Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2024), B.Sc. in CSE (Day)

Multiplayer Chess Game

Course Title: Computer Networking Lab
Course Code: CSE 312
Section: 221 D5

Students Details

Name	ID
Maymuna Akter	221902262
Sumaiya Akter Akhi	221002332

Submission Date: 18-11-2024
Course Teacher's Name: Md. Saiful Islam Bhuiyan

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

This project focuses on developing a multiplayer chess game using Java socket programming. The application allows two players to connect over a network, make their moves alternately, and view the progress of the game on a shared virtual chessboard. The game ensures synchronization between the players' boards to provide a seamless user experience. The implementation will emphasize efficient network communication, game state management, and adherence to the rules of chess.

1.2 Motivation

Chess is a classic strategic game enjoyed by millions of people worldwide, and digital versions have made it more accessible. However, building a multiplayer chess game from scratch, especially using Java socket programming, offers an excellent opportunity to explore networking, game logic, and user interaction. This project combines personal interest in games with the challenge of developing real-time multiplayer functionality, making it both exciting and educational.

1.3 Problem Definition

With the growing interest in online gaming, players seek reliable platforms to engage with friends or others in real time. Many chess applications rely on third-party services and complex infrastructure, limiting opportunities for developers to learn the underlying mechanics. A lightweight, socket-based multiplayer chess game will address this gap, providing a platform for users to enjoy chess while showcasing how networking and gaming concepts can be integrated.

1.3.1 Problem Statement

Design and implement a Java-based multiplayer chess game using socket programming. The application will enable two players to connect over a network, make moves alternately, and display all moves on a synchronized chessboard. The solution must handle network communication efficiently, validate chess moves according to the rules, and maintain a consistent game state to ensure a smooth user experience.

1.3.2 Complex Engineering Problem

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	**Explain how to address**
P1: Depth of knowledge required	Hard syntax knowledge need for this project.
P2: Range of conflicting requirements	—-
P3: Depth of analysis required	A simple syntax error can change the output.
P4: Familiarity of issues	—-
P5: Extent of applicable codes	—-
P6: Extent of stakeholder involvement and conflicting requirements	—-
P7: Interdependence	Game logic depends on these communications to validate moves, update the board state, and ensure synchronization between the players' views.

1.4 Design Goals/Objectives

Real-Time Multiplayer Connectivity: Develop a robust socket-based system to allow two players to connect over a network, ensuring real-time communication and move synchronization.

Seamless Game Play: Ensure that the chess game adheres strictly to the official rules, including valid moves, turn-based play, and special moves such as castling, en passant, and pawn promotion.

User-Friendly Interface: Create an intuitive and visually appealing chessboard GUI that displays real-time updates and allows users to easily interact with the game.

Error Handling and Stability: Incorporate mechanisms to detect and recover from network issues, invalid moves, or unexpected disconnections, ensuring a stable and smooth gaming experience.

Scalability and Modularity: Design the system with modular components (e.g., networking, game logic, and GUI) to allow for future enhancements like AI opponents or additional game modes.

Item Visibility: Items that are not on the board (such as captured pieces or extra items like timers, scores, etc.) should be clearly visible to players without obstructing the chessboard. You can display these on a sidebar or a separate panel.

Game State Validation: Ensure that the game state. is controlled by the server and not the client. Clients should only send valid moves, and the server should be responsible for validating the moves, updating the board state, and notifying all players of the changes. This prevents players from manipulating the game by sending arbitrary data.

1.5 Application

Educational Tool: Serve as a practical example for learning Java socket programming, game logic implementation, and real-time networking. Useful for computer science students or developers exploring the fundamentals of multiplayer game development.

Entertainment Platform: Provide a lightweight chess platform for casual players to enjoy multiplayer games without relying on internet-heavy services.

Skill Development: Enhance the logical and analytical skills of players through a digital chess interface.

Foundation for Advanced Features: Act as a base for future projects, such as implementing AI players, online matchmaking, or advanced graphical enhancements.

Personal Portfolio Project: Demonstrate the developer's capability in Java socket programming, GUI design, and problem-solving to potential employers or academic evaluators.