網頁程式設計
期末專案成果

組員：
01157123 - 胡家豪
01257004 - 林柏陞

後端 API：https://wordle-game-backend-v2.onrender.com
前端遊戲網站：https://garyHu951.github.io/wordle-game

WORDLE PLUS

多元猜字競賽平台

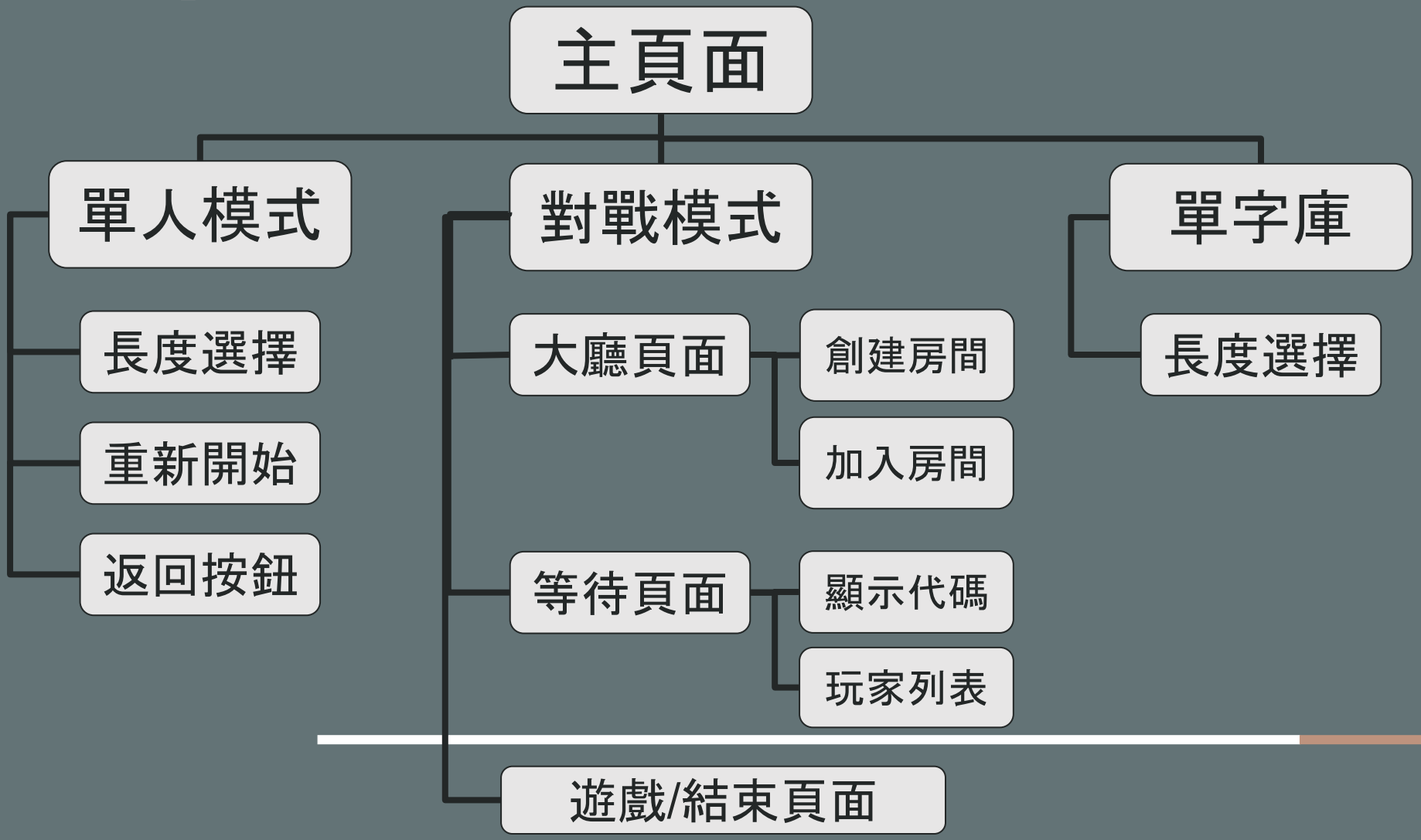
- 單人模式---可自選4~7個字母的難度進行猜字
- 多人模式---可創建房間與朋友進行連線對戰

創作動機

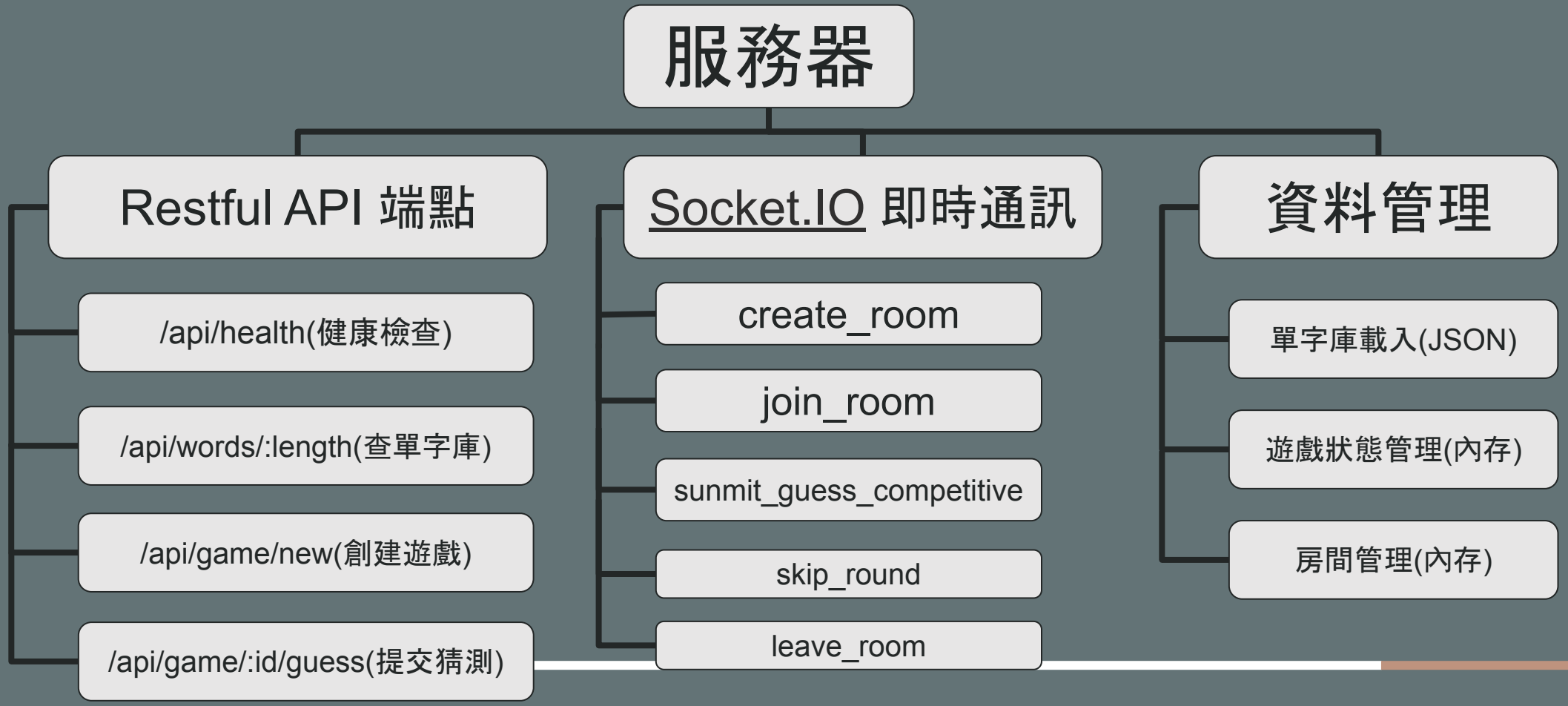
遊戲內容太單調

- 本專案靈感來自於風靡全球的 Wordle 遊戲，但我希望創造一個更具特色和互動性的版本。原版 Wordle 雖然簡潔優雅，但缺乏多人互動和視覺豐富度，因此我決定開發一個結合復古像素風格和現代 Web 技術的增強版本。

前端架構


主頁面
單人模式
長度選擇
重新開始
返回按鈕
對戰模式
大廳頁面
創建房間
加入房間
等待頁面
顯示代碼
玩家列表
遊戲/結束頁面
單字庫
長度選擇

後端架構


服務器
Restful API 端點
/api/health(健康檢查)
/api/words/:length(查單字庫)
/api/game/new(創建遊戲)
/api/game/:id/guess(提交猜測)
Socket.IO 即時通訊
create_room
join_room
sunmit_guess_competitive
skip_round
leave_room
資料管理
單字庫載入(JSON)
遊戲狀態管理(內存)
房間管理(內存)

前端技術

- React
  - 函數式組件與 Hooks
  - useState, useEffect, useRef 狀態管理
  - 組件化開發架構
- Vite
  - 快速開發伺服器
  - 熱模組替換 (HMR)
  - 優化的生產構建
- Tailwind CSS
  - 實用優先的 CSS 框架
  - 響應式設計
  - 自定義像素風格類別
- Socket.IO Client
  - 即時雙向通訊
  - 自動重連機制
  - 事件驅動架構
- Lucide React
  - 現代 SVG 圖標庫
  - 輕量級且可自定義

後端技術

- Node.js
  - 非同步 I/O 處理
  - 事件驅動架構
  - 高效能 JavaScript 運行環境
- Express.js
  - RESTful API 框架
  - 中間件支援
  - 路由管理
- Socket.io
  - 即時通訊協議
  - 房間管理
  - 廣播機制
- Mongoose
  - MongoDB 物件文檔映射
  - 資料驗證
  - 查詢建構

音效管理系統

單例模式的音效管理器，
統一處理所有音效播放

- 使用 HTML5 Audio API 進行音效控制
- 實現音效預載入機制，提升用戶體驗
- 錯誤處理機制，防止音效播放失敗影響遊戲

```
class AudioManager {
  constructor() {
    this.sounds = {};            // 音效檔案快取
    this.isMuted = false;        // 靜音狀態
    this.volume = 0.7;           // 主音量
    this.currentBgMusic = null; // 當前背景音樂
  }

  // 預載入所有音效檔案，避免播放時的延遲
  preloadSounds() {
    Object.entries(this.soundFiles).forEach(([key, path]) => {
      const audio = new Audio(path);
      audio.preload = 'auto';
      this.sounds[key] = audio;
    });
  }

  // 播放音效，支援音量控制和錯誤處理
  play(soundName, options = {}) {
    if (this.isMuted) return;
    const sound = this.sounds[soundName];
    sound.currentTime = 0; // 重置播放位置
    sound.volume = options.volume || this.volume;
    sound.play().catch(error => console.warn('播放失敗:', error));
  }
}
```

遊戲邏輯核心

實現 Wordle 遊戲的核心邏輯，
判斷每個字母的狀態

- 時間複雜度：$O(n^2)$，其中 n 是單字長度
- 空間複雜度：O(n)，用於存儲結果和標記陣列
- 邏輯正確性：兩輪掃描確保字母狀態判斷的準確性

```
function checkGuess(guess, answer) {
  const result = [];
  const answerArray = answer.split('');
  const guessArray = guess.split('');
  const used = new Array(answer.length).fill(false);

  // 第一輪：標記完全正確的字母（綠色）
  guessArray.forEach((letter, i) => {
    if (letter === answerArray[i]) {
      result[i] = 'correct';
      used[i] = true;
    }
  });

  // 第二輪：標記存在但位置錯誤的字母（黃色）
  guessArray.forEach((letter, i) => {
    if (result[i]) return; // 跳過已標記的
    const foundIndex = answerArray.findIndex((l, idx) =>
      l === letter && !used[idx]
    );
    if (foundIndex !== -1) {
      result[i] = 'present';
      used[foundIndex] = true;
    } else {
      result[i] = 'absent'; // 不存在（灰色）
    }
  });

  return result;
}
```

即時通訊系統

基於事件驅動的即時通訊,
實現多人對戰功能

-房間隔離:使用 Socket.IO 的 room 功能實現玩家隔離

- 狀態同步:即時廣播遊戲狀態變化給所有相關玩家

- 錯誤處理:連接斷開時的清理機制

```
// 後端房間管理
io.on('connection', (socket) => {
  // 創建房間
  socket.on('create_room', ({ wordLength }) => {
    const roomCode = generateRoomCode(); // 生成6位房間代碼
    rooms[roomCode] = {
      id: roomCode,
      wordLength: parseInt(wordLength),
      players: {},
      currentWords: {}, // 每個玩家的專屬單字
      status: 'waiting'
    };

    rooms[roomCode].players[socket.id] = {
      id: socket.id,
      score: 0,
      name: 'Player 1'
    };
    socket.join(roomCode);
    socket.emit('room_created', { roomCode });
  });

  // 處理對戰猜測
  socket.on('submit_guess_competitive', ({ roomCode, guess }) => {
    const room = rooms[roomCode];
    const playerWord = room.currentWords[socket.id];
    const result = checkGuess(guess.toUpperCase(), playerWord);
    const isCorrect = guess.toUpperCase() === playerWord;

    if (isCorrect) {
      room.players[socket.id].score += 5;
      // 廣播勝利訊息給所有房間成員
      io.to(roomCode).emit('round_winner', {
        winnerId: socket.id,
        word: playerWord,
        points: 5,
        updatedPlayers: room.players
      });
    }
  });
});
```

狀態管理系統

使用 React Hooks 實現
複雜的遊戲狀態管理

- 單一資料流：狀態由上而下傳遞，事件由下而上冒泡
- 副作用管理：使用 useEffect 處理 API 呼叫和音效播放
- 效能優化：避免不必要的重新渲染

```
const SinglePlayerGame = ({ onBack }) => {
  // 遊戲核心狀態
  const [gameId, setGameId] = useState(null);
  const [guesses, setGuesses] = useState([]);
  const [currentGuess, setCurrentGuess] = useState('');
  const [gameOver, setGameOver] = useState(false);
  const [won, setWon] = useState(false);

  // 處理猜測提交
  const handleSubmitGuess = async () => {
    if (currentGuess.length !== wordLength) return;

    setLoading(true);
    try {
      const response = await fetch(`${API_URL}/game/${gameId}/guess`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ guess: currentGuess })
      });

      const data = await response.json();
      if (data.success) {
        setGuesses(data.guesses);
        setGameOver(data.gameOver);
        setWon(data.won);
        setCurrentGuess('');

        // 播放對應的音效
        playResultSounds(data.result);
      }
    } catch (error) {
      console.error('提交猜測失敗:', error);
    }
    setLoading(false);
  };
};
```

字母狀態追蹤

分析所有猜測記錄，
統計每個字母的使用狀態

- 優先級處理：正確 > 存在 > 不存在的優先級邏輯
- 集合運算：使用 Set 資料結構避免重複
- 即時更新：每次猜測後自動重新計算

```
const LetterStatusTracker = ({ guesses }) => {
  const getLetterStatus = () => {
    const correct = new Set();    // 正確位置的字母
    const present = new Set();    // 存在但位置錯誤的字母
    const absent = new Set();     // 不存在的字母

    guesses.forEach(guess => {
      guess.word.split('').forEach((letter, i) => {
        if (guess.result[i] === 'correct') {
          correct.add(letter);
          present.delete(letter); // 從 present 中移除
        } else if (guess.result[i] === 'present' && !correct.has(letter)) {
          present.add(letter);
        } else if (guess.result[i] === 'absent' &&
                   !correct.has(letter) &&
                   !present.has(letter)) {
          absent.add(letter);
        }
      });
    });

    return {
      correct: Array.from(correct),
      present: Array.from(present),
      absent: Array.from(absent)
    };
  };
```

字母動畫實現

CSS 關鍵幀動畫
配合 JavaScript 類別控制

- 關鍵幀動畫：使用 `steps()` 函數實現像素風格的跳躍動畫

- 時序控制：JavaScript 控制動畫觸發時機和順序

- 性能優化：動畫結束後自動清理 CSS 類別

```
// 動畫觸發邏輯
const playResultSounds = (result) => {
  result.forEach((status, index) => {
    setTimeout(() => {
      const cell = document.querySelector(`[data-cell="${index}"]`);

      // 根據結果添加對應動畫類別
      if (status === 'correct') {
        cell.classList.add('animate-strong-impact');
        playSound('correctCell');
      } else if (status === 'present') {
        cell.classList.add('animate-medium-impact');
        playSound('presentCell');
      } else {
        cell.classList.add('animate-weak-impact');
        playSound('absentCell');
      }

      // 動畫結束後移除類別
      setTimeout(() => {
        cell.classList.remove('animate-strong-impact',
                              'animate-medium-impact',
                              'animate-weak-impact');
      }, 600);
    }, index * 100); // 依序播放動畫
  });
};
```

網站特色

遊戲玩法創新

- Wordle Plus提供4~7個字母的單字做選擇
- 新增多人對戰模式
- 新增字母追蹤功能，即時顯示字母使用狀態

美術優化

- 新增復古電玩像素風遊玩畫面動畫及遊戲音效，讓玩家擁有更好的遊玩體驗

技術架構優勢

- 使用前後端分離，易於維護
- 提供穩定的即時互動通訊
- React Hooks實現高效狀態管理
- 完整的錯誤捕獲和用戶反饋

遊戲實機演示

後端 API：https://wordle-game-backend-v2.onrender.com

前端遊戲網站：https://garyHu951.github.io/wordle-game

分工表

胡家豪 (60%)	主程式架構設計、系統架構優化、模組化組件系統、單字庫查詢功能、更新猜測檢查演算法、設計復古像素藝術風格、遊戲動畫特效製作、遊戲音效製作、遊戲前後端部署(GitHub、Render)、簡報製作
林柏陞 (40%)	單字量擴充、單字長度選擇、玩家連線功能（房間系統、即時通訊）、簡報製作、遊戲說明影片製作

THANK YOU FOR LISTENING