Design Document
Contact: Online Multiplayer Word Guessing Game

CS5610 - Web Development

Ganesh Umasankar Vignesh Pakkam Saravanan

1 Project Description

Contact is a real-time multiplayer word-guessing game designed for friends to play together from anywhere. Players join instantly without registration - just enter a nickname, create or share a room code, and start playing. The game combines cooperative gameplay with competitive scoring to create an engaging social experience.

1.1 Target Audience

Contact is designed for groups of friends (3-6 players) who enjoy word games and social gaming. The platform serves three primary user groups:

- Wordmasters: Users who enjoy choosing challenging words and defending them against guessers
- Guessers: Users who collaborate to deduce letters through clever clues and coordinated "contacts"
- Competitive Players: Users who enjoy real-time scoring and strategic gameplay within each game session

2 User Personas

2.1 Party Host - Alex

Alex is a 26-year-old who loves hosting game nights. Alex wants to start games quickly without complex setup, easily share room codes with friends via text, see everyone's status in real-time, and ensure rules are clear for new players. Alex expects instant room creation, simple controls, and smooth gameplay without technical issues.

2.2 Word Game Enthusiast - Jamie

Jamie is a 30-year-old teacher who enjoys word puzzles and strategic thinking. Jamie wants to give creative clues that stump the Wordmaster, coordinate with other guessers through careful timing, see immediate score updates during the game, and experience smooth real-time gameplay. Jamie expects fair gameplay mechanics, clear feedback on actions, and transparent scoring within each game session.

2.3 Casual Gamer - Morgan

Morgan is a 24-year-old who plays games socially with friends. Morgan wants to understand the game quickly without reading lengthy instructions, see immediate feedback on whether guesses are correct, enjoy smooth real-time updates, and see how everyone is scoring during the game. Morgan expects intuitive UI, minimal learning curve, and fun social interactions.

3 User Stories

3.1 Room Management Stories (Ganesh Umasankar)

- As a new user, I want to enter a nickname to join games without registration.
- As a room creator, I want to generate a unique room code to share with friends.
- As a player, I want to join rooms by entering a room code so I can play with friends.
- As a player in a lobby, I want to see all players and their selected roles in real-time.
- As a player, I want to select my role (Wordmaster or Guesser) before the game starts.
- As a room admin, I want to configure room settings (round time, Wordmaster guesses) to customize gameplay.
- As a room admin, I want to start the game when all players are ready.
- As a room admin, I want to kick players if needed to manage the room.
- As a player, I want to leave the room and return to the home page.
- As a player, I want my session to persist if I refresh the page so I don't lose my spot.

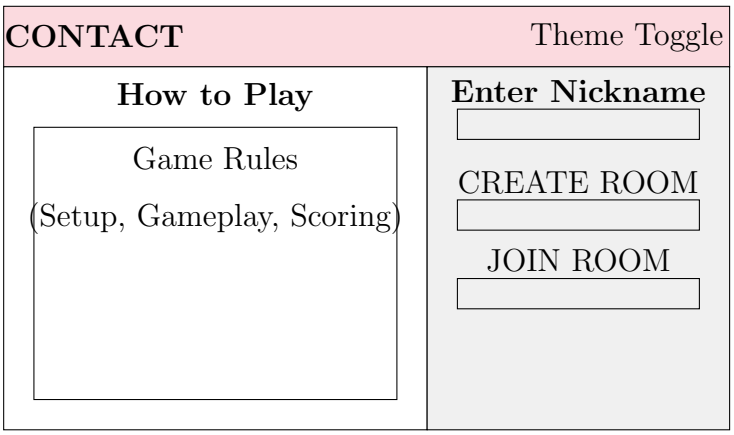
3.2 Gameplay Stories (Vignesh Pakkam Saravanan)

- As a Wordmaster, I want to privately enter a secret word to start the game.
- As a Wordmaster, I want to select the word type (noun/verb/adjective) for context.
- As a Guesser, I want to see the revealed letters and word length (e.g., "EL _ _ _ _ _ _").
- As a clue-giver, I want to enter my clue word and descriptive clue when it's my turn.
- As a clue-giver, I want to optionally provide a second clue halfway through the round.
- As a Guesser, I want to click "CONTACT!" and secretly enter my guess word.
- As a player, I want to see which players have clicked Contact in real-time.
- As a Wordmaster, I want to submit blocking guesses with remaining attempts shown.
- As a player, I want to see a round transition screen showing clue word reveals and whether contact succeeded.
- As a Guesser, I want to attempt to guess the full target word after each letter reveal.
- As a player, I want to see real-time score updates on the scoreboard.
- As a player, I want to see a detailed game log of all actions and events.
- As a player, I want to see a victory screen when someone wins.
- As a player, I want to see the final scoreboard ranked by score.
- As a player, I want to return to the room lobby to play again after the game ends.

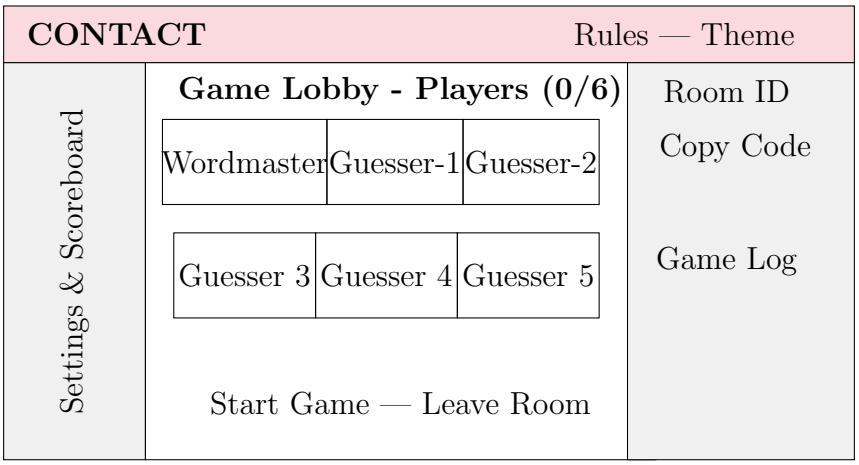
4 Design Mockups

Contact uses a three-column layout for the game interface with a clean two-column layout for the home page.

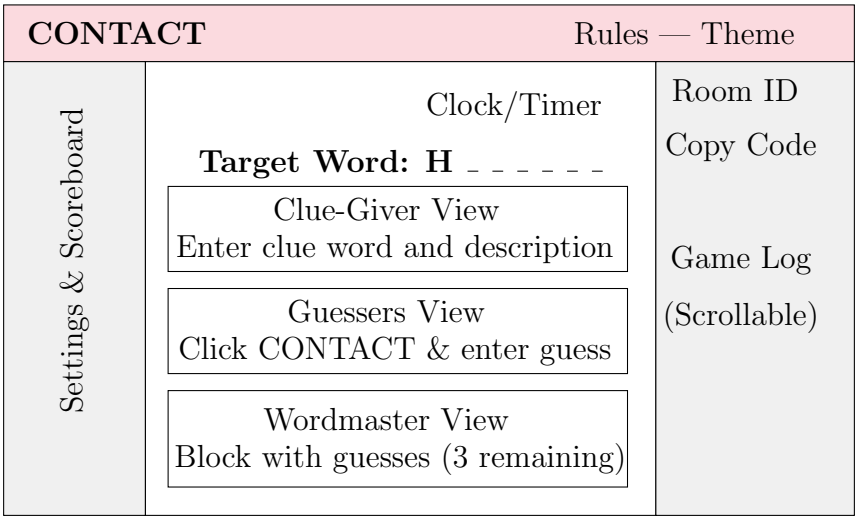
4.0.1 Home Page Layout



4.0.2 Room Lobby Layout



4.0.3 Active Game Layout



4.1 Navigation Structure

Top Header: Logo (left), Rules button, and Theme toggle (right)
Left Sidebar: Room Settings (round time, WM guesses), Player Scoreboard
Right Sidebar: Room ID with copy button, Game Log with scrollable event history
Main Content: Dynamic based on game state (lobby, timmer, Wordmaster setup, active gameplay, transitions, victory)

4.2 Key UI Components

Player Slots: Cards showing nickname, role badge (Wordmaster/Guesser/Clue-Giver), "YOU" indicator, kick button (admin only)
Game Controls: Input fields for clues/guesses, action buttons (Submit, Contact, Block), timer display
Scoreboard: Ranked list with player nickname, role badges, current score, highlighting for active clue-giver
Game Log: Chronological event list with color-coding (success=green, error=red, info=pink), timestamps, auto-scroll
Transitions: Full-screen overlays showing round results, revealed words, countdown timers (15s), letter reveals
Victory Screen: Winner announcement, secret word reveal, final scores, celebration animation, "Back to Room" button

5 Technical Overview

5.1 Technology Stack

- React 18.2.0 with Hooks, React Router DOM 6.21.1, HTML5, CSS3
- Node.js with Express 4.18.2
- MongoDB 6.3.0 native driver (no Mongoose)
- Socket.io 4.6.1 for real-time communication, Socket.io-client 4.6.1
- express-session 1.17.3, connect-mongo 5.1.0 for session management
- react-scripts 5.0.1 for build tooling, ESLint 8.56.0, Prettier 3.1.1
- Deployment: Render (Static Site + Web Service), MongoDB Atlas

5.2 Data Structure

Contact uses MongoDB with two primary collections: **Rooms** (manages lobby state, players, settings with full CRUD) and **Games** (manages active gameplay, rounds, clues, contacts, scores with full CRUD). All pages use client-side rendering with React components fetching data from Express REST API endpoints and receiving real-time updates via Socket.io events.

6 Key Features

- **Session Management**: Browser-based sessions using sessionStorage with player IDs persisting across refreshes. 5-second grace period for disconnections prevents accidental player removal.
- **Room Creation & Joining**: Generate unique 6-character room codes. Join rooms via code entry. Real-time lobby updates showing all players and roles.
- **Role Selection**: Players choose Wordmaster or Guesser roles in lobby. Visual indicators (colored badges) distinguish roles.
- **Real-time Gameplay**: Socket.io enables instant updates across all players. Live tracking of clue submissions, contact clicks, Wordmaster guesses, letter reveals, and score changes.
- **Progressive Letter Reveals**: First letter revealed at start. Additional letters revealed only after successful contacts. Visual display shows revealed letters and blanks (e.g., "EL _ _ _ _ _").
- **Game Log**: Real-time chronological event log. Color-coded entries (success/error/info). Timestamps and detailed descriptions. Auto-scroll to latest entries.

7 Implementation Challenges & Solutions

- **Real-time State Synchronization**: Keeping game state consistent across multiple connected clients with rapid interactions. Solution: Used Socket.io room broadcasting with MongoDB atomic operations ($set, $push, $inc) to prevent race conditions and ensure state consistency.
- **Session Persistence Across Refreshes**: Maintaining player identity when users refresh browser during gameplay. Solution: Store playerId in sessionStorage and validate with backend sessions using express-session + connect-mongo. Automatic reconnection via Socket.io with grace period.
- **Complex Game State Management**: Managing multi-round game flow with rotating clue-givers, simultaneous submissions, and timed events. Solution: Structured Games collection with nested rounds array, clueGiverIndex rotation, and server-side timers with Socket.io emit for countdown synchronization.
- **Handling Player Disconnections**: Gracefully managing mid-game disconnects without breaking gameplay for others. Solution: 5-second grace period before permanent disconnect. Different handling for Wordmaster vs Guesser vs Clue-Giver disconnects. Game ends if too few players remain.
- **Coordinating Multiple Simultaneous Actions**: Multiple guessers clicking Contact, Wordmaster blocking, and timer expiring at same time. Solution: Event queuing on backend with sequential processing. Locks on critical game state updates. Clear event ordering in game log.
- **CORS Configuration**: Frontend and backend on different Render domains causing CORS issues. Solution: Configured express CORS middleware with explicit origin whitelist using CLIENT_URL environment variable. Allowed credentials for session cookies.
- **React Component State vs Socket Updates**: Balancing local React state with incoming Socket.io events to avoid UI flicker. Solution: Single source of truth approach - backend owns game state, React components reflect it. Used React useState with useEffect cleanup for Socket listeners.

8 What Makes Contact Special

- **Instant Multiplayer**: No registration required - enter nickname, share code, play. Friends can join in seconds, making it perfect for spontaneous game sessions.
- **True Real-time Gameplay**: Socket.io enables seamless synchronization across all players. Every action (clue submission, contact click, blocking attempt) appears instantly for everyone.
- **Cooperative Yet Competitive**: Guessers must collaborate to make contacts succeed, but compete for the winning guess.
- **Strategic Depth**: Wordmasters must balance blocking obvious clues vs conserving guesses. Clue-givers walk the line between too obvious and too obscure.
- **Transparent Game State**: Comprehensive game log shows exactly what happened and when. Real-time scoreboard keeps everyone aware of standings. Clear visual indicators for active player, remaining attempts, and timer.
- **Modern Tech Stack**: React Hooks for clean component logic. Socket.io for reliable WebSocket communication. MongoDB native driver for flexible data modeling. Express sessions for secure state management.