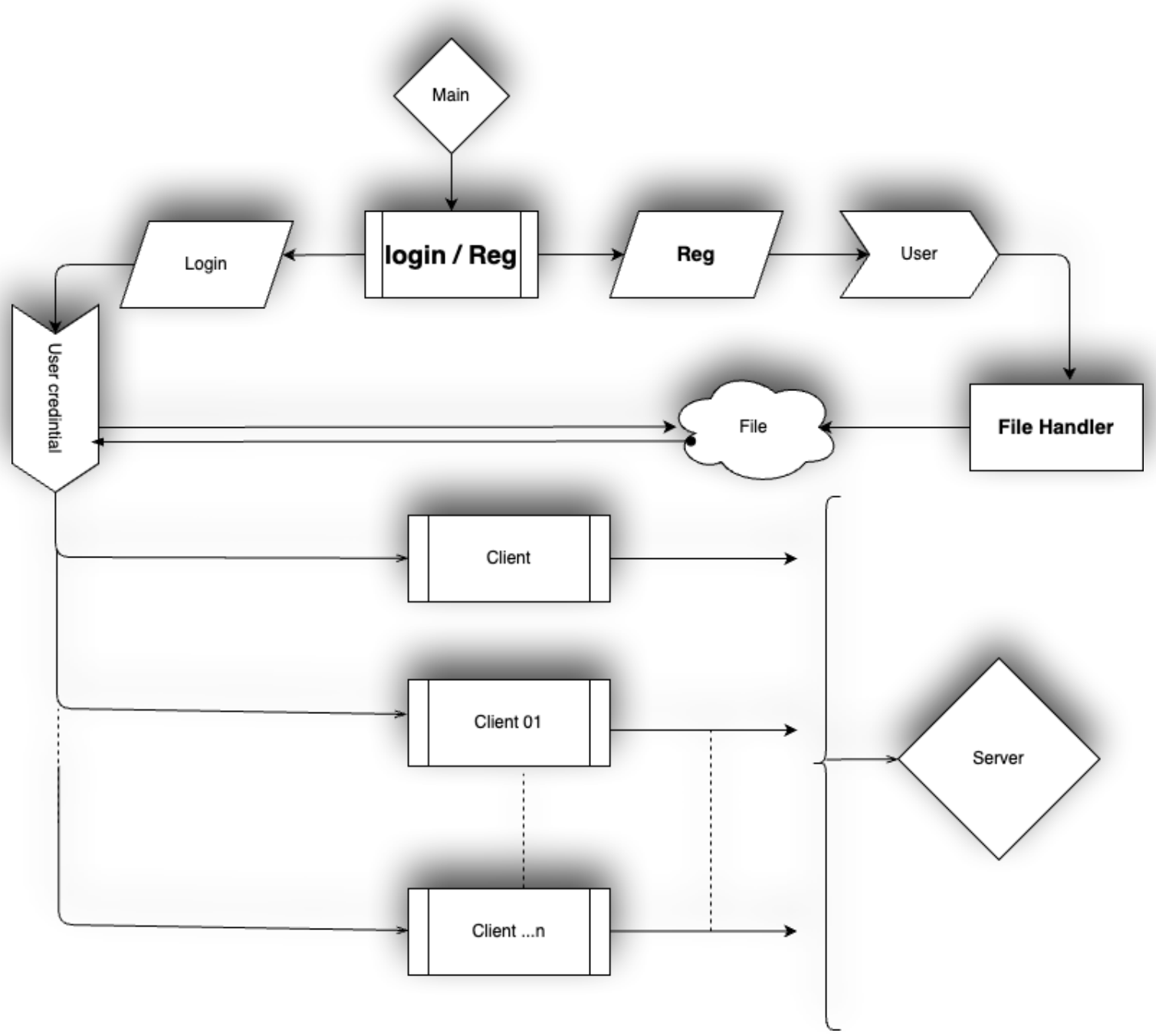
Problem Design (UML Class Diagram)

Class:

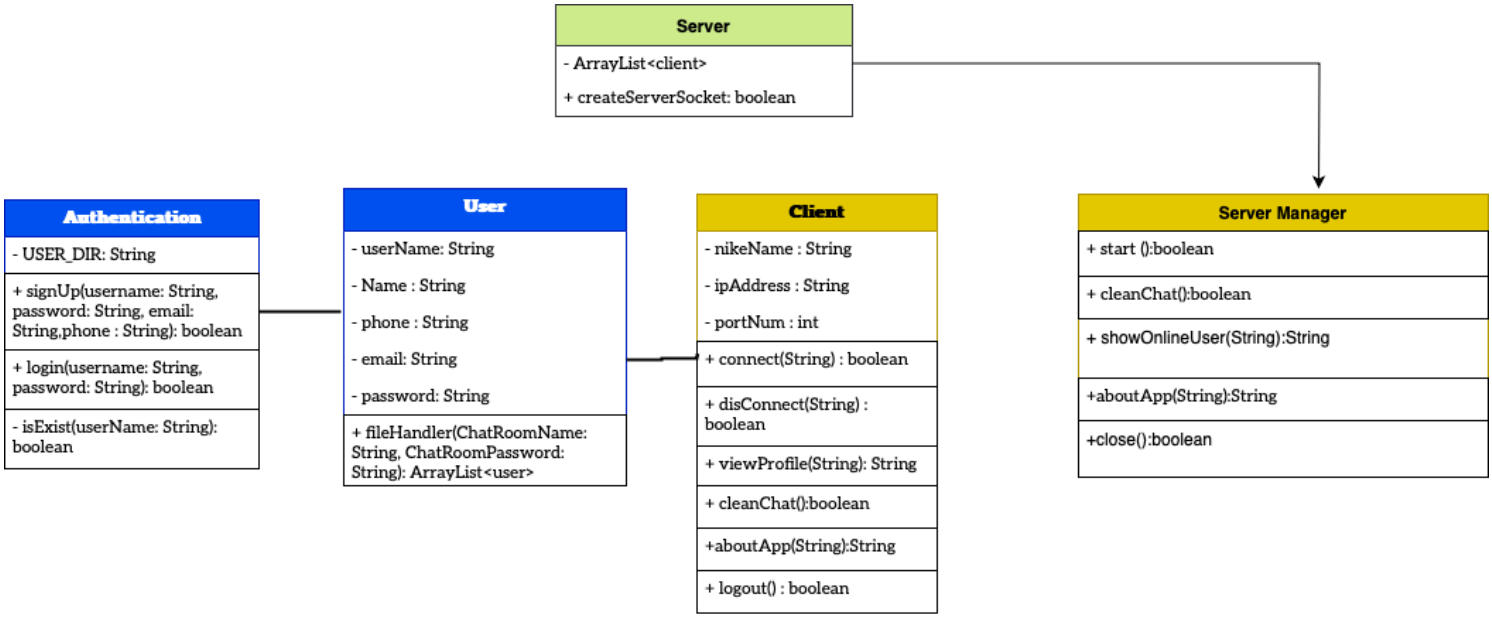
1. Registration
2. User
3. User Registration Controller
4. File handler (for Data read / Write)
5. Login
6. User

7. User Login Controller
8. Registration
9. Client
10. Server
11. Ip port nick name Handler

Working Method:



UML Diagram:



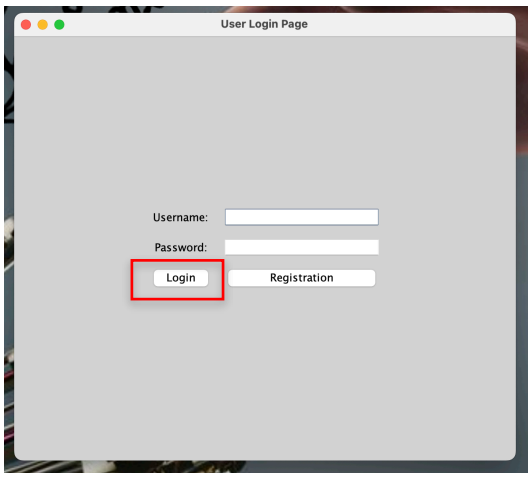
The UML class diagram serves as a foundational component of our project, Java Chat Application, providing a visual representation of the classes, their attributes, and relationships within the application. Through the UML diagram, we meticulously designed the structure of our software, delineating the key entities such as **users**, **Client**, **authentication mechanisms**, and **server-client interactions**. Each class, with its associated attributes and methods, was meticulously crafted to encapsulate data and behaviour.

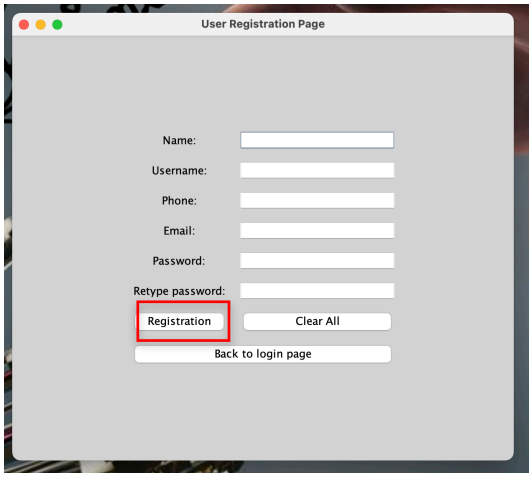
Implementation

The implementation phase of our Java Chat Application involved translating the design concepts outlined in the UML class diagram into functional code, encompassing various components and functionalities essential for a robust communication application. Here, we delve into the comprehensive details of the implementation, highlighting key aspects such as user management, chatroom creation and joining, authentication mechanisms, server-client communication, and error-handling strategies.

User Management:

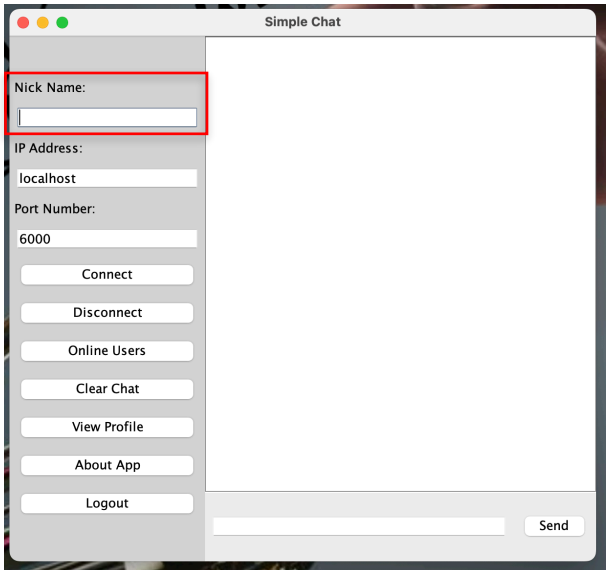
The user management module encompasses functionalities such as user sign-up, login, and session management. Users are prompted to sign up by providing a username, password, email address and phone number . The Authentication class validates user credentials during the sign-up and login processes, ensuring data integrity and security. Upon successful authentication, users are granted access to their respective accounts, where they can manage their profile information and interact with chatrooms.

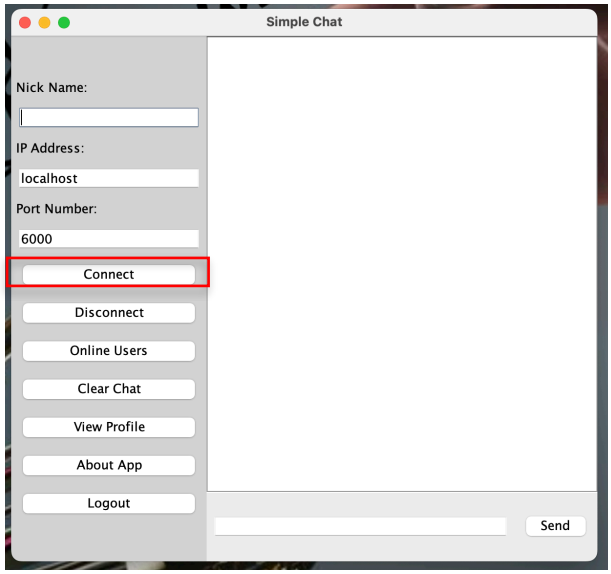




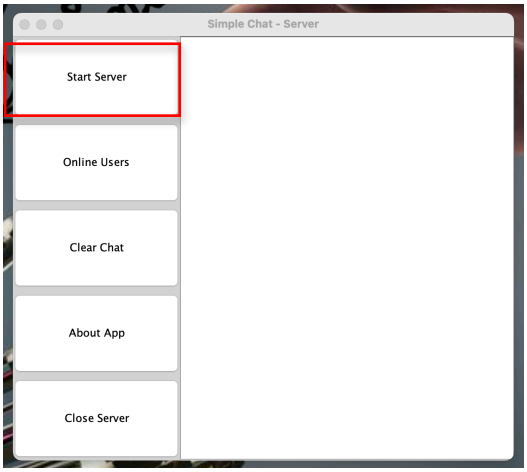
Chatroom Creation and Joining:

The chatroom module facilitates the creation, joining, and management of chatrooms. Users can create new chatrooms by specifying a name and optional password. Existing chatrooms can be joined by providing the appropriate credentials. The ChatRoomManager class handles chatroom management operations, including the addition of new chatrooms to the system and validation of user credentials during the joining process. Additionally, users can view a list of joined chatrooms and enter specific chatrooms to participate in discussions.



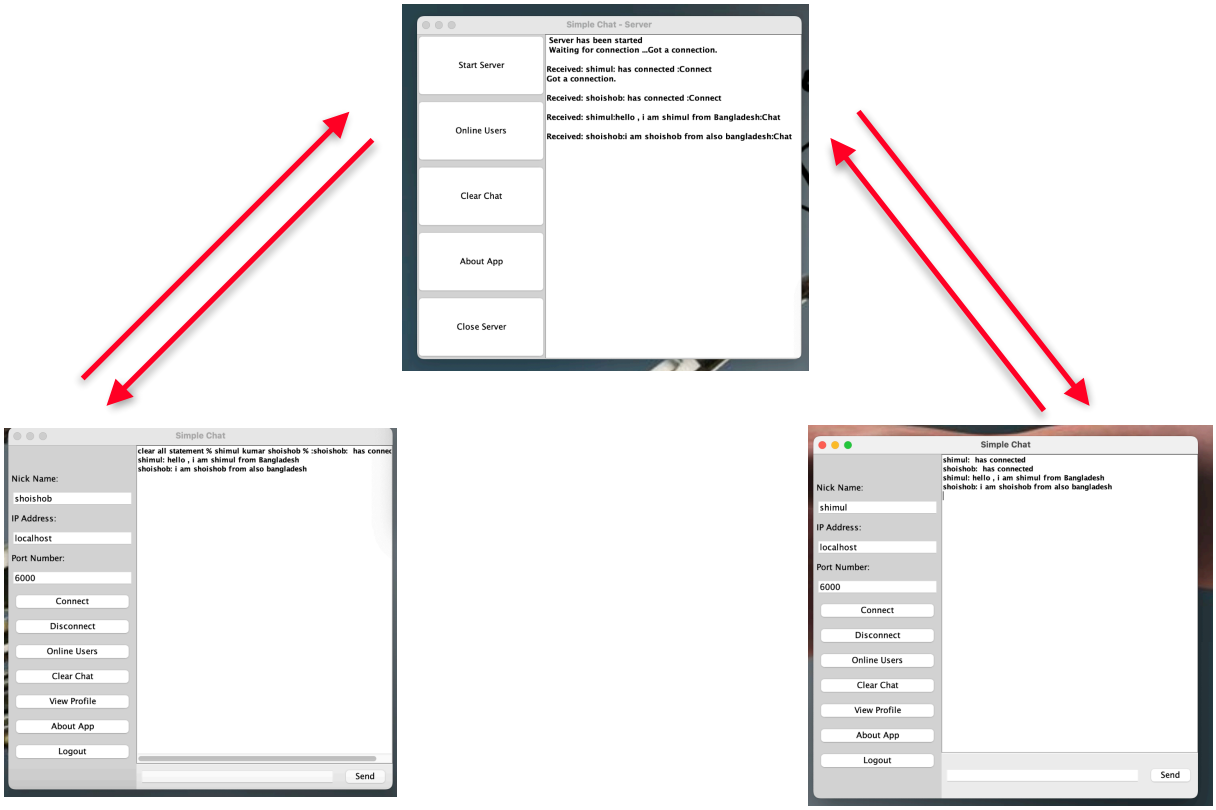


Server Joining:



Authentication Mechanisms:
Robust authentication mechanisms are implemented to ensure secure user access and protect against unauthorized usage. The Authentication class utilizes file-based storage to maintain user credentials securely. Passwords are hashed and stored in encrypted format to prevent unauthorized access to sensitive user information. During the login process, user credentials are verified against the stored data to authenticate user identity accurately.

Server-Client Communication:
The server-client communication module facilitates real-time messaging between users within chatrooms. The ChatServer class serves as the central component responsible for managing client connections, message broadcasting, and user interactions. Upon establishing a connection, clients are assigned unique identifiers and prompted to choose a nickname. Messages sent by clients are relayed to the server, which broadcasts them to all connected clients within the same chatroom. The ChatClient class enables users to interact with the server by sending and receiving messages seamlessly.



Error Handling Strategies:
Comprehensive error-handling strategies are implemented to enhance the reliability and stability of the application. The system employs exception-handling mechanisms to detect and handle errors gracefully, preventing application crashes and ensuring uninterrupted user experience. Error messages are displayed to users to communicate issues such as invalid input, authentication failures, or network errors effectively. Additionally, logging mechanisms are incorporated to capture and log runtime exceptions for debugging and troubleshooting purposes.

Reflection on Individual and Teamwork

The development of Java ChatChit was a collaborative effort, requiring effective communication and coordination among team members. Each team member contributed to different aspects of the project, including class implementation and documentation.

Limitations and Future Work

Limitations:

Limited Security Measures: The current implementation lacks advanced security features such as encryption for communication between clients and the server, potentially exposing user data to security risks.

Scalability Concerns: The application's architecture may face scalability challenges when handling a large number of concurrent users or chatrooms, potentially leading to performance degradation.

Basic User Interface: The user interface (UI) of the application is rudimentary and lacks advanced features for enhancing user experience, such as customization options or real-time updates.

Simplified Authentication: The authentication mechanism relies on basic username-password pairs stored in text files, which may not suffice for robust user authentication in real-world scenarios.

Future Work:

Enhanced Security Measures: Implement advanced security measures such as SSL/TLS encryption for secure communication, user authentication protocols like OAuth, and encryption for storing sensitive user data.

Scalability Improvements: Optimize the application's architecture to handle a larger user base and chatroom capacity efficiently, possibly by implementing load balancing and distributed computing techniques.

Improved User Interface: Revamp the user interface with modern design principles, responsive layouts, and interactive features to enhance usability and user engagement.

Advanced Authentication: Introduce multi-factor authentication (MFA), biometric authentication, or other advanced authentication mechanisms to enhance user security and prevent unauthorized access.

Additional Features: Incorporate additional features such as private messaging, multimedia support (e.g., file sharing, image/video embedding), message formatting options (e.g., markdown support),

and notification systems for improved user experience and functionality.

Performance Optimization: Conduct performance profiling and optimization to improve the application's responsiveness, reduce latency, and optimize resource utilization for better overall performance.

Localization and Internationalization: Implement support for multiple languages and localization options to cater to users from diverse linguistic backgrounds and geographical regions.

Continuous Testing and Quality Assurance: Establish robust testing frameworks and practices, including automated testing, regression testing, and security audits, to ensure the reliability, stability, and security of the application throughout its lifecycle.

End

~shimul kumar shoishob