Technical Report: Right vs Wrong — Kids Learning Game

Page 1: Introduction and Project Overview

1.1 Project Purpose and Scope

Right vs Wrong is an interactive educational web application designed to teach children ethical decision-making through scenario-based learning. The application presents users with video scenarios followed by multiple-choice questions that challenge them to identify correct versus incorrect behaviors. Built as a single-page application using vanilla HTML, CSS, and JavaScript, the game incorporates gamification elements to maintain engagement while delivering educational content.

1.2 Technical Architecture

The application follows a client-side architecture with all game logic executed in the browser. Key architectural components include:

- **Frontend Layer**: Pure HTML5, CSS3, and vanilla JavaScript
- **Data Persistence**: Browser local Storage API for saving progress and settings
- **External Integration**: YouTube embed API for video content
- **Security Model**: Client-side password hashing for teacher controls

The application maintains complete state management within the browser, requiring no server-side components for core functionality, making it easily deployable to any static web hosting service.

1.3 Target Audience and Use Cases

Primary users are children aged 6-12, with secondary users being parents and educators who can customize content through the teacher settings panel. The application supports both individual learning and classroom settings, with progress tracking that allows multiple children to use the same device while maintaining separate achievement records.

Page 2: Design System and User Interface

2.1 Visual Design Principles

The interface employs a child-friendly design language with specific considerations for the target demographic:

Color Psychology:

- Primary accent colors (orange/yellow) evoke energy and positivity
- High contrast between correct (green) and incorrect (red) states
- Muted background tones reduce visual fatigue during extended use
- Consistent color coding throughout the application

Typography:

- Comic Sans MS as primary font for its readability and child-friendly appearance
- Clear hierarchy with appropriate font sizing for different content types
- Support for accessibility through large text toggle functionality

2.2 Responsive Layout System

The application implements a sophisticated responsive grid that adapts to various screen sizes:

```
main.app {
  display: grid;
  grid-template-columns: 1fr 420px;
  gap: 20px;
}

@media (max-width: 900px) {
  main.app {
    grid-template-columns: 1fr;
  }
}
```

This CSS Grid implementation ensures optimal viewing on both desktop and mobile devices, with the sidebar collapsing below the main content area on smaller screens. The flexible layout maintains usability across tablets, smartphones, and desktop computers.

2.3 Component-Based UI Architecture

The interface is constructed from reusable UI components:

- **Card components** for content containers with consistent shadow and border properties
- **Option components** with hover and selection states
- **Modal system** for overlays and settings panels
- **Progress indicators** with animated fill transitions
- **Sticker reward system** with floating animation effects

Page 3: Core Game Logic and State Management

3.1 Application State Architecture

The game implements a comprehensive state management system that tracks multiple dimensions of user progress:

```
let state = {
  order: [], // Randomized question order
  idx: 0, // Current question index
  score: 0, // Current game score
  stickers: [], // Collected reward stickers
  streak: 0, // Consecutive correct answers
  best: 0, // All-time best score
  timer: null, // Timer reference
  timerSec: 10, // Timer duration
  mute: false, // Sound preference
  childName: "", // User identification
  hintPenalty: 1, // Point deduction for hints
  shuffleQuestions: true, // Game configuration
  lastPlayed: {} // Progress persistence
};
```

3.2 Question Flow and Randomization

The game engine implements sophisticated question management:

Pool Generation:

```
function buildPool() {
  let pool = DATA.slice();
  if (document.getElementById("shuffleQ").checked) {
    pool = shuffle(pool);
  }
  state.order = pool;
}
```

Answer Processing:

- Options are dynamically shuffled to prevent pattern recognition
- Immediate visual feedback with color-coded correct/incorrect states
- Fact collection only on correct answers to reinforce learning
- Streak tracking with localStorage persistence

3.3 Scoring and Reward System

The application employs multiple reinforcement mechanisms:

- **Base Scoring**: 1 point per correct answer
- **Bonus Systems**: Streak multipliers and timer bonuses
- **Sticker Economy**: Random emoji rewards for correct answers
- **Hint Economy**: Strategic penalty system (1 point/sticker deduction)
- **Achievement Tracking**: Personal best records with persistent storage

Page 4: Technical Implementation Details

4.1 Data Persistence Strategy

The application leverages the Web Storage API with careful key management:

```
// Primary data stores
localStorage.setItem("rvs_quiz", JSON.stringify(DATA)); // Question
bank
localStorage.setItem("rvs_settings", JSON.stringify(SETTINGS)); //
User preferences
localStorage.setItem("rvs_stickers", JSON.stringify(state.stickers));
// Rewards
localStorage.setItem("rvs_streak", state.streak); // Progress tracking
localStorage.setItem("rvs_best", state.best); // Achievement records
localStorage.setItem("rvs_childName", state.childName); // User
identification
localStorage.setItem("rvs_facts", JSON.stringify(facts)); // Learning
outcomes
```

This modular storage approach allows independent management of different data types and supports multiple user profiles on the same device.

4.2 Security Implementation

For the teacher controls panel, the application implements client-side security:

Password Hashing:

```
async function hashCode(str) {
  const enc = new TextEncoder().encode(str);
  const hashBuffer = await crypto.subtle.digest('SHA-256', enc);
  const hashArray = Array.from(new Uint8Array(hashBuffer));
  return hashArray.map(b => b.toString(16).padStart(2, '0')).join('');
}
```

The default password "67" is hashed using SHA-256 and stored for verification. While this provides basic protection, it's important to note this is client-side security suitable only for preventing accidental access rather than determined attackers.

4.3 Animation and Feedback Systems

The user experience is enhanced through multiple animation systems:

CSS Transitions:

- Smooth progress bar fills with easing functions
- Hover effects with transform translations
- Modal entrances with opacity transitions

JavaScript Animations:

- Confetti celebration using Canvas API
- Floating sticker rewards with keyframe animations
- Buddy character reactions (jumping, shaking)
- Timer countdown with visual updates

Audio Feedback:

- Distinct sounds for correct/incorrect answers and game completion
- Mute functionality with persistent preference storage
- External audio sources from Google Sounds library

Page 5: Accessibility and Educational Design

5.1 Accessibility Features

The application incorporates multiple accessibility considerations:

Semantic HTML:

- Proper ARIA labels for screen readers (`aria-live`, `aria-hidden`, `aria-modal`)
- Logical heading structure and landmark regions
- Form labels associated with inputs

Keyboard Navigation:

- Full tab-index support throughout the interface
- Focus indicators for interactive elements
- Keyboard-operable modals and controls

Visual Accessibility:

- High color contrast ratios exceeding WCAG guidelines
- Scalable text through CSS relative units
- Non-color-dependent indicators (icons + text)

5.2 Educational Psychology Principles

The game design incorporates established learning principles:

Scaffolded Learning:

- Difficulty progression from easy to medium questions
- Immediate feedback with explanatory facts
- Hint system that encourages problem-solving before revealing answers

Motivational Design:

- Variable reward schedule through random sticker distribution
- Progress visualization with filling level bars
- Achievement markers (streaks, personal bests)
- Positive reinforcement through celebratory animations

Metacognitive Support:

- Fact collection system that reinforces learning objectives
- Progress tracking that enables self-assessment
- Non-punitive error correction with explanatory feedback

5.3 Customization Framework

The teacher settings panel provides extensive customization:

Content Management:

- Full CRUD operations for question bank
- JSON import/export for content sharing
- Search and filtering within question library

Gameplay Configuration:

- Timer toggle for difficulty adjustment

- Question shuffling to prevent memorization
- Reset functionality for multiple user support

Page 6: Challenges, Learnings, and Future Enhancements

6.1 Technical Challenges and Solutions

State Synchronization: Managing multiple interdependent state variables (score, stickers, streak) required careful update sequencing to maintain consistency across localStorage and UI representations.

Cross-browser Compatibility: The Web Crypto API for password hashing required fallback considerations for older browsers, though the target demographic primarily uses modern browsers.

Performance Optimization: Large question banks necessitated efficient search and filtering algorithms, implemented through debounced input handlers and selective re-rendering.

6.2 Key Learnings

Educational Technology Insights:

- Immediate feedback is crucial for concept retention in children
- Reward variability maintains engagement better than predictable systems
- Visual and audio reinforcement together improve learning outcomes

Technical Implementation Insights:

- localStorage provides sufficient persistence for edutainment applications
- CSS Grid offers superior layout control for responsive educational content
- Vanilla JavaScript can effectively manage complex state without frameworks

User Experience Insights:

- Children respond better to character-based feedback (buddy system)
- Progress visualization significantly increases completion rates

- Sound effects should be optional but enhance the experience when enabled

6.3 Potential Enhancements

Technical Improvements:

- Service Worker implementation for offline functionality
- Cloud synchronization for cross-device progress tracking
- Analytics integration for learning outcome assessment

Content Expansion:

- Multiple difficulty tiers with adaptive questioning
- Themed question packs (safety, hygiene, environmental awareness)
- Multilingual support for diverse educational settings

Feature Additions:

- Avatar customization for increased personalization
- Social features (with appropriate privacy safeguards)
- Parent/teacher dashboard for progress monitoring
- Accessibility enhancements for children with special needs

6.4 Conclusion

Right vs Wrong successfully demonstrates how modern web technologies can create engaging educational experiences without complex infrastructure. The application balances educational effectiveness with entertainment value through careful implementation of game mechanics and learning principles. The technical architecture provides a solid foundation for future enhancements while maintaining simplicity and accessibility. This project highlights the potential of browser-based applications to deliver meaningful educational content in an accessible, scalable format.

The complete implementation shows that sophisticated educational software can be built with standard web technologies, providing a template for future educational game development that prioritizes both learning outcomes and user engagement.