Web-Based Family Feud Survey System Analysis and Functional Design

Prepared by ChatGPT (GPT-5) for Neba Eric

This document provides a comprehensive analysis and design overview for developing a **web-based Family Feud-style survey system**. The system aims to simulate the popular game show format in which participants guess the most popular answers to survey questions, as collected from a predefined sample size (e.g., 100 people). The application will combine fun gameplay with real-time interactivity, automated scoring, and engaging audio-visual feedback.

1. Core Concept and Objective The main goal of the system is to replicate the *Family Feud* gameplay experience on a web platform. The survey assistant (game control system) acts as the central logic hub — it records user answers, tracks scores, triggers sound effects, and displays real-time updates. The assistant interacts with the host interface, database, and player interface seamlessly. **Core objectives:** - Manage survey data (questions and answer frequency counts). - Dynamically calculate and assign points based on how common an answer is. - Provide real-time feedback (e.g., right/wrong answer sounds). - Display question timers and team scores. - Handle sound triggers and progress updates during gameplay.

2. Survey Data Collection and Structure Before the game begins, a sample size of about 100 participants is surveyed for each question. The answers are stored in a database, sorted by frequency. The frequency determines how many points a given answer earns during gameplay. **Database Example Structure:** | Question ID | Question Text | Answer | Frequency | Points | |-------------|---------------|--------|-----------|--------| | 1 | Name something you find in a kitchen | Spoon | 50 | 50 | | 1 | Name something you find in a kitchen | Knife | 30 | 30 | | 1 | Name something you find in a kitchen | Plate | 20 | 20 | **Point Allocation Rule:** - Each answer’s frequency (out of 100) equals its score value. - Top frequent answers yield higher scores.

3. Gameplay Mechanics and Flow - The host reads a question to the contestants. - Players provide answers within a given time (e.g., 20 seconds). - The assistant checks if the answer exists in the database. - If correct: The system reveals the answer and awards the corresponding points. - If incorrect: The system plays a “wrong answer” sound and deducts a chance. - After three strikes or time expiration, control passes to the opposing team. - Scores are tallied and displayed dynamically. **Game Rounds:** - The game progresses through several rounds with different point multipliers (1x, 2x, 3x). - Final round may include “Fast Money” where only two players respond to multiple questions in a limited time.

4. System Components **a. Frontend (Player/Host Interface):** - Developed using HTML, CSS, and JavaScript (React or Vue recommended). - Displays questions, timers,

and scoreboards. - Includes audio and visual feedback animations (e.g., sound effects, lights, etc.). **b. Backend (Game Logic):** - Powered by Node.js or Python (Flask/Django). - Handles question retrieval, score calculation, and state management. - Communicates with frontend via RESTful API or WebSocket for real-time updates. **c. Database:** - MySQL or MongoDB recommended. - Stores survey questions, answer frequencies, and player/team records. **d. Sound and Effects:** - Triggered automatically on correct/incorrect answers. - Plays countdown and buzzer sounds as appropriate.

5. Extended Rules and Functionalities - Each team has 3 attempts (strikes) per round. - A timer limits how long a player can answer each question. - Points accumulate per round and carry over to final scoring. - Admin dashboard allows creating and editing survey questions and answers. - The system should support sound muting, pausing, and score resetting. - All activity should be logged for analysis and debugging.

6. Suggested Technology Stack | Layer | Technology | Purpose | |--------|-------------|----------| | Frontend | React.js / Vue.js | Build dynamic, real-time UI | | Backend | Node.js / Flask / Django | Game logic, API, and control | | Database | MySQL / MongoDB | Store survey data and results | | Real-time Updates | WebSocket / Socket.IO | Live question updates and scoring | | Audio Control | HTML5 Audio API | Sound triggers and timing | | Hosting | AWS / Firebase / Vercel | Deployment and hosting |

7. Future Enhancements - Add AI-driven question generation from real survey data. - Implement multilingual support for wider audiences. - Enable live multiplayer gameplay via online connections. - Create analytics dashboard for survey trends and player behavior.

Conclusion This system combines entertainment and technology to replicate the Family Feud experience on the web. By integrating real-time feedback, survey-based scoring, and an interactive interface, this web application will engage users while maintaining the authentic spirit of the original show.