# **ADHD儿童注意力训练网页技术设计文档 (TDD)**

## **1. 引言**

### **1.1 文档目的**

本文档旨在为ADHD儿童注意力训练网页应用提供详细的技术设计方案，指导开发团队进行前端、后端（数据库）以及部署运维的实现。

### **1.2 范围**

本设计文档涵盖了网页应用的初始版本功能，包括用户账号体系、网页基础框架、简介页面和“按顺序点击数字”游戏页面的技术实现细节。

## **2. 架构概述**

本应用将采用前后端分离的架构。

* **前端：** 基于React框架，负责用户界面渲染、用户交互逻辑和与后端API的通信。
* **后端（数据库）：** 采用Google Firebase/Firestore，提供用户认证、数据存储（用户数据、游戏成绩、配置信息）和实时数据同步能力。Firebase的托管服务也方便部署。
* **部署与运维：** 前端应用部署在腾讯云静态托管服务上，通过GitHub Actions实现CI/CD，结合Cursor进行代码管理和更新。

**图1：系统架构图**

## **3. 技术选型**

* **前端框架：** React.js (版本：最新稳定版)
  + **UI组件库/样式框架：** Tailwind CSS (版本：最新稳定版)
  + **状态管理：** React Context API 或 Zustand (根据项目复杂性选择，初期可优先考虑Context)
  + **图标库：** Lucide React (或 Font Awesome/Phosphor Icons，根据具体图标需求选择)
* **后端/数据库：** Google Firebase (版本：最新稳定版)
  + **认证服务：** Firebase Authentication
  + **数据库：** Cloud Firestore
* **部署：** 腾讯云静态网站托管
* **CI/CD：** GitHub Actions
* **代码管理：** GitHub, Cursor

## **4. 详细设计**

### **4.1 前端设计**

#### **4.1.1 项目结构**

/src  
├── App.js // 主应用组件，负责整体布局和路由管理  
├── index.js // 应用入口  
├── components/ // 可复用UI组件  
│ ├── Sidebar.js // 侧边栏导航组件  
│ ├── Button.js // 统一风格按钮  
│ ├── InputField.js // 统一风格输入框  
│ └── Modal.js // 弹窗组件 (用于隐私条款、错误提示等)  
├── pages/ // 独立页面组件  
│ ├── Auth/ // 认证相关页面  
│ │ ├── LoginPage.js  
│ │ └── RegisterPage.js  
│ ├── IntroductionPage.js // 简介页面  
│ ├── NumberClickGame.js // 数字点击游戏页面  
│ └── Leaderboard.js // 排行榜组件  
├── contexts/ // React Contexts (例如 AuthContext, AppConfigContext)  
│ ├── AuthContext.js  
│ └── AppConfigContext.js  
├── assets/ // 静态资源 (图片等)  
├── utils/ // 工具函数  
│ ├── authService.js // Firebase认证服务封装  
│ ├── firestoreService.js // Firestore数据操作封装  
│ └── gameUtils.js // 游戏相关工具函数  
└── tailwind.config.js // Tailwind CSS配置文件

#### **4.1.2 核心组件设计**

* **App.js:**
  + 作为应用的根组件，负责初始化Firebase。
  + 管理全局认证状态 (userId, isAuthenticated)。
  + 根据用户认证状态和当前路由/页面选择，渲染 Auth 页面或 MainLayout (包含 Sidebar 和 MainContent)。
  + 使用 AuthContext.Provider 封装整个应用，提供认证状态给子组件。
* **Sidebar.js:**
  + 渲染可收起的导航菜单。
  + 根据配置（例如从 AppConfigContext 获取）动态生成菜单项。
  + 点击菜单项时，更新 App.js 中的当前页面状态，从而切换右侧内容。
  + 使用Tailwind CSS实现响应式和可收起效果。
* **LoginPage.js / RegisterPage.js:**
  + 包含用户名、密码输入框和“同意隐私条款”复选框。
  + 调用 authService.js 中的登录/注册方法。
  + 展示加载状态和错误信息。
  + “隐私条款”链接点击后，通过 Modal.js 弹窗显示条款内容或跳转新页面。
* **IntroductionPage.js:**
  + 展示ADHD简介和网站目的的文本和图片。
  + 文本和图片URL通过 AppConfigContext 或直接从Firestore加载。
* **NumberClickGame.js:**
  + **状态管理：**
    - gridNumbers: 存储5x5网格中的数字数组（1-25）。
    - clickedNumbers: 存储已点击的数字，用于校验顺序。
    - timer: 游戏计时器。
    - gameStarted: 布尔值，表示游戏是否开始。
    - gameOver: 布尔值，表示游戏是否结束。
    - currentScore: 当前游戏用时。
  + **游戏逻辑：**
    - **初始化：** 游戏开始时，生成1-25的随机排列，填充到网格中。
    - **点击处理：**
      * 监听每个数字方格的点击事件。
      * 校验点击的数字是否是当前应点击的下一个数字（例如，如果已点击1，则下一个必须是2）。
      * 如果正确，更新 clickedNumbers，并高亮已点击的方格。
      * 如果错误，提供视觉反馈（例如，方格短暂变红）并可能播放错误音效。
    - **计时器：** 使用 setInterval 或 requestAnimationFrame 实现计时，精确到毫秒。
    - **游戏结束：** 当点击完25后，停止计时，计算总用时。
    - **成绩提交：** 调用 firestoreService.js 将成绩提交到Firestore。
  + **UI渲染：** 使用Flexbox或Grid布局渲染5x5的方格。每个方格是一个独立的React组件，显示数字并处理点击事件。
* **Leaderboard.js:**
  + 从Firestore实时监听排行榜数据 (onSnapshot)。
  + 按得分（用时）升序排序并显示。

#### **4.1.3 UI/UX规范**

* **字体：** 统一使用 "Inter" 字体。
* **颜色：** 定义一套主色、辅色和中性色板。
* **圆角：** 所有可交互元素和内容容器都应使用 rounded-lg 或 rounded-xl 等圆角。
* **间距：** 遵循Tailwind CSS的间距规范 (p-x, m-x)，保持一致的视觉节奏。
* **按钮：** 统一的按钮样式（背景色、文字颜色、hover效果、圆角、阴影）。
* **输入框：** 统一的输入框样式（边框、背景色、focus效果、圆角）。
* **响应式：** 广泛使用Tailwind的响应式前缀（sm:, md:, lg:）确保在不同设备上的良好显示。
* **动画/过渡：** 适当使用CSS过渡效果增强用户体验（例如，按钮点击反馈、页面切换）。

### **4.2 后端/数据库设计 (Firestore)**

#### **4.2.1 Firebase初始化与认证**

* 在 index.js 或 App.js 中初始化Firebase应用。
* 使用 getAuth() 获取认证实例。
* 在应用启动时，使用 \_\_initial\_auth\_token 进行自定义令牌登录 (signInWithCustomToken)，如果令牌不存在则匿名登录 (signInAnonymously)。
* 使用 onAuthStateChanged 监听认证状态变化，并更新前端的 userId 和 isAuthenticated 状态。

// utils/authService.js  
import { getAuth, signInAnonymously, signInWithCustomToken, onAuthStateChanged } from 'firebase/auth';  
import { initializeApp } from 'firebase/app';  
  
let app;  
let auth;  
let db; // Firestore instance  
  
export const initFirebase = (firebaseConfig, initialAuthToken) => {  
 if (!app) {  
 app = initializeApp(firebaseConfig);  
 auth = getAuth(app);  
 db = getFirestore(app); // Initialize Firestore here  
   
 // Handle initial authentication  
 if (typeof initialAuthToken !== 'undefined' && initialAuthToken) {  
 signInWithCustomToken(auth, initialAuthToken).catch(error => {  
 console.error("Error signing in with custom token:", error);  
 signInAnonymously(auth).catch(anonError => console.error("Error signing in anonymously:", anonError));  
 });  
 } else {  
 signInAnonymously(auth).catch(error => console.error("Error signing in anonymously:", error));  
 }  
 }  
 return { auth, db };  
};  
  
export const onAuthChange = (callback) => {  
 return onAuthStateChanged(auth, callback);  
};  
  
export const getUserId = () => auth.currentUser?.uid || crypto.randomUUID(); // Fallback for unauthenticated

#### **4.2.2 Firestore数据模型**

* **users 集合 (私有数据):** artifacts/{appId}/users/{userId}/profile
  + **文档ID:** profile (每个用户一个文档)
  + **字段:**
    - username: String (用户注册的唯一用户名)
    - createdAt: Timestamp
    - lastLoginAt: Timestamp
  + **用途：** 存储用户注册的用户名，方便在排行榜中显示。
* **numberClickScores 集合 (公共数据):** artifacts/{appId}/public/data/numberClickScores
  + **文档ID:** 自动生成
  + **字段:**
    - userId: String (记录成绩的用户ID)
    - username: String (记录成绩时的用户名，冗余存储方便查询)
    - score: Number (用时，单位毫秒)
    - timestamp: Timestamp (成绩提交时间)
  + **用途：** 存储“按顺序点击数字”游戏的成绩，用于排行榜。
* **appConfigs 集合 (公共数据):** artifacts/{appId}/public/data/appConfigs
  + **文档ID:** general (一个文档包含所有通用配置)
  + **字段:**
    - privacyPolicyUrl: String (隐私条款链接)
    - introductionText: String (简介页面文本内容)
    - loginGuidanceText: String (登录注册页面引导文本)
    - sidebarItems: Array of Objects (侧边栏菜单配置，例如 [{ id: 'intro', label: '简介', path: '/intro' }, { id: 'numberGame', label: '数字点击', path: '/game/number-click' }])
  + **用途：** 存储可配置的文本内容和链接，方便后期维护。

#### **4.2.3 Firestore操作封装**

封装Firestore的CRUD操作，确保数据路径和用户权限管理正确。

// utils/firestoreService.js  
import { getFirestore, doc, getDoc, setDoc, addDoc, collection, query, orderBy, onSnapshot } from 'firebase/firestore';  
  
let db;  
let appId;  
  
export const initFirestore = (firestoreInstance, currentAppId) => {  
 db = firestoreInstance;  
 appId = currentAppId;  
};  
  
// 获取用户私有配置，例如用户名  
export const getUserProfile = async (userId) => {  
 if (!db || !appId || !userId) return null;  
 const docRef = doc(db, `artifacts/${appId}/users/${userId}/profile`, 'profile');  
 const docSnap = await getDoc(docRef);  
 return docSnap.exists() ? docSnap.data() : null;  
};  
  
// 设置用户私有配置，例如用户名  
export const setUserProfile = async (userId, data) => {  
 if (!db || !appId || !userId) return;  
 const docRef = doc(db, `artifacts/${appId}/users/${userId}/profile`, 'profile');  
 await setDoc(docRef, data, { merge: true });  
};  
  
// 添加数字点击游戏成绩  
export const addNumberClickScore = async (scoreData) => {  
 if (!db || !appId) return;  
 const collectionRef = collection(db, `artifacts/${appId}/public/data/numberClickScores`);  
 await addDoc(collectionRef, scoreData);  
};  
  
// 实时监听数字点击游戏排行榜  
export const listenToNumberClickScores = (callback) => {  
 if (!db || !appId) return () => {};  
 // 注意：Firestore的orderBy可能需要索引，为避免复杂性，这里先不使用orderBy，而是在客户端排序  
 const q = collection(db, `artifacts/${appId}/public/data/numberClickScores`);  
 const unsubscribe = onSnapshot(q, (snapshot) => {  
 const scores = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));  
 // 客户端排序：按分数升序  
 scores.sort((a, b) => a.score - b.score);  
 callback(scores);  
 }, (error) => {  
 console.error("Error listening to number click scores:", error);  
 });  
 return unsubscribe;  
};  
  
// 获取应用配置  
export const getAppConfigs = async () => {  
 if (!db || !appId) return null;  
 const docRef = doc(db, `artifacts/${appId}/public/data/appConfigs`, 'general');  
 const docSnap = await getDoc(docRef);  
 return docSnap.exists() ? docSnap.data() : null;  
};  
  
// 设置应用配置 (通常由管理员操作，这里仅为示例)  
export const setAppConfigs = async (data) => {  
 if (!db || !appId) return;  
 const docRef = doc(db, `artifacts/${appId}/public/data/appConfigs`, 'general');  
 await setDoc(docRef, data, { merge: true });  
};

#### **4.2.4 Firestore安全规则 (示例)**

Firebase安全规则将确保数据的访问权限。以下是基本示例，实际部署时需根据需求细化。

rules\_version = '2';  
service cloud.firestore {  
 match /databases/{database}/documents {  
  
 // 公共数据：只有认证用户可读写  
 match /artifacts/{appId}/public/data/{collection}/{document} {  
 allow read, write: if request.auth != null;  
 }  
  
 // 用户私有数据：只有认证用户且是数据所有者可读写  
 match /artifacts/{appId}/users/{userId}/{collection}/{document} {  
 allow read, write: if request.auth != null && request.auth.uid == userId;  
 }  
 }  
}

### **4.3 部署与运维设计**

* **前端构建：** React应用将通过npm run build命令构建为静态文件（HTML, CSS, JS）。
* **腾讯云静态网站托管：** 构建后的静态文件将上传至腾讯云的COS（对象存储）桶，并配置为静态网站托管。
* **GitHub Actions CI/CD：**
  + 配置一个GitHub Actions工作流 (.github/workflows/deploy.yml)。
  + 触发条件：例如，每次向 main 分支推送代码时。
  + 工作流步骤：
    1. 检出代码。
    2. 安装Node.js依赖。
    3. 运行 npm install。
    4. 运行 npm run build。
    5. 使用腾讯云CLI或GitHub Actions市场提供的COS上传Action将构建产物同步到COS桶。
* **Cursor集成：** 开发者在Cursor中修改代码后，通过Cursor的Git集成功能将代码推送到GitHub仓库，从而触发上述CI/CD流程。

### **4.4 可扩展性与维护性**

* **模块化设计：** 前端组件化，后端服务封装，确保各部分职责单一，易于独立开发和测试。
* **配置化管理：** 文本内容、链接、侧边栏菜单等通过Firestore进行配置，无需修改代码即可更新。
* **命名规范：**
  + **React组件：** PascalCase (例如 NumberClickGame, InputField)
  + **JavaScript变量/函数：** camelCase (例如 userId, initFirebase)
  + **Firestore集合/文档ID/字段：** camelCase 或 snake\_case (例如 numberClickScores, privacyPolicyUrl)
  + **CSS类/Tailwind类：** kebab-case (例如 bg-blue-500, text-center)
* **注释：** 关键逻辑、复杂算法、函数接口等都应有详细注释。
* **错误处理：** 前端提供友好的错误提示，后端（Firebase）的错误日志可通过控制台查看。

## **5. 未来扩展考虑**

* **更多游戏：** 增加记忆力、反应力、逻辑思维等不同类型的小游戏。
* **用户档案：** 更详细的用户信息、游戏进度追踪、成就系统。
* **家长端功能：** 家长可查看孩子的游戏数据、设置游戏时间等。
* **内容管理系统 (CMS)：** 开发一个简单的后台管理界面，方便非技术人员管理配置内容、图片等。
* **数据可视化：** 以图表形式展示孩子的注意力提升趋势。