NAME: PIYUSH PRASAD
REGISTRATION NUMBER: 21BCE2639

1)
Server-Side Implementation (Node.js)
CODE:

```
const WebSocket = require('ws');
const wss = new WebSocket.Server({ port: 8080 });

let gameState = initializeGameState();
let currentPlayer = 'A'; // Starting player

function initializeGameState() {
  // Initialize a 5x5 grid with characters
  return {
    board: Array(5).fill(null).map(() => Array(5).fill(null)),
    positions: {
      A: { P1: [0, 0], P2: [0, 1], P3: [0, 2], P4: [0, 3], P5: [0, 4] },
      B: { P1: [4, 0], P2: [4, 1], P3: [4, 2], P4: [4, 3], P5: [4, 4] }
    },
    turn: 'A'
  };
}

function processMove(data) {
  const { character, move } = data;
  const player = gameState.turn;
  const positions = gameState.positions[player];
  const [x, y] = positions[character];

  // Check if the move is valid based on the character type and position
  if (move === 'L') y--;
  else if (move === 'R') y++;
  else if (move === 'F') x--;
  else if (move === 'B') x++;
  else return { valid: false, error: 'Invalid move' };

  if (x < 0 || x >= 5 || y < 0 || y >= 5) return { valid: false, error: 'Out of bounds' };

  // Apply the move
  positions[character] = [x, y];

  // Switch turns
  gameState.turn = gameState.turn === 'A' ? 'B' : 'A';

  return { valid: true, newState: gameState };
}

wss.on('connection', ws => {
  ws.on('message', message => {
    const { type, data } = JSON.parse(message);

    switch (type) {
      case 'initialize':
        ws.send(JSON.stringify({ type: 'state', data: gameState }));
        break;
      case 'move':
```

```
                const result = processMove(data);
                if (result.valid) {
                    wss.clients.forEach(client => {
                        if (client.readyState === WebSocket.OPEN) {
                            client.send(JSON.stringify({ type: 'state', data: result.newState }));
                        }
                    });
                } else {
                    ws.send(JSON.stringify({ type: 'invalid_move', data: result.error }));
                }
                break;
        }
    });
});

console.log('WebSocket server is running on ws://localhost:8080');
```

2) Client-Side Implementation
index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Chess-like Game</title>
    <style>
        .grid {
            display: grid;
            grid-template-columns: repeat(5, 50px);
            gap: 2px;
            margin-bottom: 20px;
        }
        .cell {
            width: 50px;
            height: 50px;
            border: 1px solid #ccc;
            text-align: center;
            line-height: 50px;
            font-size: 20px;
        }
        .button {
            margin: 5px;
        }
    </style>
</head>
<body>
    <div id="gameBoard" class="grid"></div>
    <div id="controls"></div>
    <div id="status"></div>

    <script>
        const ws = new WebSocket('ws://localhost:8080');

        ws.onmessage = (event) => {
            const message = JSON.parse(event.data);
            switch (message.type) {
                case 'state':
                    updateBoard(message.data);
                    break;
                case 'invalid_move':
                    alert(`Invalid Move: ${message.data}`);
                    break;
                default:
                    console.log('Unknown message type:', message.type);
            }
        };

        function updateBoard(state) {
            const board = document.getElementById('gameBoard');
            board.innerHTML = '';
            for (let row = 0; row < 5; row++) {
                for (let col = 0; col < 5; col++) {
                    const cell = document.createElement('div');
                    cell.className = 'cell';
                    cell.textContent = state.positions['A'][`P${row*5+col+1}`] ? 'A' :
(state.positions['B'][`P${row*5+col+1}`] ? 'B' : '');
                    board.appendChild(cell);
                }
            }
            document.getElementById('status').textContent = `Current Turn: ${state.turn}`;
        }

        function sendMove(character, move) {
            ws.send(JSON.stringify({ type: 'move', data: { character, move } }));
        }

        // Example of how you could set up controls (not functional)
        // You would need to add logic to handle clicks and send appropriate moves
        document.getElementById('controls').innerHTML = `
            <button class="button" onclick="sendMove('P1', 'L')">Move P1 Left</button>
            <button class="button" onclick="sendMove('P1', 'R')">Move P1 Right</button>
            <button class="button" onclick="sendMove('P1', 'F')">Move P1 Forward</button>
            <button class="button" onclick="sendMove('P1', 'B')">Move P1 Backward</button>
        `;
    </script>
</body>
</html>
```