Gaming System

Team Leader:- Arnab Paul(22CS8051)

Team Members:-

Saptak Chakrabarti(22CS8052)

Soumojit Dutta(22CS8056)

Sushmita Khatun(22CS8053)

Y.Anand Reddy(22CS8057)

Baidurjya Sinha(22CS8054)

Rohit Sharma(22CS8058)

N.Yashwanth Reddy(22CS8055)

Dinesh Kumar Goud(22CS8059)

Ishani Sarkar(22CS8060)

Client-Server Architecture

Server Responsibilities

The server will handle user authentication, game state management, and turn coordination.

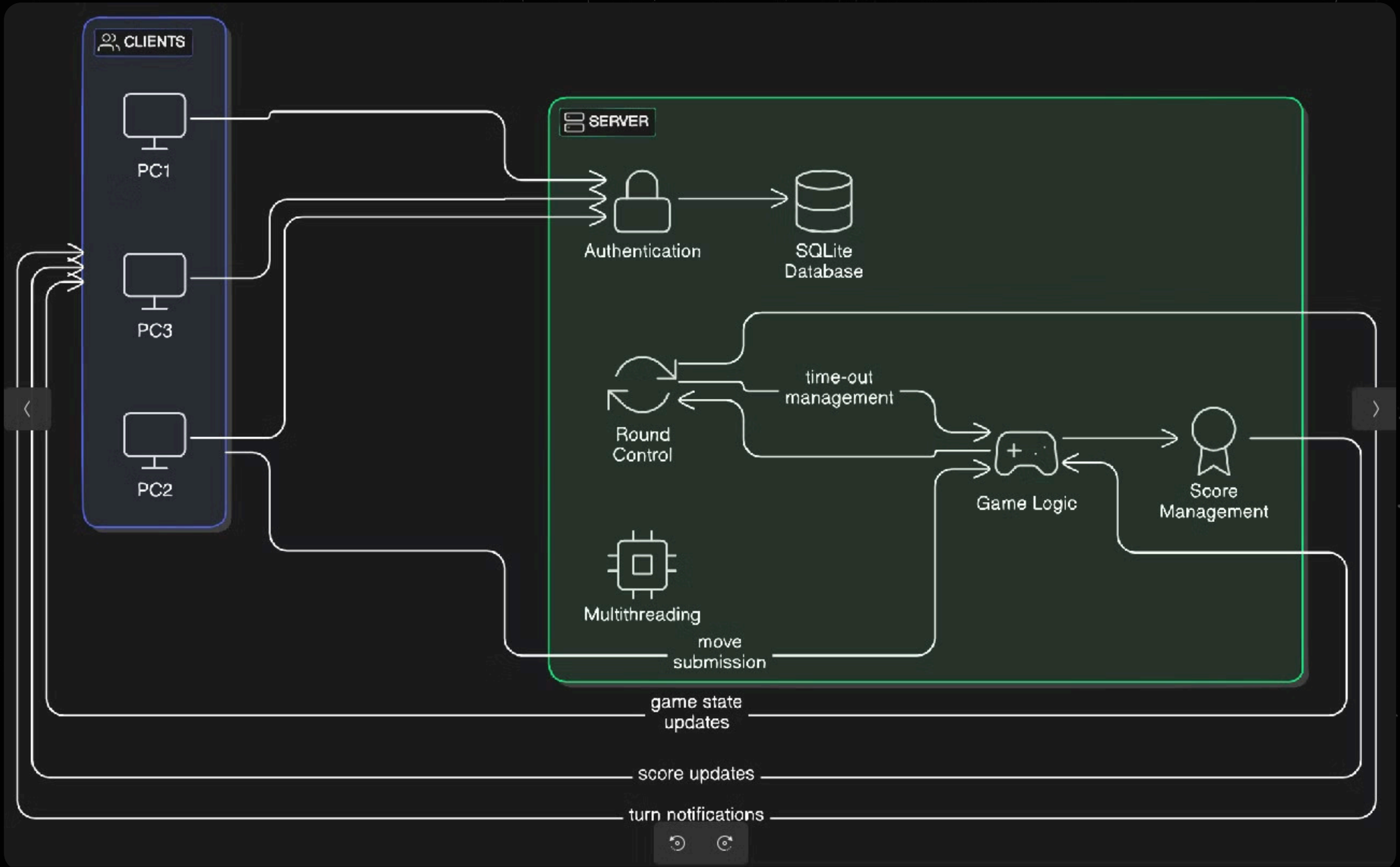
Client Connectivity

Clients will connect to the server from different terminals, logging in and making game moves.

Communication Protocol

The server and clients will communicate using sockets in C, with the server listening for and facilitating client connections.

Client-Server Architecture Breakdown



Server Functionality

1 Authentication

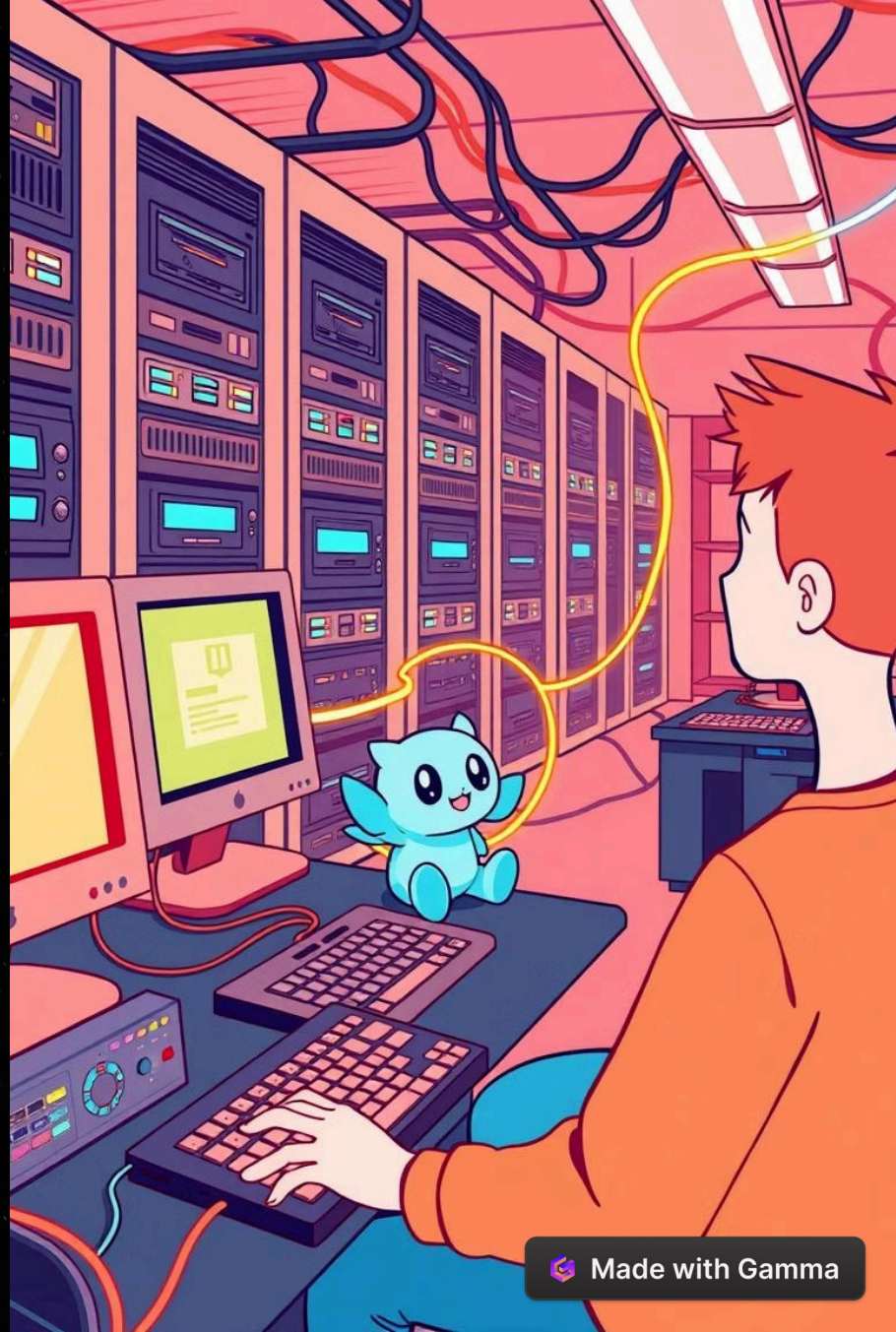
The server will be responsible for verifying user credentials and managing secure login sessions.

2 Game State

The server will maintain the current state of the game, including player positions, scores, and other critical data.

3 Turn Management

The server will coordinate the game turns, ensuring a smooth and synchronized gameplay experience.



Client Connectivity

1

Login

Clients will connect to the server and authenticate with their credentials.

2

Gameplay

Clients will interact with the game, making moves and updates that are relayed to the server.

3

Coordination

The server will manage the clients' actions, ensuring a synchronized and fair gameplay experience.

Networking with Sockets

Socket Creation

The server will create and bind a socket to listen for incoming client connections.

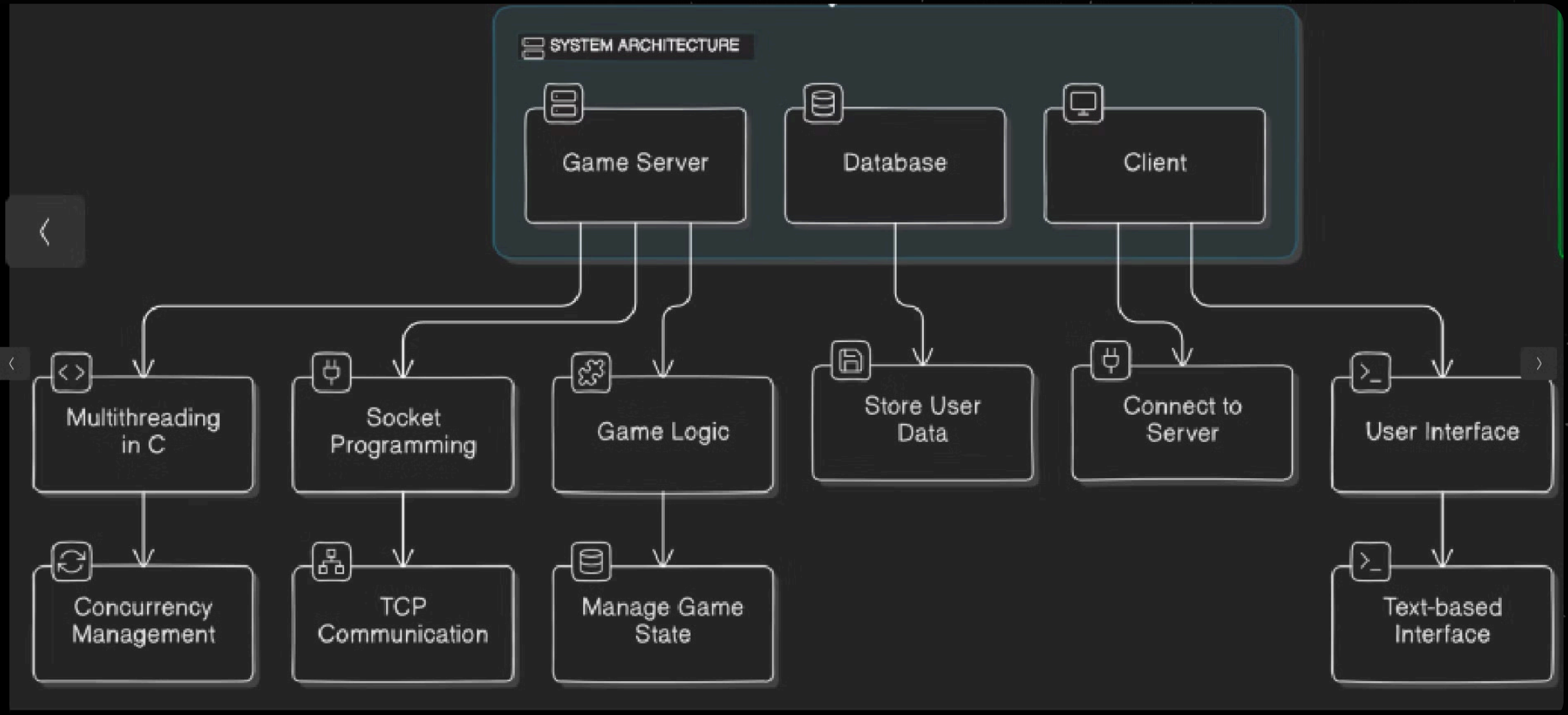
Connection Handling

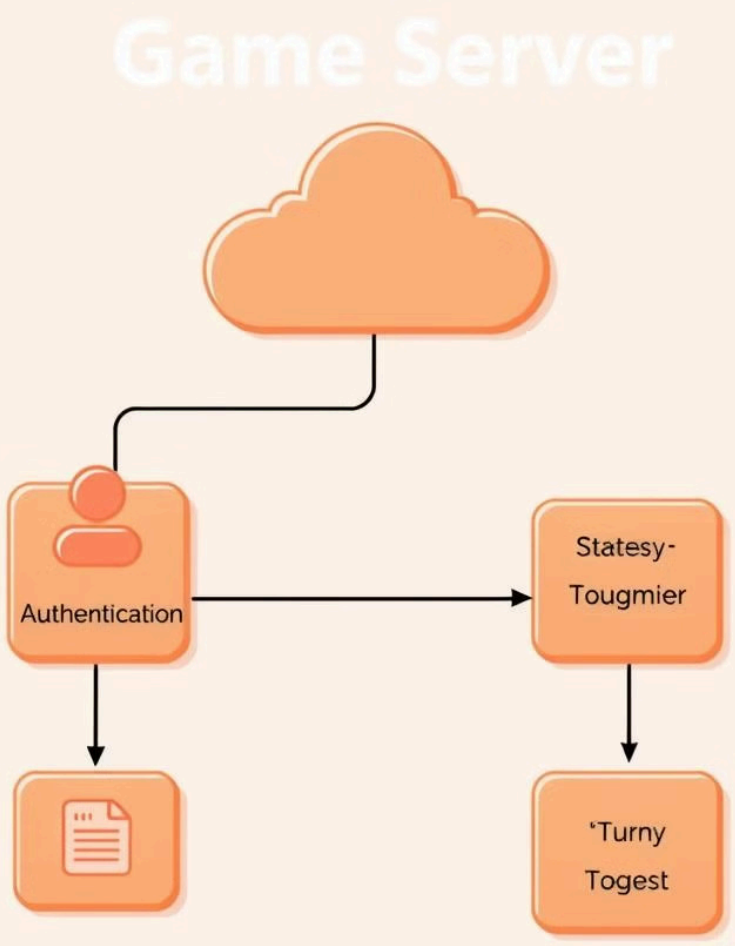
The server will accept client connections and manage their sessions using the socket API.

Data Transfer

Clients and the server will exchange game-related data, such as player actions and updates, through the socket interface.

Components Flowchart





Components Overview

Authentication

Secure user login and session management.

Game State

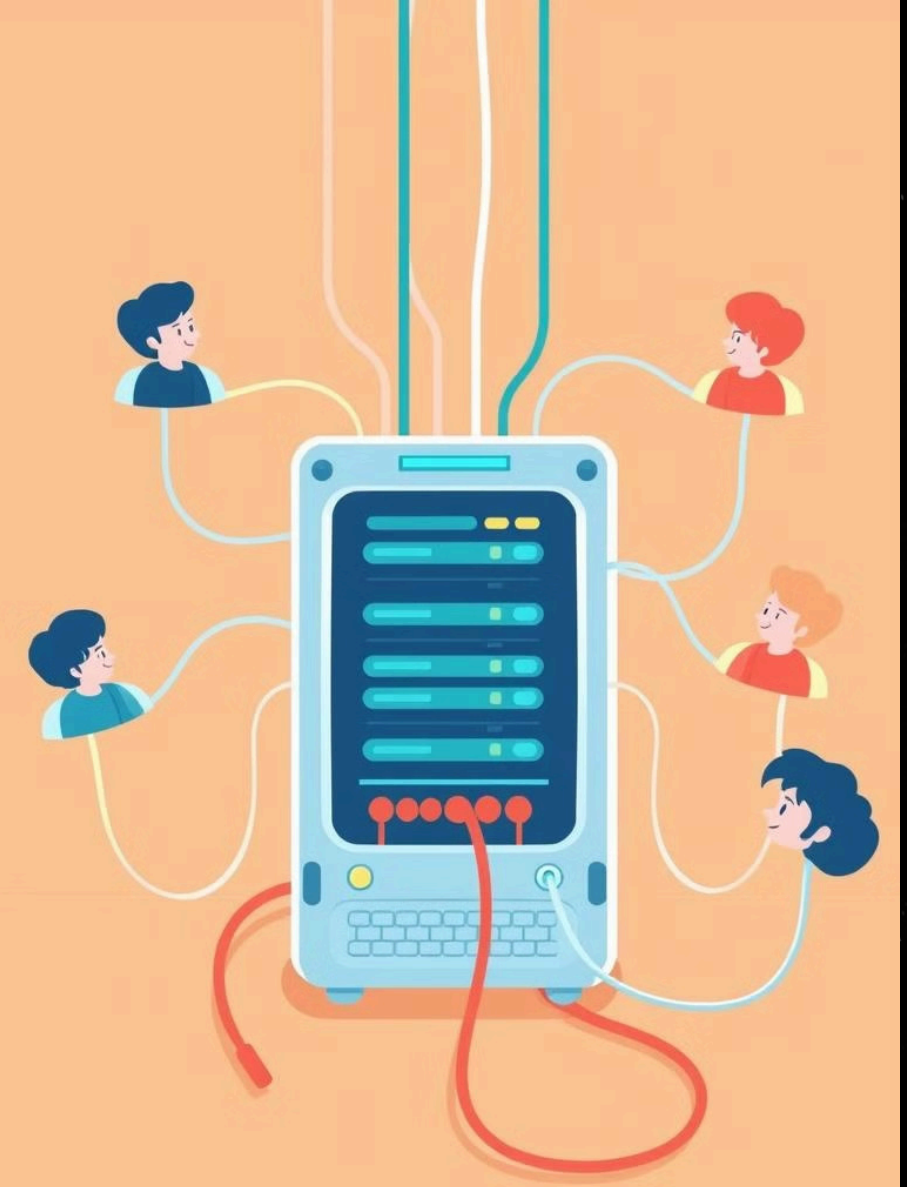
Maintaining the current state of the game.

Turn Management

Coordinating player turns and moves.

Networking

Facilitating client-server communication.



Game Server (C-based)

1

Thread Pooling

The server will use POSIX threads (pthread.h) to create a pool of worker threads to handle multiple client connections.

2

Connection Management

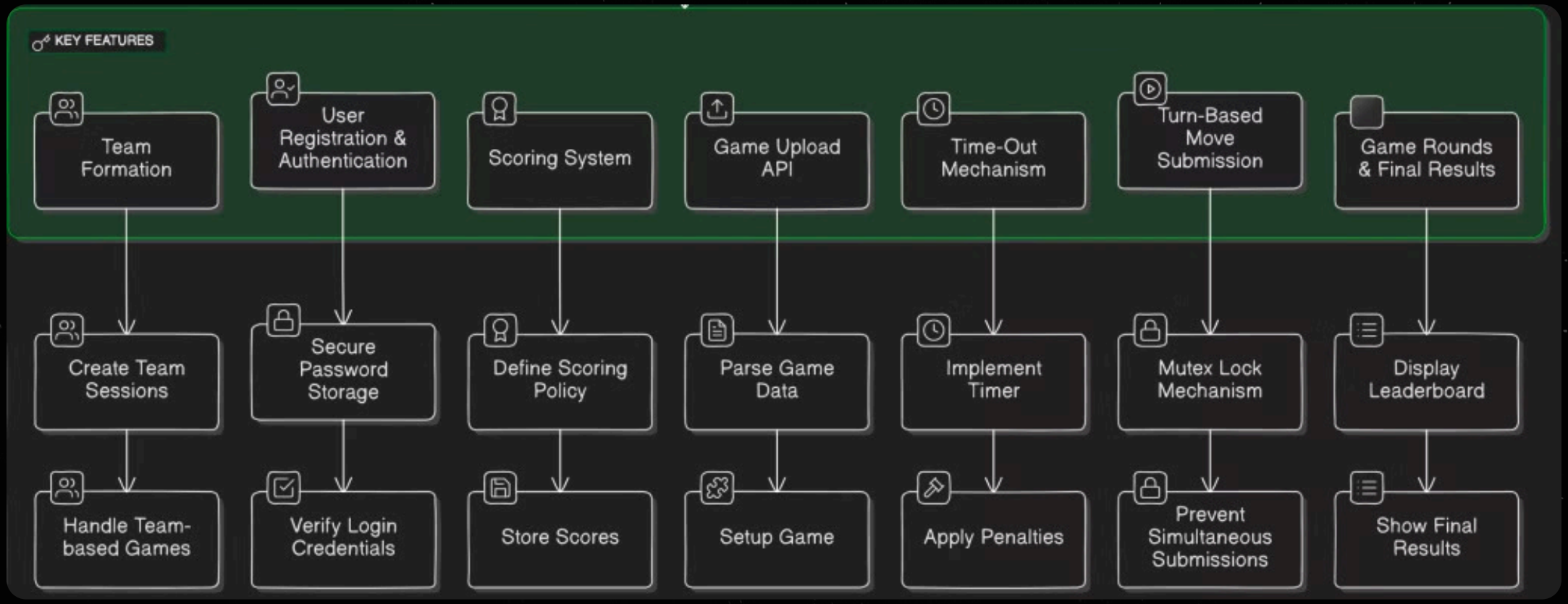
Each client connection will be assigned to a dedicated worker thread, which will manage the client's gameplay and interactions.

3

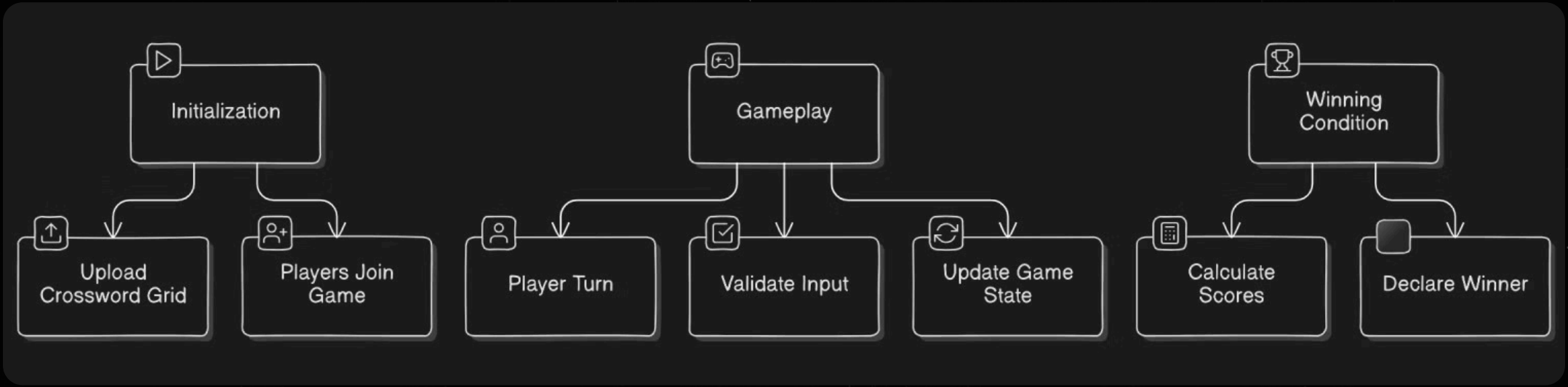
Synchronization

The worker threads will coordinate with the main server thread to ensure synchronized game state updates and turn processing.

Key Features



Game Mechanism



Conclusion

By leveraging a client-server architecture and utilizing C-based multithreading, the game server will be able to efficiently handle multiple client connections, manage game state, and facilitate a smooth and synchronized gameplay experience.