1. High-Level Architecture 🏗️

The application uses a modern, full-stack approach with a clear separation between the client, real-time server, and optional persistence layer.

Component	**Technology**	**Role**	**Deployment Target**
Frontend	**Next.js** (App Router), **Tailwind CSS**	Responsive UI, client-side game logic, microphone access.	**Vercel**
Realtime Layer	**Socket.IO Server** (Node/Express)	Manages rooms, players, roles, game state, and real-time events.	**Render / Fly / DigitalOcean / Heroku**
Speech-to-Text	**Browser Web Speech API**	Client-side transcription of Speaker clues. Transcripts sent to server for validation.	Client-Side (Browser)
Persistence (Optional)	Postgres / Firestore	Store decks, user data, leaderboards. *MVP will use local JSON for decks.*	Separate DB Host

Note on Mic Access: The deployed application **must use HTTPS** to enable microphone access via the Web Speech API.

---

2. Game Flow (Sequence) ➡️

This outlines the typical sequence of events from joining the site to game completion.

1. **Lobby:** Players open the site $\rightarrow$ **join a lobby** (enter name) $\rightarrow$ **create/join a room**.
2. **Start:** When the room has **4 players**, the game can start. The Server picks **Player 1 as the initial Speaker**.
3. **Round Setup:** Server sends a **Card** (target + forbidden words) to the **Speaker only**.
4. **Clue Giving:**
    - Speaker presses **Start Round** $\rightarrow$ microphone activates.
    - Speaker gives up to **4 clues**, one by one.

    - Each clue is **transcribed by the browser** (Web Speech API) and sent to the server (speaker-transcript).
5. **Forbidden Check:**
    - Server checks the transcript for **forbidden words**.
    - If a forbidden word is found: Server applies **penalty** to the Speaker and notifies all clients (forbidden-detected).
6. **Guessing:**
    - Guessers each have a **6-guess counter**.
    - Guessers submit guesses (voice or text) (guesser-guess).
    - Server checks if the guess equals the normalized target word.
7. **Round End Conditions:**
    - **Success:** If correct, the server awards points, **ends the round**, and broadcasts results (round-ended).
    - **Failure 1 (Exhausted Guesses):** If all Guessers exhaust **6 guesses each**, the round ends without a correct answer.
    - **Failure 2 (Rule Break):** If the Speaker says a forbidden word (and the chosen edge case rule is to end the round).
8. **Next Round:** The **next Speaker is chosen** (round-robin). Repeat until the game ends (e.g., all cards played).
9. **Game End:** Final scoreboard shown.

---

3. Data Models (TypeScript Interfaces) 💾

Card Interface

TypeScript

```
None
// Card
interface Card {
  id: string;
  target: string;
  forbidden: string[]; // normalized words/phrases
  category?: string;
}
```

Player Interface

TypeScript

```
None
// Player
interface Player {
  id: string; // socket id or uuid
  name: string;
  score: number;
  guessesUsed: number; // reset each round
  isSpeaker: boolean;
}
```

Room State (Server)

TypeScript

```
None
// Room state (server)
interface Room {
  id: string;
  players: Player[]; // length 4
  deck: Card[]; // shuffled
  currentCard?: Card;
  currentSpeakerIndex: number;
  roundNumber: number;
  roundActive: boolean;
  guessesRemaining: Record<string, number>; // playerId ->
remaining (6)
  clueCount: number; // 0..4
}
```

---

4. Scoring Rules 🥇

Scoring is based on the speed of the correct guess and adherence to the rules.

Event	**Speaker Points**	**Guesser Points**	**Notes**
Correct guess after clue 1 or 2	**+6**	**+4**	Rewards fast clues/guesses.
Correct guess after clue 3 or 4	**+5**	**+3**	Reduced reward for slower success.

Forbidden word spoken	**−5**	**0**	Penalty applied to the Speaker.
Guesser uses all 6 guesses	**0**	**-1 (Optional)**	Optional penalty for a Guesser exhausting their limit.
Bonus: Unused clues	**+1 per unused clue**	**0**	Bonus for the Speaker for being efficient.
Bonus: Unused guesses	**0**	**+0.5 per unused guess**	Bonus for each Guesser for being quick.

Decision Point 1: Forbidden Word Rule

The server logic currently deducts points and notifies. Recommended decision: Continue the round with a penalty unless all 4 clues have been given. This allows the chance for a correct guess after the penalty.

---

5. Socket Events (Contract) 🔌

Client $\rightarrow$ Server Events

Event Name	**Payload Example**	**Description**
join-room	{ roomId, name }	Player joins an existing room.
create-room	{ name }	Player creates a new room.
start-game	{ roomId }	Initiates the game when the room is full.
start-round	*none*	Speaker signals the start of the mic recording/clue.

speaker-transcript	{ roomId, text }	Broadcasts a clue segment transcribed by the client.
guesser-guess	{ playerId, guessText }	Guesser submits a guess.

Server $\rightarrow$ Client Events

Event Name	**Payload Example**	**Description**
room-updated	{ roomState }	Broadcasts changes in players/status to all clients in the room.
card-assigned	{ card }	**Only emitted to the Speaker's socket.**
clue-broadcast	{ speakerId, transcript, clueNumber }	Broadcasts the validated clue to all clients.
forbidden-detected	{ speakerId, matchedWords }	Notifies all clients of a forbidden word violation.
guess-result	{ playerId, correct: boolean, remainingGuesses }	Notifies the guesser and checks against target.
round-ended	{ summary }	Broadcasts round results and prepares for the next round.

---

6. Pages & Components (Next.js) 🖥️

Path / Component	**Role / Description**

/app/(auth)/join/page.tsx	Landing page: Allows user to enter their name and join or create a room.
/app/room/[id]/page.tsx	Room lobby: Displays player list and a "Start Game" button (visible to host/first player).
/app/room/[id]/game/page.tsx	**Primary Game UI:** Renders the specific Speaker or Guesser view based on the player's role.
CardView.tsx	Shows the target/forbidden words (**Speaker-only view**).
MicControl.tsx	Button/UI to activate/deactivate the microphone for the Speaker.
GuessInput.tsx	Input area for Guessers to submit their guesses (text or voice).
ScoreBoard.tsx	Shared component showing current scores and the active Speaker.

---

7. Key Implementation Notes + Code Snippets ⚙️

7.1 Socket Server (Node.js/Express/Socket.IO)

The server is responsible for maintaining the central **Room state** in memory.

- **Socket.IO Setup:** Uses standard setup with CORS enabled.
- **State Management:** `const rooms = new Map();` is used for in-memory persistence of all room states for the MVP.
- **Core Logic:** Handles `create-room`, `join-room`, `start-game`, `speaker-transcript` (triggers forbidden check), and `guesser-guess` (triggers scoring/round end).

7.2 Frontend: SpeechRecognition Hook

The `useSpeechRecognition` hook abstracts the **Browser Web Speech API** for reliable client-side transcription.

- **Polyfill/Fallback:** Uses (window as any).SpeechRecognition || (window as any).webkitSpeechRecognition.
- **Configuration:** Set to continuous: true for persistent listening and interimResults: interim (set to false for final clue submission).
- **Output:** The onResult callback is responsible for sending the transcript to the server via socket.emit('speaker-transcript', ...) when a complete clue is ready.

7.3 Forbidden Check Utility (utils/forbiddenCheck.ts)

This utility ensures both the clue text and the forbidden words are standardized before comparison.

1. **Normalization:** normalize(s): Converts the string to **lowercase** and **removes punctuation** (replace(/[^\w\s]/g, '')). This makes the comparison less susceptible to casing or minor text errors.
2. **Comparison:** checkForbidden(text, forbidden): Iterates through the forbidden list, checking if the **normalized clue text** includes the **normalized forbidden word/phrase**.

---

8. Responsive Design & Accessibility ♿

- **Technology: Tailwind CSS** will be used for rapid, utility-first styling and responsive breakpoints (mobile-first approach).
- **Layout:**
  - **Mobile:** Vertical stack (Card/Clue box $\rightarrow$ Guess input $\rightarrow$ Collapsible Scoreboard).
  - **Tablet/Desktop:** Two-column layout (Left: Main Play Area; Right: Scoreboard & Player panels).
- **A11y:** Ensure **ARIA roles** are used for interactive elements, with support for **keyboard navigation** and **high-contrast colors** for legibility.

9.🧪 Test Cases for Forbidden Word Game

The tests are organized based on the components: **Utilities**, **Game Flow/State Management**, and **Scoring Logic**.

1. Utility Tests (utils/forbiddenCheck.ts)

These tests validate the text normalization and forbidden word detection functions, which are vital for the game's core mechanic.

ID	Description	Input Text	Input Forbidden List	Expected Output (Matches)	Expected Score Change
UT-101	**Normalization:** Basic case (case-insensitivity).	"Hello, World"	["hello"]	["hello"]	N/A
UT-102	**Normalization:** Punctuation removal.	"Is it a Cat?!"	["cat"]	["cat"]	N/A
UT-103	**Forbidden Check:** Simple match.	"The animal is a dog"	["dog", "pup"]	["dog"]	-5
UT-104	**Forbidden Check:** No match.	"It's a small object."	["big", "large"]	[]	No change
UT-105	**Forbidden Check:** Match within a word (Edge Case).	"The target is a carpet."	["car"]	["car"]	-5 (This validates the includes method logic; may need refinement if whole-word matching is desired)

UT-106	**Forbidden Check:** Multiple matches.	"Fast car, big house, big dog."	["house", "dog", "car"]	["car", "house", "dog"] (Order may vary)	-5
UT-107	**Normalization:** Text containing numbers/symbols.	"The year is 2025."	["2025"]	[] (Since normalization removes non-word characters, use case dependent)	N/A

---

2. Socket Event & Game State Tests (Server)

These integration tests simulate client-server interactions using the defined socket contract.

ID	**Event(s) Fired**	**Pre-Condition / Setup**	**Expected Server Response / State Change**	**Expected Client Broadcast**
ET-201	create-room	No rooms exist.	New room created (Room.id generated). Player added.	room-updated (to creator only)
ET-202	3 players join-room	Room exists, 1 player present.	Room.players length is 4.	4 x room-updated (to all players)
ET-203	start-game	Room has 4 players.	Room.roundNumber = 1. Room.currentSpeakerIndex = 0. Room.currentCard is set.	game-started (to all); card-assigned (to Speaker only)

ET-204	speaker-transcript	Speaker sends **valid** clue. (clueCount = 0)	Room.clueCount increments to 1.	clue-broadcast (to all)
ET-205	speaker-transcript	Speaker sends **forbidden** clue.	Speaker's score decreases by 5.	forbidden-detected; score-updated
ET-206	guesser-guess	Guesser sends **wrong** guess. (guessesUsed=0)	Player.guessesUsed increments to 1. Room.guessesRemaining decreases by 1.	guess-result (correct: false)
ET-207	disconnect	Player 2 disconnects during the game.	Player 2 removed from Room.players. Room potentially disabled if required player count not met.	room-updated (to remaining players)

3. Scoring & Round Logic Tests

These tests validate the point calculation (computePoints) and the state transitions upon success or failure.

ID	Action	Context (Clue Count / Guesses Used)	Target Word	Expected Speaker Score Change	Expected Guesser Score Change	Expected Round State
SC-301	**Correct Guess**	Clue Count: 1	Matched	**+6**	**+4**	**Round Ended**
SC-302	**Correct Guess**	Clue Count: 4	Matched	**+5**	**+3**	**Round Ended**

SC-303	**Guesser Fails**	Guesser A: 6th wrong guess.	Not Matched	No change	No change (or -1 if optional penalty is active)	Round continues (unless all players exhausted)
SC-304	**Round Exhausted**	4 players, all used 6 guesses.	Not Matched	No change	No change	**Round Ended** (New Speaker selected)
SC-305	**Forbidden Penalty**	Speaker uses forbidden word.	N/A	**-5**	No change	Round continues (based on recommended decision)

Scoring Functions (computePoints details):

- If clueCount is 1 or $2 \implies$ Speaker $+6$, Guesser $+4$.
- If clueCount is 3 or $4 \implies$ Speaker $+5$, Guesser $+3$.

---

4. Frontend (UI/UX) Tests

These manual tests ensure the client-side user experience and Speech API integration are functional.

ID	**Step**	**Role**	**Expected Behavior / Observation**
FE-401	Open site via **HTTP**	Any	Browser should **prevent mic access** and display an error/warning about HTTPS requirement.
FE-402	Click "Start Round"	Speaker	**Mic indicator activates.** Speech recognition starts and displays interim results.
FE-403	Speak a clue (e.g., "The target is red")	Speaker	The transcribed text appears on the speaker's screen, and then broadcasts to Guessers.

FE-404	Speak a forbidden word	Speaker	Transcription stops. Speaker receives a visible "Forbidden Word Detected" alert. Guesser scores update (-5 penalty).
FE-405	Submit a text guess	Guesser	The guess text is cleared. Remaining guess counter decreases. The result (correct/wrong) flashes briefly on the screen.
FE-406	View on Mobile (Portrait)	Any	UI adjusts to the vertical stack layout. Text is legible. Tap targets are large.