Project Report

Word Guessing Game

1. Project Title

Word Guessing Game – A JavaScript-Based Browser Game

2. Introduction

This project involves the design and implementation of an interactive browser-based game called the Word Guessing Game. The game challenges the user to guess a hidden word based on a shuffled version or clue. It is built entirely using front-end web technologies: HTML, CSS, and JavaScript. The game incorporates three levels of difficulty, a life system (3 lives per game), and engaging sound effects to provide a better user experience.

3. Objectives

- Developing a functional game that runs in any modern web browser.
- Incorporating difficulty levels to provide varied challenges.
- Implementing a simple UI/UX with feedback and sound.
- Reinforcing understanding of DOM manipulation and JavaScript game logic.

4. Tools and Technologies Used

- HTML – Page structure and layout
- CSS – Styling and visual design
- JavaScript – Game logic, interaction, and sound integration
- Audio (.mp3) – Sound effects for game actions

5. System Description

5.1 Game Flow:

1. The user selects a difficulty level: Normal, Medium, or Hard.
2. The system randomly selects a word from a predefined list based on the selected difficulty.
3. A shuffled version of the word is shown as a hint.
4. The user enters a guess in the input field.
5. If the guess is correct:
   - A success message is displayed.
   - A new word is presented.
   - A "correct" sound is played.
6. If the guess is incorrect:
   - A life is deducted.
   - A "wrong" sound is played.
7. When lives reach zero:
   - A "Game Over" message is displayed.

- A "game over" sound is played.
- Input is disabled until reset.

5.2 Difficulty Levels:
- Normal: Simple, short, common words (e.g., cat, dog, sun).
- Medium: Slightly longer and more challenging words (e.g., apple, house, plane).
- Hard: Long or less common words (e.g., dinosaur, microscope).

6. Implementation Overview

6.1 HTML:
Provides structural elements like heading, input box, lives display, difficulty selector, and buttons.

6.2 CSS:
Used for styling the interface with a clean, simple layout. Provides visual cues and spacing to enhance UX.

6.3 JavaScript:
Handles core functionality:
- Selecting and shuffling words.
- Monitoring user input.
- Updating lives and messages.
- Playing sound effects.
- Enabling/disabling game flow.

7. Key Features

- Three levels of difficulty.
- Life counter and restart mechanism.
- Shuffled word hints for challenge.
- Sound feedback for interactions.
- Responsive UI for various screen sizes.

8. Challenges Faced

- Ensuring fair shuffling while keeping the word guessable.
- Preventing repeated words during sessions.
- Managing sound playback for rapid events.
- Handling DOM updates smoothly for user feedback.

9. Future Enhancements

- Add a scoring system and high score tracking.
- Include actual hints instead of word scrambling.
- Provide timer-based challenge mode.
- Add background music and sound on/off toggle.
- Store words in external JSON or connect to an API.

10. Limitations

1. **Static Word List**
   The words used for each difficulty level are hardcoded. This limits variety and replay value.
2. **No Hint Mechanism Beyond Shuffling**
   Players only see a shuffled version of the word—no contextual or semantic hints are given.
3. **No Score Tracking**
   The game doesn't record or display the player's performance over time, such as high scores or win streaks.
4. **Limited Sound Feedback**
   Only basic sound effects are included (correct, wrong, game over). No options for muting or customizing sound.
5. **No Timer or Time-Based Challenge**
   There's no countdown or urgency element that would increase difficulty or excitement.
6. **No Input Validation or Sanitization**
   The system may not gracefully handle unusual inputs (e.g., empty strings, special characters).
7. **Non-Persistent State**
   The game doesn't store progress or history. Refreshing the page resets everything.
8. **Single Language Support**
   The game only supports English. There's no multilingual support.
9. **No Accessibility Features**
   The game doesn't include ARIA labels, keyboard navigation support, or screen reader compatibility.
10. **Basic UI Design**
    While functional, the UI is minimal and may not be engaging for all users, especially on mobile devices.

11. Conclusion

The Word Guessing Game successfully meets its design objectives by delivering an engaging and interactive experience. It demonstrates the effective use of front-end web technologies for game development and provides a strong foundation for further enhancements in complexity and interactivity. This project also serves as a valuable exercise in applying JavaScript for real-time user interaction and game logic.