```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="color-scheme" content="light dark">
    <!--
      [AI VERSION RULE]
      Do NOT update the version number in <title> or any other place.
      The user designates the version number at the START of an update cycle.
    -->
    <title>时间银行 - Time Bank v7.12.0</title>
    <link rel="manifest" href="manifest.json">
    <meta name="theme-color" content="#667eea">
    <meta name="apple-mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-status-bar-style" content="black-translucent">
    <meta name="apple-mobile-web-app-title" content="时间银行">
    <link rel="apple-touch-icon" href="icon-192.png">
    <!-- [v7.3.4] 立即应用主题，避免首次加载白屏闪烁 -->
    <script>
    (function() {
        try {
            var savedTheme = localStorage.getItem('themePreference') || 'system';
            var theme = savedTheme;
            if (savedTheme === 'system') {
                 theme = window.matchMedia('(prefers-color-scheme: dark)').matches ?
'dark' : 'light';
            }
            document.documentElement.setAttribute('data-theme', theme);
            // 立即应用主题色
            var accentTheme = localStorage.getItem('accentTheme');
            if (accentTheme) {
                document.documentElement.setAttribute('data-accent', accentTheme);
            }
        } catch(e) {
            console.warn('[EarlyTheme] Failed:', e);
        }
    })();
    </script>
    <!-- [v6.6.0] 腾讯云 CloudBase SDK v1.7.1 - 使用本地文件 -->
    <script>
        // SDK 加载状态标记
        window.cloudbaseSDKLoaded = false;
        window.cloudbaseSDKError = null;
    </script>
    <script src="cloudbase.v2.bundle.js"
            onload="window.cloudbaseSDKLoaded = true; console.log('[CloudBase] SDK v2
loaded successfully');"
                  onerror="window.cloudbaseSDKError = 'Local SDK v2 load failed';
```

```
console.error('[CloudBase] Local SDK v2 load failed!');"></script>
    <style>
        /* --- [v3.7.7] Fixed Layout Implementation --- */
        :root {
            /* [v4.8.2] 修复安卓原生下拉框暗色模式背景问题 */
            color-scheme: light;

            /* 1.3.2. Spacing Tokens */
            --space-xs: 4px;
            --space-sm: 8px;
            --space-md: 12px;
            --space-lg: 16px;
            --space-xl: 24px;
            --space-xxl: 32px;

            /* Other variables */
            --bg-gradient: var(--bg-gradient-themed, linear-gradient(135deg, #667eea
0%, #764ba2 100%));
            --card-bg: rgba(255, 255, 255, 0.95);
            --card-shadow: rgba(0, 0, 0, 0.1);
            --text-color: #333;
            --text-color-light: #666;
            --border-color: #ddd;
            --input-bg: #fff;
            --header-text: white;
            --section-title-color: white;
            --btn-secondary-bg: #f5f5f5;
            --btn-secondary-text: #666;
            --btn-secondary-border: #ddd;
            --habit-border-width: 2px; /* [v3.12.6] Reduced from 4px */
            --dropdown-bg: #ffffff;
            --dropdown-shadow: rgba(0, 0, 0, 0.15);
            --color-primary: #2196F3;
            --color-primary-rgb: 33, 150, 243; /* [v6.1.0] 用于透明度调节 */
            --color-positive: #4CAF50;
            --color-negative: #f44336;
            --color-warning: #FF9800;
            --color-neutral: #607D8B;
            --color-other: #BDBDBD; /* [v3.10.4] Lighter Gray */
            --android-nav-bottom: 0px; /* Android 三键导航栏适配 */
            /* [v6.4.x] 全局通透强度（1=默认） */
            --glass-strength: 1;
            --glass-opacity-scale: 1;
            --glass-blur-scale: 1;

            /* [v6.0.0] 主题色渐变方案 - 使用中间色避免渐变脏污 */
            --accent-gradient: linear-gradient(135deg, #2196F3 0%, #5c6bc0 50%, #7c4dff
100%);
            --accent-start: #2196F3;
            --accent-mid: #5c6bc0;
            --accent-end: #7c4dff;
            /* [v6.0.0] 主题背景色 */
            --bg-gradient-themed: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

            /* [v4.5.0] Task Calendar Colors (Green) */
            --color-task-green-1: #9be9a8;
            --color-task-green-2: #40c463;
            --color-task-green-3: #216e39;
            /* [v4.5.0] Task Calendar Colors (Red) */
            --color-task-red-1: #ffcdd2;
            --color-task-red-2: #e57373;
            --color-task-red-3: #f44336;
        }

        [data-theme="dark"] {
            /* [v4.8.2] 修复安卓原生下拉框暗色模式背景问题 */
            color-scheme: dark;

            /* [v6.0.0] 暗色模式主题背景 */
              --bg-gradient-themed: linear-gradient(135deg, #1a1a2e 0%, #16213e 50%,
#0f3460 100%);
            --bg-gradient: var(--bg-gradient-themed);
            --card-bg: rgba(42, 42, 42, 0.95);
            --card-shadow: rgba(0, 0, 0, 0.3);
            --text-color: #e0e0e0;
            --text-color-light: #aaa;
            --border-color: #444;
            --input-bg: #2a2a2a;
            --header-text: #e0e0e0;
            --section-title-color: #e0e0e0;
            --btn-secondary-bg: #3a3a3a;
            --btn-secondary-text: #e0e0e0;
            --btn-secondary-border: #555;
            --dropdown-bg: #3a3a3a;
            --dropdown-shadow: rgba(0, 0, 0, 0.5);
            --color-other: #616161; /* [v3.10.4] Lighter Gray for Dark Mode */

            /* [v4.5.2] FIX: Removed 6 incorrect dark mode color variables for calendars
*/
        }

        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            transition: background-color 0.3s ease, color 0.3s ease, border-color 0.3s
ease;
            -webkit-tap-highlight-color: transparent; /* [v5.9.0] 全局禁用点击蓝色高亮
```

```
*/
        }

        /* [v5.0.0] 彻底修复移动端过度滚动拉伸问题 */
        html, body {
            margin: 0;
            padding: 0;
            height: 100%;
            width: 100%;
            overflow: hidden;
            position: fixed;
            top: 0;
            left: 0;
            right: 0;
            bottom: 0;
            overscroll-behavior: none;
            -webkit-overflow-scrolling: touch;
        }

        body {
                font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto,
sans-serif;
            background: var(--bg-gradient);
            color: var(--text-color);
            display: flex;
            flex-direction: column;
        }

        /* 主滚动容器 */
        .app-scroll-container {
            flex: 1;
            overflow-y: auto;
            overflow-x: hidden;
            overscroll-behavior: contain;
            -webkit-overflow-scrolling: touch;
                     padding-bottom:  calc(80px  +  var(--android-nav-bottom,  0px)  +
env(safe-area-inset-bottom, 0px));
            /* 防止触摸穿透和弹性滚动 */
            touch-action: pan-y;
        }

        .header {
            text-align: center;
                   padding:  calc(var(--space-xxl)  +  env(safe-area-inset-top,  0px))
var(--space-xl) var(--space-sm);
            color: var(--header-text);
        }

        .header h1 {
            font-size: 2rem;
            margin-bottom: 0;
        }

        .header p {
            opacity: 0.9;
            font-size: 0.9rem;
        }

        .main-container {
            padding: 0 var(--space-lg);
        }
        @media (min-width: 480px) {
            .main-container { padding: 0 var(--space-xl); }
        }

        .balance-card {
            margin-top: var(--space-sm);
            margin-bottom: var(--space-lg);
            background: var(--card-bg);
            border-radius: 15px;
            padding: var(--space-lg);
            text-align: center;
            box-shadow: 0 8px 32px var(--card-shadow);
            backdrop-filter: blur(10px);
            border: 2px solid transparent;
            cursor: pointer; /* [v4.1.0] Added */
        }

        .balance-card.negative { background: rgba(255, 235, 238, 0.95); border-color:
var(--color-negative); }
         [data-theme="dark"] .balance-card.negative { background: rgba(40, 20, 20,
0.95); }

        /* [v6.0.0] 时间余额卡片通透模式 - 使用 CSS 变量响应通透强度调节 */
        .balance-card.glass {
                  background: linear-gradient(135deg, rgba(255,255,255, calc(0.08 *
var(--glass-opacity-scale)))      0%,      rgba(255,255,255,      calc(0.02      *
var(--glass-opacity-scale))) 100%);
                            border:  1px  solid  rgba(255,  255,  255,  calc(0.3  *
var(--glass-opacity-scale)));
                        border-top:  1px  solid  rgba(255,  255,  255,  calc(0.45  *
var(--glass-opacity-scale)));
            box-shadow: 0 8px 32px rgba(0, 0, 0, 0.15);
            backdrop-filter: blur(calc(20px * var(--glass-blur-scale)));
            -webkit-backdrop-filter: blur(calc(20px * var(--glass-blur-scale)));
        }
        .balance-card.glass .balance-title {
            color: rgba(255, 255, 255, 0.85);
            text-shadow: 0 1px 2px rgba(0, 0, 0, 0.3);
        }
        .balance-card.glass .balance-amount {
            color: white;
            text-shadow: 0 2px 4px rgba(0, 0, 0, 0.4);
        }
        .balance-card.glass .daily-change {
            color: rgba(255, 255, 255, 0.8);
            text-shadow: 0 1px 2px rgba(0, 0, 0, 0.3);
        }
        .balance-card.glass .daily-change .positive { color: #90EE90; }
        .balance-card.glass .daily-change .negative { color: #FFB6C1; }

        /* 暗色模式下的通透样式 */
        [data-theme="dark"] .balance-card.glass {
            background: linear-gradient(135deg, rgba(255,255,255, calc(0.06 * var(--glass-opacity-scale))) 0%, rgba(255,255,255, calc(0.01 * var(--glass-opacity-scale))) 100%);
            border: 1px solid rgba(255, 255, 255, calc(0.12 * var(--glass-opacity-scale)));
            border-top: 1px solid rgba(255, 255, 255, calc(0.2 * var(--glass-opacity-scale)));
            box-shadow: 0 8px 32px rgba(0, 0, 0, 0.3);
        }

        /* 余额为负时的通透模式 */
        .balance-card.glass.negative {
            background: linear-gradient(135deg, rgba(255,100,100, calc(0.12 * var(--glass-opacity-scale))) 0%, rgba(255,80,80, calc(0.05 * var(--glass-opacity-scale))) 100%);
            border-color: rgba(255, 150, 150, calc(0.35 * var(--glass-opacity-scale)));
        }
        [data-theme="dark"] .balance-card.glass.negative {
            background: linear-gradient(135deg, rgba(255,80,80, calc(0.10 * var(--glass-opacity-scale))) 0%, rgba(255,60,60, calc(0.04 * var(--glass-opacity-scale))) 100%);
            border-color: rgba(255, 120, 120, calc(0.25 * var(--glass-opacity-scale)));
        }

        .balance-title { font-size: 0.9rem; color: var(--text-color-light); margin-bottom: var(--space-sm); }
        .balance-amount { font-size: 2.2rem; font-weight: bold; margin-bottom: var(--space-lg); color: var(--color-primary); }
        .daily-changes { display: flex; justify-content: space-between; font-size: 0.8rem; }
        .daily-change { color: var(--text-color-light); }

        /* [v5.10.0] 卡片堆叠系统 */
        .card-stack {
            margin-top: var(--space-lg);
            margin-bottom: var(--space-lg);
            position: relative;
            touch-action: pan-x pinch-zoom;
        }
        .card-stack .balance-card {
            position: relative;
            z-index: 2; /* 确保在堆叠容器上方 */
            margin-top: 0;
            margin-bottom: 0;
            /* 圆角保持默认，不做修改 */
        }
        /* 堆叠卡片容器 */
        .stacked-cards-container {
            position: relative;
            z-index: 1; /* 在余额卡片下方，负 margin 时被遮挡 */
            overflow: hidden;
            margin-top: -12px; /* 默认堆叠状态 */
            transition: margin-top 0.35s cubic-bezier(0.4, 0, 0.2, 1);
            will-change: margin-top; /* 优化动画性能 */
        }
        /* [v7.4.0] 容器的 margin 只由屏幕时间卡片状态决定 */
        .stacked-cards-container.st-expanded {
            margin-top: 12px; /* 屏幕时间展开时，统一与睡眠卡片间距 */
        }

        /* [v5.10.0] 统一屏幕时间卡片 wrapper */
        .screen-time-wrapper {
            background: linear-gradient(135deg, #27ae60 0%, #1abc9c 100%); /* 默认绿色，
由 JS 动态更新 */
            border-radius: 0 0 15px 15px;
            color: white;
            cursor: pointer;
            position: relative;
            z-index: 1; /* 高于睡眠卡片 */
            /* 统一所有动画参数：0.35s + cubic-bezier(0.4, 0, 0.2, 1) */
            transition: background 0.35s cubic-bezier(0.4, 0, 0.2, 1),
                        border-radius 0.35s cubic-bezier(0.4, 0, 0.2, 1),
                        box-shadow 0.35s cubic-bezier(0.4, 0, 0.2, 1);
            box-shadow: 0 4px 12px rgba(74, 144, 226, 0.25);
            overflow: hidden;
        }
        .screen-time-wrapper.expanded {
            border-radius: 15px; /* 展开时完整圆角 */
            box-shadow: 0 4px 16px rgba(74, 144, 226, 0.3);
        }
        [data-theme="dark"] .screen-time-wrapper.classic {
            filter: brightness(0.75);
        }
        /* [v5.10.0] 通透模式样式 - iOS 透明玻璃质感 */
        .screen-time-wrapper.glass {
            background: linear-gradient(135deg, rgba(255,255,255, calc(0.08 *
var(--glass-opacity-scale))) 0%, rgba(255,255,255, calc(0.02 *
var(--glass-opacity-scale))) 100%);
            border: 1px solid rgba(255, 255, 255, calc(0.3 *
var(--glass-opacity-scale)));
            border-top: 1px solid rgba(255, 255, 255, calc(0.5 *
var(--glass-opacity-scale)));
            color: white;
            text-shadow: 0 1px 3px rgba(0, 0, 0, 0.5);
            box-shadow: 0 4px 16px rgba(0, 0, 0, 0.12);
            backdrop-filter: blur(calc(20px * var(--glass-blur-scale)));
            -webkit-backdrop-filter: blur(calc(20px * var(--glass-blur-scale)));
        }
        [data-theme="dark"] .screen-time-wrapper.glass {
            background: linear-gradient(135deg, rgba(255,255,255, calc(0.06 *
var(--glass-opacity-scale))) 0%, rgba(255,255,255, calc(0.01 *
var(--glass-opacity-scale))) 100%);
            border: 1px solid rgba(255, 255, 255, calc(0.12 *
var(--glass-opacity-scale)));
            border-top: 1px solid rgba(255, 255, 255, calc(0.2 *
var(--glass-opacity-scale)));
            box-shadow: 0 4px 16px rgba(0, 0, 0, 0.25);
        }
        .screen-time-wrapper.glass.expanded {
            box-shadow: 0 8px 24px rgba(0, 0, 0, 0.18);
        }
        [data-theme="dark"] .screen-time-wrapper.glass.expanded {
            box-shadow: 0 8px 24px rgba(0, 0, 0, 0.35);
        }
        /* 通透模式进度条背景 */
        .screen-time-wrapper.glass .st-progress {
            background: rgba(255, 255, 255, calc(0.2 *
var(--glass-opacity-scale))) !important;
        }
        [data-theme="dark"] .screen-time-wrapper.glass .st-progress {
            background: rgba(255, 255, 255, calc(0.12 *
var(--glass-opacity-scale))) !important;
        }
        /* 屏幕时间进度条通透模式 - 背景毛玻璃，保留原有进度条颜色 */
        body.glass-mode .st-progress {
            background: rgba(255, 255, 255, calc(0.15 *
var(--glass-opacity-scale))) !important;
            backdrop-filter: blur(calc(8px * var(--glass-blur-scale)));
            -webkit-backdrop-filter: blur(calc(8px * var(--glass-blur-scale)));
            border: 1px solid rgba(255, 255, 255, calc(0.2 *
var(--glass-opacity-scale)));
```

```
        }
        /* 进度条保留原有颜色，仅添加光晕效果 */
        body.glass-mode .st-progress-bar {
            box-shadow: 0 0 8px currentColor;
        }
        /* 通透模式文字和分隔线 */
        .screen-time-wrapper.glass .st-footer {
            border-top-color: rgba(255, 255, 255, 0.15);
        }
        [data-theme="dark"] .screen-time-wrapper.glass .st-footer {
            border-top-color: rgba(255, 255, 255, 0.1);
        }

        /* 共享 header - 始终显示 */
        .screen-time-shared-header {
            display: flex;
            align-items: center;
            padding: 12px 16px;
            transition: padding 0.35s cubic-bezier(0.4, 0, 0.2, 1);
        }
        /* 容器堆叠时，屏幕时间 header 需要顶部 padding 补偿 */
        .stacked-cards-container:not(.st-expanded) .screen-time-shared-header {
            padding: 22px 16px 12px 16px;
        }
        .screen-time-shared-header .st-icon { font-size: 1.1rem; margin-right: 8px; }
        .screen-time-shared-header .st-title { font-weight: 600; font-size: 0.95rem;
flex: 1; }
         .screen-time-shared-header .st-percent { font-size: 1rem; font-weight: 700;
margin-right: 8px; }
        .screen-time-shared-header .st-arrow {
            font-size: 0.8rem;
            opacity: 0.8;
            /* 统一动画参数 */
            transition: transform 0.35s cubic-bezier(0.4, 0, 0.2, 1),
                        opacity 0.35s cubic-bezier(0.4, 0, 0.2, 1);
        }
        .screen-time-wrapper.expanded .screen-time-shared-header .st-arrow {
            transform: rotate(180deg);
            opacity: 0; /* 展开时箭头淡出 */
        }

        /* 可展开的 body - 使用 max-height 动画 */
        .screen-time-expandable-body {
            max-height: 0;
            opacity: 0;
            overflow: hidden;
            padding: 0 16px;
            /* 统一所有动画参数，包括 padding */
            transition: max-height 0.35s cubic-bezier(0.4, 0, 0.2, 1),
                        opacity 0.35s cubic-bezier(0.4, 0, 0.2, 1),
                        padding 0.35s cubic-bezier(0.4, 0, 0.2, 1);
        }
        .screen-time-wrapper.expanded .screen-time-expandable-body {
            max-height: 150px; /* 足够容纳内容 */
            opacity: 1;
            padding: 0 16px 16px 16px;
        }

        /* body 内部元素样式 */
        .screen-time-expandable-body .st-progress {
            height: 8px;
            background: rgba(255, 255, 255, 0.3);
            border-radius: 4px;
            overflow: hidden;
            margin-bottom: 8px;
        }
        .screen-time-expandable-body .st-progress-bar {
            height: 100%;
            background: white;
            border-radius: 4px;
            transition: width 0.3s ease;
            width: 0%;
        }
        .screen-time-expandable-body .st-stats {
            display: flex;
            justify-content: space-between;
            font-size: 0.9rem;
            opacity: 0.95;
            margin-bottom: 8px;
        }
        .screen-time-expandable-body .st-footer {
            font-size: 0.85rem;
            opacity: 0.9;
            text-align: center;
            padding-top: 4px;
            border-top: 1px solid rgba(255,255,255,0.2);
        }

        /* [v7.4.0] 睡眠时间管理卡片 */
        .sleep-card-wrapper {
            background: linear-gradient(135deg, #6366f1 0%, #8b5cf6 100%);
            border-radius: 0 0 15px 15px;
            color: white;
            cursor: pointer;
            margin-top: -12px; /* 默认堆叠状态 */
            position: relative;
            z-index: 0; /* 低于屏幕时间卡片 */
            overflow: hidden;
```

```
            transition: all 0.35s cubic-bezier(0.4, 0, 0.2, 1);
            box-shadow: 0 4px 12px rgba(99, 102, 241, 0.25);
        }
        .screen-time-wrapper {
            position: relative;
            z-index: 1; /* 高于睡眠卡片 */
        }
        /* 睡眠卡片展开时才有正常间距 */
        .sleep-card-wrapper.expanded {
            border-radius: 15px;
            margin-top: 12px;
            box-shadow: 0 4px 16px rgba(99, 102, 241, 0.3);
        }
        .sleep-card-wrapper:active {
            transform: scale(0.98);
        }
        [data-theme="dark"] .sleep-card-wrapper {
            filter: brightness(0.85);
        }
        /* 通透模式睡眠卡片 */
        body.glass-mode .sleep-card-wrapper {
                  background: linear-gradient(135deg, rgba(255,255,255, calc(0.08 *
var(--glass-opacity-scale))) 0%, rgba(255,255,255, calc(0.02 *
var(--glass-opacity-scale))) 100%);
                            border: 1px solid rgba(255, 255, 255, calc(0.3 *
var(--glass-opacity-scale)));
            text-shadow: 0 1px 3px rgba(0, 0, 0, 0.5);
            box-shadow: 0 4px 16px rgba(0, 0, 0, 0.12);
            backdrop-filter: blur(calc(20px * var(--glass-blur-scale)));
            -webkit-backdrop-filter: blur(calc(20px * var(--glass-blur-scale)));
        }
        [data-theme="dark"] body.glass-mode .sleep-card-wrapper {
                  background: linear-gradient(135deg, rgba(255,255,255, calc(0.06 *
var(--glass-opacity-scale))) 0%, rgba(255,255,255, calc(0.01 *
var(--glass-opacity-scale))) 100%);
                            border: 1px solid rgba(255, 255, 255, calc(0.12 *
var(--glass-opacity-scale)));
            filter: none;
        }
        .sleep-card-header {
            display: flex;
            align-items: center;
            padding: 12px 16px;
            transition: padding 0.35s cubic-bezier(0.4, 0, 0.2, 1);
        }
        /* 睡眠卡片收起时需要 padding 补偿（因为有负 margin 堆叠） */
        .sleep-card-wrapper:not(.expanded) .sleep-card-header {
            padding: 24px 16px 12px 16px; /* 顶部补偿负 margin */
        }
```

```
.sleep-card-header .sleep-icon {
    font-size: 1.1rem;
    margin-right: 8px;
}
.sleep-card-header .sleep-title {
    font-weight: 600;
    font-size: 0.95rem;
}
.sleep-card-header .sleep-status {
    font-size: 0.85rem;
    opacity: 0.9;
    padding: 2px 8px;
    background: rgba(255,255,255,0.2);
    border-radius: 10px;
    margin-left: auto; /* 状态右对齐 */
}
.sleep-mode-switch {
    font-size: 0.75rem;
    padding: 2px 8px;
    margin-left: 6px;
    background: rgba(255,255,255,0.25);
    border: none;
    border-radius: 10px;
    color: inherit;
    cursor: pointer;
    opacity: 0.9;
}
.sleep-mode-switch:active {
    opacity: 0.7;
}
/* [v7.9.7] 手动添加按钮 */
.sleep-add-btn {
    font-size: 1rem;
    font-weight: 600;
    width: 24px;
    height: 24px;
    margin-left: 6px;
    background: rgba(255,255,255,0.25);
    border: none;
    border-radius: 50%;
    color: inherit;
    cursor: pointer;
    opacity: 0.9;
    display: flex;
    align-items: center;
    justify-content: center;
    line-height: 1;
}
.sleep-add-btn:active {
```

```css
    opacity: 0.7;
    transform: scale(0.95);
}
/* 收起状态隐藏手动添加按钮 */
.sleep-card-wrapper:not(.expanded) .sleep-add-btn {
    display: none;
}
.sleep-mode-label {
    font-size: 0.95rem;
    font-weight: 600;
    opacity: 0.9;
}
/* 收起状态隐藏模式标签和切换按钮 */
.sleep-card-wrapper:not(.expanded) .sleep-mode-label,
.sleep-card-wrapper:not(.expanded) .sleep-mode-switch {
    display: none;
}
.sleep-card-header .sleep-arrow {
    font-size: 0.8rem;
    opacity: 0.8;
    margin-left: 8px;
    transition: transform 0.35s cubic-bezier(0.4, 0, 0.2, 1),
                opacity 0.35s cubic-bezier(0.4, 0, 0.2, 1);
}
.sleep-card-wrapper.expanded .sleep-card-header .sleep-arrow {
    transform: rotate(180deg);
    opacity: 0;
}
.sleep-card-body {
    max-height: 0;
    opacity: 0;
    overflow: hidden;
    padding: 0 16px;
    transition: max-height 0.35s cubic-bezier(0.4, 0, 0.2, 1),
                opacity 0.35s cubic-bezier(0.4, 0, 0.2, 1),
                padding 0.35s cubic-bezier(0.4, 0, 0.2, 1);
}
.sleep-card-wrapper.expanded .sleep-card-body {
    max-height: 230px;
    opacity: 1;
    padding: 0 16px 16px 16px;
}
.sleep-time-row {
    display: flex;
    justify-content: space-around;
    margin-bottom: 12px;
}
.sleep-time-item {
    text-align: center;
```

```css
}
.sleep-time-label {
    display: block;
    font-size: 0.75rem;
    opacity: 0.8;
    margin-bottom: 2px;
}
.sleep-time-value {
    font-size: 1.2rem;
    font-weight: 700;
}
/* [v7.4.0] 睡眠时长显示行 */
.sleep-duration-row {
    text-align: center;
    padding: 8px 0;
    border-top: 1px solid rgba(255,255,255,0.15);
    margin-top: 8px;
}
.sleep-duration-label {
    font-size: 0.85rem;
    opacity: 0.8;
    margin-right: 8px;
}
.sleep-duration-value {
    font-size: 1.3rem;
    font-weight: 700;
}
/* [v7.4.2] 睡眠简报 */
.sleep-summary {
    margin-top: 10px;
    padding: 8px 10px 6px 10px;
    border-top: 1px solid rgba(255,255,255,0.15);
    font-size: 0.85rem;
    line-height: 1.4;
}
.sleep-summary-title {
    font-size: 0.75rem;
    opacity: 0.8;
    margin-bottom: 4px;
}
.sleep-summary-line {
    opacity: 0.95;
    margin-bottom: 2px;
}
.sleep-action-row {
    text-align: center;
    display: flex;
    gap: 8px;
    justify-content: center;
```

```css
}
.sleep-action-btn {
    background: rgba(255,255,255,0.25);
    border: 1px solid rgba(255,255,255,0.4);
    color: white;
    padding: 10px 24px;
    border-radius: 20px;
    font-size: 0.95rem;
    font-weight: 600;
    cursor: pointer;
    transition: all 0.2s ease;
    flex: 1;
}
.sleep-action-btn.secondary {
    background: rgba(255,255,255,0.1);
    flex: 0 0 auto;
    padding: 10px 16px;
}
.sleep-action-btn:hover {
    background: rgba(255,255,255,0.35);
}
.sleep-action-btn:active {
    transform: scale(0.95);
}
.sleep-action-btn:disabled {
    opacity: 0.5;
    cursor: not-allowed;
}
/* 睡眠中状态 */
.sleep-card-wrapper.sleeping {
    background: linear-gradient(135deg, #1e1b4b 0%, #312e81 100%);
}
.sleep-card-wrapper.sleeping .sleep-status {
    background: rgba(99, 102, 241, 0.5);
}
/* 今日已取消状态 */
.sleep-card-wrapper.cancelled {
    background: linear-gradient(135deg, #6b7280 0%, #9ca3af 100%);
    opacity: 0.7;
}

/* [v5.2.0] 屏幕时间管理卡片 */
.screen-time-card {
    background: linear-gradient(135deg, var(--color-primary) 0%, #5a8dee 100%);
    border-radius: 15px;
    padding: 16px;
    margin-bottom: var(--space-lg);
    color: white;
    cursor: pointer;
```

```
            transition: transform 0.2s ease, box-shadow 0.2s ease, border-radius 0.3s
ease;
            box-shadow: 0 4px 16px rgba(74, 144, 226, 0.3);
        }
        /* [v5.10.0] 堆叠时的样式调整 */
        .card-stack .screen-time-card {
            margin-bottom: 0;
            border-radius: 0 0 15px 15px;
        }
        .screen-time-card:active { transform: scale(0.98); }
        [data-theme="dark"] .screen-time-card {
            filter: brightness(0.70);
            box-shadow: 0 4px 16px rgba(0, 0, 0, 0.4);
        }
        .screen-time-header {
            display: flex;
            align-items: center;
            margin-bottom: 12px;
        }
        .screen-time-icon { font-size: 1.2rem; margin-right: 8px; }
        .screen-time-title { font-weight: 600; flex: 1; font-size: 1rem; }
        .screen-time-percent { font-size: 1.1rem; font-weight: 700; opacity: 0.95; }
        .screen-time-body { margin-bottom: 8px; }
        .screen-time-progress {
            height: 8px;
            background: rgba(255, 255, 255, 0.3);
            border-radius: 4px;
            overflow: hidden;
            margin-bottom: 8px;
        }
        .screen-time-progress-bar {
            height: 100%;
            background: white;
            border-radius: 4px;
            transition: width 0.3s ease;
            width: 0%;
        }
        .screen-time-stats {
            display: flex;
            justify-content: space-between;
            font-size: 0.9rem;
            opacity: 0.95;
        }
        .screen-time-footer {
            padding-top: 10px;
            border-top: 1px solid rgba(255, 255, 255, 0.2);
            font-size: 0.85rem;
            text-align: center;
            opacity: 0.9;
        }
        .screen-time-card.over-limit {
            box-shadow: 0 4px 16px rgba(231, 76, 60, 0.3);
        }
        .screen-time-card.over-limit .screen-time-footer { color: rgba(255, 200, 200,
0.9); }

        /* 白名单弹窗样式 */
           #whitelistModal .modal-content { display: flex; flex-direction: column;
max-height: 70vh; }
           #whitelistModal .modal-body { flex: 1; overflow: hidden; display: flex;
flex-direction: column; }
         #whitelistModal .modal-footer { display: flex; gap: 12px; justify-content:
flex-end; }
        .whitelist-modal-content { flex: 1; max-height: 40vh; overflow-y: auto; }
        .whitelist-search {
            width: 100%;
            padding: 10px 12px;
            border: 1px solid var(--border-color);
            border-radius: 8px;
            margin-bottom: 12px;
            font-size: 0.95rem;
            background: var(--bg-color);
            color: var(--text-color);
        }
        .whitelist-item {
            display: flex;
            align-items: center;
            padding: 12px;
            border-bottom: 1px solid var(--border-color);
            cursor: pointer;
            transition: background 0.15s ease;
        }
        .whitelist-item:last-child { border-bottom: none; }
        .whitelist-item:active { background: var(--hover-bg); }
        .whitelist-item input[type="checkbox"] {
            width: 20px;
            height: 20px;
            margin-right: 12px;
            accent-color: var(--color-primary);
        }
        .whitelist-item-name { flex: 1; font-size: 0.95rem; }
          .whitelist-item-pkg { font-size: 0.75rem; color: var(--text-color-light);
margin-top: 2px; }

        /* [v5.2.0] 应用选择下拉列表 */
        .app-dropdown {
            position: absolute;
            top: 100%;
            left: 0;
            right: 0;
            max-height: 200px;
            overflow-y: auto;
            background: var(--card-bg);
            border: 1px solid var(--border-color);
            border-radius: 8px;
            box-shadow: 0 4px 12px var(--card-shadow);
            z-index: 100;
            margin-top: 4px;
        }
        .app-dropdown.hidden { display: none; }
        .app-dropdown-item {
            padding: 10px 12px;
            cursor: pointer;
            border-bottom: 1px solid var(--border-color);
            transition: background 0.15s ease;
        }
        .app-dropdown-item:last-child { border-bottom: none; }
                .app-dropdown-item:hover,  .app-dropdown-item:active  {  background:
var(--hover-bg); }
        .app-dropdown-item-name { font-size: 0.95rem; color: var(--text-color); }
        .app-dropdown-item-pkg { font-size: 0.75rem; color: var(--text-color-light);
margin-top: 2px; }

        .tab-content { display: none; }
        .tab-content.active { display: block; }

        .section-title {
            font-size: 1.1rem;
            font-weight: 600;
            margin: var(--space-xl) 0 var(--space-lg) 0;
            color: var(--section-title-color);
        }

        /* [v4.4.0] 布局 section 标题和新按钮 */
        .section-title-container {
            display: flex;
            justify-content: space-between;
            align-items: center;
             margin: var(--space-xl) 0 var(--space-lg) 0; /* 复制原 section-title 的
margin */
        }
        .section-title-container .section-title {
            margin: 0; /* 移除 section-title 的默认 margin */
        }
        .btn-highlight-habits-v2 {
            background: none;
            border: 1px solid var(--section-title-color);
            color: var(--section-title-color);
            opacity: 0.8;
            border-radius: 6px;
            padding: 4px 8px;
            font-size: 0.8rem;
            font-weight: 500;
            cursor: pointer;
            flex-shrink: 0;
            white-space: nowrap;
        }
        /* [v5.4.0] 仅桌面端启用 hover，移动端用 active */
        @media (hover: hover) and (pointer: fine) {
            .btn-highlight-habits-v2:hover {
                opacity: 1;
                background: rgba(255,255,255,0.1);
            }
            [data-theme="light"] .btn-highlight-habits-v2:hover {
                background: rgba(0,0,0,0.05);
            }
        }
                    .btn-highlight-habits-v2:active   {   opacity:   1;   background:
rgba(255,255,255,0.1); }
        [data-theme="light"] .btn-highlight-habits-v2 {
            border-color: #fff;
            color: #fff;
        }
                [data-theme="light"]  .btn-highlight-habits-v2:active  {  background:
rgba(0,0,0,0.05); }

        .recent-tasks-grid, .category-tasks-grid {
            display: grid;
            grid-template-columns: 1fr 1fr;
            gap: var(--space-md);
        }
        @media (min-width: 480px) {
            .recent-tasks-grid, .category-tasks-grid {
                gap: var(--space-lg);
            }
        }

        .task-card {
            background: var(--card-bg);
            border-radius: 12px;
            padding: var(--space-md);
            box-shadow: 0 4px 16px var(--card-shadow);
            backdrop-filter: blur(10px);
            position: relative; /* Needed for ::before */
            overflow: visible;
            display: flex;
            flex-direction: column;
            gap: var(--space-xs);
            min-width: 0;
        }

        /* [v3.12.6] Add default border to all cards for visual consistency */
        .task-card::before {
            content: '';
            position: absolute;
            left: 0;
            top: 0;
            bottom: 0;
            right: 0;
            box-shadow: inset 0 0 0 var(--habit-border-width) var(--border-color); /*
Default border */
            border-radius: 12px;
            pointer-events: none;
        }

        /* [v3.12.5] Changed to full inset border shadow */
        /* [v3.12.6] Simplified to only override the color */
        .task-card.is-habit::before {
            /* All other properties are inherited from .task-card::before */
            box-shadow: inset 0 0 0 var(--habit-border-width) var(--habit-color, #ccc);
/* Habit color override */
        }

        /* [v4.4.0] 高亮未完成习惯的样式 (高优先级) */
        .task-card.is-habit.highlight-incomplete::before {
                            box-shadow:  inset  0  0  0  var(--habit-border-width)
var(--color-negative) !important; /* 强制红色边框 */
        }

        /* [v4.12.0] 任务卡片拖动排序样式 */
        .category-tasks-grid {
            -webkit-touch-callout: none;
            position: relative;
        }
        .task-card {
            -webkit-touch-callout: none;
            -webkit-user-select: none;
            user-select: none;
            touch-action: manipulation; /* 默认允许滚动 */
            will-change: transform; /* 提示 GPU 合成 */
        }
        .task-card.task-dragging {
            opacity: 0.95;
            box-shadow: 0 16px 40px rgba(0, 0, 0, 0.3);
            z-index: 100;
            position: relative;
            touch-action: none; /* 拖动时阻断所有默认触摸 */
            transition: none !important; /* 强制移除过渡 */
            transform: translate3d(0, 0, 0); /* 初始 GPU 层 */
        }
        .task-card.task-dragging::after {
            content: '';
            position: absolute;
            inset: 0;
            border-radius: 12px;
            box-shadow: inset 0 0 0 3px var(--color-primary);
            pointer-events: none;
        }
        /* [v7.2.3] 拖拽悬停目标样式（HTML5 拖拽 API） */
        .task-card.drag-over {
            outline: 2px dashed var(--color-primary);
            outline-offset: 2px;
            background: var(--color-primary-light, rgba(76, 175, 80, 0.1));
        }
        /* 占位符样式：防止布局塌缩 */
        .task-card-placeholder {
            visibility: hidden;
            pointer-events: none;
        }
        /* [v5.0.0] 被挤压移动的卡片 - 使用 CSS transition，0.4s 丝滑动画 */
        .task-card.task-shifting {
            transition: transform 0.4s cubic-bezier(0.22, 1, 0.36, 1);
            will-change: transform;
        }
        /* 禁用 transition 的状态 */
        .task-card.task-shift-instant {
            transition: none !important;
        }

        /* [v6.0.0] 任务卡片通透模式 - 参照屏幕时间卡片 */
        .task-card.glass {
                background: linear-gradient(135deg, rgba(255,255,255, calc(0.08 *
    var(--glass-opacity-scale))) 0%, rgba(255,255,255, calc(0.02 *
    var(--glass-opacity-scale))) 100%);
                    border: 1px solid rgba(255, 255, 255, calc(0.3 *
    var(--glass-opacity-scale)));
                         border-top: 1px solid rgba(255, 255, 255, calc(0.5 *
    var(--glass-opacity-scale)));
            box-shadow: 0 4px 16px rgba(0, 0, 0, 0.12);
            backdrop-filter: blur(calc(16px * var(--glass-blur-scale)));
            -webkit-backdrop-filter: blur(calc(16px * var(--glass-blur-scale)));
        }
        .task-card.glass::before {
            display: none !important; /* 完全隐藏边框伪元素，包括习惯任务 */
        }
        [data-theme="dark"] .task-card.glass {
            background: linear-gradient(135deg, rgba(255,255,255, calc(0.06 * var(--glass-opacity-scale))) 0%, rgba(255,255,255, calc(0.01 * var(--glass-opacity-scale))) 100%);
            border: 1px solid rgba(255, 255, 255, calc(0.12 * var(--glass-opacity-scale)));
            border-top: 1px solid rgba(255, 255, 255, calc(0.2 * var(--glass-opacity-scale)));
            box-shadow: 0 4px 16px rgba(0, 0, 0, 0.25);
            backdrop-filter: blur(calc(16px * var(--glass-blur-scale)));
            -webkit-backdrop-filter: blur(calc(16px * var(--glass-blur-scale)));
        }
        /* 通透模式文字 - 任务名称 */
        .task-card.glass .task-name {
            color: white;
            text-shadow: 0 1px 3px rgba(0, 0, 0, 0.5);
        }
        /* 通透模式文字 - 第二行时间/状态 */
        .task-card.glass .task-time,
        .task-card.glass .task-description {
            color: rgba(255, 255, 255, 0.85);
            text-shadow: 0 1px 2px rgba(0, 0, 0, 0.4);
        }
        /* 通透模式文字 - 第三行参数/详情（增强可读性） */
        .task-card.glass .task-parameters {
            color: rgba(255, 255, 255, 0.95);
            text-shadow: 0 1px 3px rgba(0, 0, 0, 0.6);
            font-weight: 500;
        }
        .task-card.glass .more-btn {
            color: rgba(255, 255, 255, 0.7);
            /* [v6.2.4-Cleanup] 标签改为 div 后，无需复杂的重置代码 */
            background: transparent;
            border-radius: 4px;

            /* 确保层级正确 */
            position: relative;
            z-index: 2;
        }

        /* 恢复点击/悬停时的高亮反馈 */
        .task-card.glass .more-btn:hover,
        .task-card.glass .more-btn:focus-visible {
            background-color: rgba(255, 255, 255, 0.15);
            color: rgba(255, 255, 255, 0.95);
        }
        .task-card.glass .more-btn:active {
            background-color: rgba(255, 255, 255, 0.25);
            color: white;
            transition: background-color 0.1s ease;
        }
        /* 通透模式按钮基础样式 */
        .task-card.glass .task-btn {
            background: rgba(255, 255, 255, 0.15);
            color: white;
            border: 1px solid rgba(255, 255, 255, 0.25);
        }
        .task-card.glass .task-btn.primary {
            background: rgba(var(--color-primary-rgb), 0.6);
        }
        .task-card.glass .task-btn.success {
            background: rgba(76, 175, 80, 0.6);
        }
        .task-card.glass .task-btn.warning {
            background: rgba(255, 152, 0, 0.6); /* 橙黄色暂停按钮 */
        }
        .task-card.glass .task-btn.danger {
            background: rgba(244, 67, 54, 0.6);
        }
        .task-card.glass .task-btn.secondary {
            background: rgba(120, 120, 120, 0.4); /* 灰色取消按钮，更淡 */
            border: 1px solid rgba(255, 255, 255, 0.15);
        }
        /* 拖动时隐藏边框效果 */
        .task-card.glass.task-dragging {
            border: 1px solid rgba(255, 255, 255, 0.4);
            box-shadow: 0 16px 40px rgba(0, 0, 0, 0.3);
        }
        .task-card.glass.task-dragging::after {
            display: none; /* 隐藏拖动时的蓝色边框 */
        }
        /* [v6.4.6] 全局浮动菜单通透模式 - 与弹窗 modal-content 样式一致，含最小值保障 */
        body.glass-mode .global-task-menu {
                  background: linear-gradient(135deg, rgba(255,255,255, calc(0.12 *
max(var(--glass-opacity-scale), 0.2))) 0%, rgba(255,255,255, calc(0.06 *
max(var(--glass-opacity-scale), 0.2))) 100%) !important;
                           border: 1px solid rgba(255, 255, 255, calc(0.3 *
max(var(--glass-opacity-scale), 0.2))) !important;
                         border-top: 1px solid rgba(255, 255, 255, calc(0.5 *
max(var(--glass-opacity-scale), 0.2))) !important;
                     backdrop-filter: blur(calc(25px * max(var(--glass-blur-scale),
0.2))) !important;
              -webkit-backdrop-filter: blur(calc(25px * max(var(--glass-blur-scale),
0.2))) !important;
            box-shadow: 0 8px 32px rgba(0, 0, 0, 0.2) !important;
        }
        body.glass-mode .global-task-menu-item {
            color: white;
            text-shadow: 0 1px 2px rgba(0, 0, 0, 0.4);
        }
        body.glass-mode .global-task-menu-item:hover,
        body.glass-mode .global-task-menu-item:active {
            background: rgba(255, 255, 255, 0.15);
        }
        [data-theme="dark"] body.glass-mode .global-task-menu {
            background: linear-gradient(135deg, rgba(255,255,255, calc(0.08 * max(var(--glass-opacity-scale), 0.2))) 0%, rgba(255,255,255, calc(0.03 * max(var(--glass-opacity-scale), 0.2))) 100%) !important;
            border: 1px solid rgba(255, 255, 255, calc(0.15 * max(var(--glass-opacity-scale), 0.2))) !important;
            border-top: 1px solid rgba(255, 255, 255, calc(0.25 * max(var(--glass-opacity-scale), 0.2))) !important;
            box-shadow: 0 8px 32px rgba(0, 0, 0, 0.4) !important;
        }
        [data-theme="dark"] body.glass-mode .global-task-menu-item:hover,
        [data-theme="dark"] body.glass-mode .global-task-menu-item:active {
            background: rgba(255, 255, 255, 0.1);
        }
        /* 非通透模式菜单项 hover */
        @media (hover: hover) and (pointer: fine) {
            .global-task-menu-item:hover { background: rgba(0,0,0,0.05); }
            [data-theme="dark"] .global-task-menu-item:hover { background: rgba(255,255,255,0.1); }
        }
        .global-task-menu-item:active { background: rgba(0,0,0,0.05); }
        [data-theme="dark"] .global-task-menu-item:active { background: rgba(255,255,255,0.1); }

        .task-row { display: flex; align-items: center; min-width: 0; }
         .task-row.title-row { justify-content: space-between; gap: var(--space-sm); margin-bottom: var(--space-xs); }

        /* [v3.17.1] UI Change: Limit task name width and prevent wrap */
        .task-name {
            font-size: 0.9rem;
            font-weight: 600;
            flex: 1;
            /* word-wrap: break-word; */ /* [v3.17.1] Removed */
            /* white-space: normal; */ /* [v3.17.1] Removed */
            white-space: nowrap; /* [v3.17.1] Added */
            overflow: hidden; /* [v3.17.1] Added */
            text-overflow: clip; /* [v3.17.1] Added */
            max-width: 7.5em; /* [v3.17.1] Added */
        }
        /* [v3.17.1] Removed @media rule for 1080px as requested */
        .task-details { font-size: 0.75rem; color: var(--text-color-light); display: flex; flex-wrap: wrap; align-items: center; gap: 6px; }
            /* [v5.1.0] 参数行单行显示，超出省略；[v5.9.0] 固定高度=徽章高度(21px)+padding(4px)，防止布局跳动 */
            .task-parameters { font-size: 0.85rem; color: var(--text-color-light); justify-content: flex-start; padding: 2px 0; display: flex; flex-wrap: nowrap; align-items: center; overflow: hidden; gap: 6px; min-height: 25px; }
        .task-parameters > span { white-space: nowrap; overflow: hidden; text-overflow: ellipsis; min-width: 0; }
        /* [v5.1.0] 分类标签使用渐变（左深右浅） */
        .task-category { color: white; padding: 2px 8px; border-radius: 10px; font-size: 0.7rem; font-weight: 500; background: var(--category-gradient, var(--color-primary)); }
           .task-completion-count { color: var(--color-positive); font-weight: 500; font-size: 0.7rem; }

        .task-actions { display: flex; gap: var(--space-xs); width: 100%; margin-top: var(--space-xs); }
        .task-btn { padding: 8px; border: none; border-radius: 8px; font-size: 0.85rem; font-weight: 600; cursor: pointer; white-space: nowrap; text-align: center; flex: 1; }
        .task-btn.wide { padding: 10px; }
        /* [v5.4.0] 仅桌面端启用 hover，移动端用 active */
        @media (hover: hover) and (pointer: fine) {
            .task-btn:hover { transform: translateY(-1px); box-shadow: 0 2px 8px rgba(0, 0, 0, 0.2); }
        }
        .task-btn:active { transform: translateY(-1px); box-shadow: 0 2px 8px rgba(0, 0, 0, 0.2); }
                .task-btn.primary { background: var(--color-primary); color: white; } .task-btn.success { background: var(--color-positive); color: white; } .task-btn.warning { background: var(--color-warning); color: white; } .task-btn.danger { background: var(--color-negative); color: white; } .task-btn.secondary { background: #9E9E9E; color: white; }

        /* [v5.1.0] 运行中任务的悬浮徽章（内联胶囊，不撑高布局） */
        .task-timer-badge {
            display: inline-flex;
            align-items: center;
            gap: 6px;
            padding: 4px 7px;
            border-radius: 999px;
            background: var(--accent-gradient);
            color: #fff;
            font-size: 0.78rem;
            font-weight: 700;
            line-height: 1;
            white-space: nowrap;
            overflow: hidden;
            text-overflow: ellipsis;
            pointer-events: none;
        }
        [data-theme="dark"] .task-timer-badge {
            background: var(--accent-gradient);
            color: #f8f8f8;
        }
        .task-timer-badge.with-progress {
            font-size: 0.74rem;
        }
        .task-parameters.has-timer-badge {
            justify-content: space-between;
            align-items: center;
            gap: var(--space-sm);
        }

        .more-btn {
            background: none;
            border: none;
            font-size: 1.2rem;
            font-weight: bold;
            cursor: pointer;
            color: var(--text-color-light);
            padding: 0 4px;
            border-radius: 4px;
            flex-shrink: 0;
            line-height: 1;
            -webkit-tap-highlight-color: transparent;

            /* [v6.2.4-Fix] 适配 div 标签：增加布局属性 */
            display: inline-flex;
            align-items: center;
            justify-content: center;
            user-select: none;
            -webkit-user-select: none;
        }
        /* [v5.4.0] 仅桌面端启用 hover */
        @media (hover: hover) and (pointer: fine) {
                    .more-btn:hover  {  background:  var(--btn-secondary-bg);  color:
var(--text-color); }
        }
                 .more-btn:active  {  background:  var(--btn-secondary-bg);  color:
var(--text-color); }
        .more-btn:focus { outline: none; }
              .more-btn:focus-visible  {  outline:  2px  solid  var(--primary-color);
outline-offset: 2px; border-radius: 6px; background: var(--btn-secondary-bg); color:
var(--text-color); }
        /* [v6.4.6] 任务卡片内嵌菜单（隐藏，仅作为数据容器） */
        .task-card-menu { display: none !important; }
        /* [v6.4.6] 全局浮动菜单 - 独立于任务卡片，backdrop-filter 可正常工作 */
        .global-task-menu {
            display: none;
            position: fixed;
            background: var(--dropdown-bg);
            border-radius: 8px;
            box-shadow: 0 4px 12px var(--dropdown-shadow);
            z-index: 1001;
            overflow-y: auto;
            border: 1px solid var(--border-color);
        }
        .global-task-menu.show { display: flex; flex-direction: column; }
        /* [v6.4.6] 菜单项自适应高度 */
            .global-task-menu-item { padding: 8px 15px; font-size: 0.85rem; color: var(--text-color); cursor: pointer; white-space: nowrap; outline: none; -webkit-tap-highlight-color: transparent; display: flex; align-items: center; min-height: 36px; }
        @media (hover: hover) and (pointer: fine) {
            .global-task-menu-item:hover { background: rgba(0,0,0,0.05); }
                    [data-theme="dark"] .global-task-menu-item:hover { background: rgba(255,255,255,0.1); }
        }
        .global-task-menu-item:active { background: rgba(0,0,0,0.05); }
                  [data-theme="dark"] .global-task-menu-item:active { background: rgba(255,255,255,0.1); }
        .global-task-menu-item:focus-visible { background: rgba(0,0,0,0.06); }
             [data-theme="dark"] .global-task-menu-item:focus-visible { background: rgba(255,255,255,0.14); }

        /* [v6.4.0] 自定义下拉选择器 - 参考 task-card-menu 样式 */
        .custom-select-wrapper { position: relative; width: 100%; }
        .custom-select-trigger {
            display: flex;
            align-items: center;
            justify-content: space-between;
            width: 100%;
            padding: var(--space-md);
            border: 2px solid var(--border-color);
            border-radius: 8px;
            background: var(--input-bg);
            color: var(--text-color);
            cursor: pointer;
            font-size: 1rem;
            -webkit-tap-highlight-color: transparent;
        }
        .custom-select-trigger::after {
            content: '▼';
            font-size: 0.6em;
            color: var(--text-color-light);
            margin-left: 8px;
        }
                    .custom-select-trigger:focus  {  outline:  none;  border-color:
var(--color-primary); }
        .dropdown-menu {
            position: absolute;
            top: calc(100% + 4px);
            left: 0;
            right: 0;
            background: var(--dropdown-bg);
            border-radius: 8px;
            box-shadow: 0 4px 12px var(--dropdown-shadow);
            z-index: 100;
            overflow: hidden;
            border: 1px solid var(--border-color);
            display: none;
        }
        .dropdown-menu.show { display: block; }
        .dropdown-menu-item {
            padding: 10px 15px;
            font-size: 0.9rem;
            color: var(--text-color);
            cursor: pointer;
            white-space: nowrap;
            -webkit-tap-highlight-color: transparent;
        }
        @media (hover: hover) and (pointer: fine) {
            .dropdown-menu-item:hover { background: rgba(0,0,0,0.05); }
                       [data-theme="dark"]  .dropdown-menu-item:hover  {  background:
rgba(255,255,255,0.1); }
        }
        .dropdown-menu-item:active { background: rgba(0,0,0,0.05); }
                    [data-theme="dark"]   .dropdown-menu-item:active   {   background:
rgba(255,255,255,0.1); }
        .dropdown-menu-item.selected {
            background: rgba(var(--color-primary-rgb), 0.1);
            color: var(--color-primary);
            font-weight: 500;
        }
        /* [v6.4.0] 通透模式下拉菜单 - 自定义下拉选择器专用样式 */
        /* 触发器样式 - 与普通输入框一致 */
        body.glass-mode .custom-select-trigger,
        body.glass-mode .custom-select-wrapper .custom-select-trigger {
            background: rgba(255, 255, 255, 0.08) !important;
            border: 1px solid rgba(255, 255, 255, 0.25) !important;
            color: white !important;
        }
        body.glass-mode .custom-select-trigger::after,
        body.glass-mode .custom-select-wrapper .custom-select-trigger::after {
            color: rgba(255,255,255,0.7);
        }
        /* 下拉菜单样式 - 参考任务卡片本身的毛玻璃效果 */
        body.glass-mode .custom-select-wrapper .dropdown-menu {
            background: linear-gradient(
                135deg,
                rgba(87, 87, 87, calc(0.8 * max(var(--glass-opacity-scale), 0.2))) 0%,
                rgba(88, 88, 88, calc(0.8 * max(var(--glass-opacity-scale), 0.2))) 100%
            ) !important;
                           border:  1px  solid  rgba(255,  255,  255,  calc(0.25  *
max(var(--glass-opacity-scale), 0.2))) !important;
                    backdrop-filter:  blur(calc(25px  *  max(var(--glass-blur-scale),
0.2))) !important;
              -webkit-backdrop-filter: blur(calc(25px * max(var(--glass-blur-scale),
0.2))) !important;
            box-shadow: 0 8px 32px rgba(0, 0, 0, 0.4) !important;
        }
        body.glass-mode .custom-select-wrapper .dropdown-menu-item {
            color: white !important;
            padding: 12px 16px;
            background: transparent !important;
            text-shadow: 0 1px 2px rgba(0, 0, 0, 0.5);
        }
        body.glass-mode .custom-select-wrapper .dropdown-menu-item:hover,
        body.glass-mode .custom-select-wrapper .dropdown-menu-item:active {
            background: rgba(255,255,255,0.12) !important;
        }
        body.glass-mode .custom-select-wrapper .dropdown-menu-item.selected {
            background: rgba(var(--color-primary-rgb), 0.4) !important;
            color: white !important;
            font-weight: 500;
        }

        /* [v6.4.0] 任务类型选择弹窗样式 */
        #taskTypeModal { z-index: 2100 !important; }
        .task-type-section { margin-bottom: var(--space-lg); }
        .task-type-group-label {
            font-size: 0.85rem;
            font-weight: 600;
            color: var(--text-color-light);
            margin-bottom: var(--space-sm);
            padding-left: 4px;
        }
        .task-type-option {
            display: flex;
            align-items: center;
            gap: var(--space-md);
            padding: var(--space-md);
            border-radius: 10px;
            cursor: pointer;
            margin-bottom: var(--space-sm);
            background: var(--btn-secondary-bg);
            border: 2px solid transparent;
            transition: all 0.2s ease;
            -webkit-tap-highlight-color: transparent;
        }
        .task-type-option:last-child { margin-bottom: 0; }
        @media (hover: hover) and (pointer: fine) {
            .task-type-option:hover {
                border-color: var(--color-primary);
                background: rgba(var(--color-primary-rgb), 0.08);
            }
        }
        .task-type-option:active {
            border-color: var(--color-primary);
            background: rgba(var(--color-primary-rgb), 0.08);
        }
        .task-type-option.selected {
            border-color: var(--color-primary);
            background: rgba(var(--color-primary-rgb), 0.12);
        }
        .task-type-option-icon { font-size: 1.5rem; width: 40px; text-align: center; }
        .task-type-option-info { flex: 1; }
        .task-type-option-name { font-weight: 600; font-size: 0.95rem; margin-bottom:
2px; }
        .task-type-option-desc { font-size: 0.8rem; color: var(--text-color-light); }
        /* 通透模式 */
        body.glass-mode .task-type-option {
            background: rgba(255, 255, 255, 0.1) !important;
            border-color: rgba(255, 255, 255, 0.15);
        }
        body.glass-mode .task-type-option:hover,
        body.glass-mode .task-type-option:active {
            border-color: rgba(var(--color-primary-rgb), 0.7) !important;
            background: rgba(var(--color-primary-rgb), 0.2) !important;
        }
        body.glass-mode .task-type-option.selected {
            border-color: var(--color-primary) !important;
            background: rgba(var(--color-primary-rgb), 0.25) !important;
        }
        body.glass-mode .task-type-group-label { color: rgba(255,255,255,0.7); }
        body.glass-mode .task-type-option-name { color: white; text-shadow: 0 1px 2px
rgba(0,0,0,0.35); }
        body.glass-mode .task-type-option-desc { color: rgba(255,255,255,0.78); }
        /* [v6.4.1] 戒除习惯奖励小字通透模式适配 */
        body.glass-mode .reward-desc-text { color: rgba(255,255,255,0.7); }

        /* [v6.4.0] 底部抽屉式弹窗样式 */
        String channelId = "floating_timer_channel";
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel channel = new NotificationChannel(
                    channelId, "悬浮窗服务", NotificationManager.IMPORTANCE_LOW);
            NotificationManager manager = getSystemService(NotificationManager.class);
            manager.createNotificationChannel(channel);
        }
        Notification.Builder builder;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            builder = new Notification.Builder(this, channelId);
        } else {
            builder = new Notification.Builder(this);
        }
        return              builder.setContentTitle("TimeBank").setContentText("Timer
Running").setSmallIcon(R.mipmap.ic_launcher).build();
    }

    /**
     * [v5.8.1] 处理屏幕旋转
     */
    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);

        // 切换到对应屏幕方向的保存位置
        if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
            currentX = landscapeX;
            currentY = landscapeY;
        } else {
            currentX = portraitX;
            currentY = portraitY;
        }

        // 更新所有悬浮窗位置
        updateAllPositions();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        longPressHandler.removeCallbacksAndMessages(null);
        for (TimerInfo info : timerMap.values()) {
            if (info.timerRunnable != null) {
                handler.removeCallbacks(info.timerRunnable);
            }
            if (info.view != null) {
                try { windowManager.removeView(info.view); } catch (Exception e) {}
            }
        }
        timerMap.clear();
        timerOrder.clear();
    }
}
package com.jianglicheng.timebank;

import android.app.AppOpsManager;
import android.Manifest;
import android.app.usage.UsageStats;
import android.app.usage.UsageStatsManager;
import android.appwidget.AppWidgetManager;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Build;
import android.os.Environment;
import android.os.PowerManager;
import android.os.VibrationEffect;
import android.os.Vibrator;
import android.provider.Settings;
import android.util.Base64;
import android.webkit.JavascriptInterface;
import android.widget.Toast;
import androidx.core.content.ContextCompat;

import org.json.JSONArray;
import org.json.JSONObject;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Collections;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class WebAppInterface {
    Context mContext;

    WebAppInterface(Context c) {
        mContext = c;
    }
    // [v7.2.1] 获取设备唯一标识符（用于屏幕时间多设备去重）
    @JavascriptInterface
    public String getDeviceId() {
        try {
            return          Settings.Secure.getString(mContext.getContentResolver(),
Settings.Secure.ANDROID_ID);
        } catch (Exception e) {
            e.printStackTrace();
            return "unknown";
        }
    }

    // [v5.7.0] 震动反馈接口
    @JavascriptInterface
    public void vibrate(int milliseconds) {
        try {
            Vibrator                vibrator                =               (Vibrator)
mContext.getSystemService(Context.VIBRATOR_SERVICE);
            if (vibrator != null && vibrator.hasVibrator()) {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
                    vibrator.vibrate(VibrationEffect.createOneShot(milliseconds,
VibrationEffect.DEFAULT_AMPLITUDE));
                } else {
                    vibrator.vibrate(milliseconds);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // 直接保存 JSON 字符串到下载目录
    @JavascriptInterface
    public void saveFileDirectly(String jsonContent, String fileName) {
        try {
            // 保存到下载目录（若重名则自动追加 (1), (2) ...）
            File file = getUniqueDownloadFile(fileName);

            FileOutputStream fos = new FileOutputStream(file);
            fos.write(jsonContent.getBytes("UTF-8"));
            fos.close();

            // 在主线程显示 Toast
            android.os.Handler               handler               =               new
android.os.Handler(mContext.getMainLooper());
            handler.post(() -> Toast.makeText(mContext, "✅ 已保存到: Download/" +
file.getName(), Toast.LENGTH_LONG).show());
        } catch (Exception e) {
            e.printStackTrace();
            android.os.Handler handler = new
android.os.Handler(mContext.getMainLooper());
            handler.post(() -> Toast.makeText(mContext, "❌ 保存失败: " + e.getMessage(),
Toast.LENGTH_LONG).show());
        }
    }

    // 保存文件到下载目录 (base64 版本)
    @JavascriptInterface
    public void saveFile(String dataUrl, String fileName) {
        try {
            // 解析 data URL
            String base64Data = dataUrl.substring(dataUrl.indexOf(",") + 1);
            byte[] data = Base64.decode(base64Data, Base64.DEFAULT);

            // 生成文件名
            String timestamp = new java.text.SimpleDateFormat("yyyy-MM-dd",
java.util.Locale.getDefault()).format(new java.util.Date());
            String finalFileName = "timebank_backup_" + timestamp + ".json";

            // 保存到下载目录（若重名则自动追加 (1), (2) ...）
            File file = getUniqueDownloadFile(finalFileName);

            FileOutputStream fos = new FileOutputStream(file);
            fos.write(data);
            fos.close();

            Toast.makeText(mContext, "✅ 已保存到: Download/" + file.getName(),
Toast.LENGTH_LONG).show();
        } catch (Exception e) {
            e.printStackTrace();
            Toast.makeText(mContext, "❌ 保存失败: " + e.getMessage(),
Toast.LENGTH_LONG).show();
        }
    }

    // 生成不重名的下载文件名：file.json -> file (1).json
    private File getUniqueDownloadFile(String fileName) {
        File downloadsDir =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
        String baseName = fileName;
        String extension = "";
        int dotIndex = fileName.lastIndexOf('.');
        if (dotIndex > 0 && dotIndex < fileName.length() - 1) {
            baseName = fileName.substring(0, dotIndex);
            extension = fileName.substring(dotIndex);
        }

        File file = new File(downloadsDir, fileName);
        int counter = 1;
        while (file.exists()) {
            String candidate = baseName + " (" + counter + ")" + extension;
            file = new File(downloadsDir, candidate);
            counter++;
        }
        return file;
    }

    // 发送普通通知
    @JavascriptInterface
    public void showNotification(String title, String message) {
        Intent intent = new Intent(mContext, AlarmReceiver.class);
        intent.setAction("com.jianglicheng.timebank.SHOW_NOTIFICATION");
        intent.putExtra("title", title);
        intent.putExtra("message", message);
        mContext.sendBroadcast(intent);
    }

    // 开启悬浮窗
    @JavascriptInterface
    public void startFloatingTimer(String taskName, int durationSeconds, String colorHex)
{
        if          (Build.VERSION.SDK_INT          >=          Build.VERSION_CODES.M
&& !Settings.canDrawOverlays(mContext)) {
            Intent intent = new Intent(Settings.ACTION_MANAGE_OVERLAY_PERMISSION,
                    Uri.parse("package:" + mContext.getPackageName()));
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            mContext.startActivity(intent);
            return;
        }

        Intent serviceIntent = new Intent(mContext, FloatingTimerService.class);
        serviceIntent.putExtra("TASK_NAME", taskName);
        serviceIntent.putExtra("DURATION", durationSeconds);
        serviceIntent.putExtra("COLOR", colorHex);

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            mContext.startForegroundService(serviceIntent);
        } else {
            mContext.startService(serviceIntent);
        }
    }

    // 停止悬浮窗 [v5.3.0] 支持按任务名称停止特定计时器
    @JavascriptInterface
    public void stopFloatingTimer(String taskName) {
        Intent serviceIntent = new Intent(mContext, FloatingTimerService.class);
        serviceIntent.putExtra("ACTION", "STOP");
        serviceIntent.putExtra("TASK_NAME", taskName);
        mContext.startService(serviceIntent);
    }

    // [v5.8.1] 暂停悬浮窗计时器
    @JavascriptInterface
    public void pauseFloatingTimer(String taskName) {
        Intent serviceIntent = new Intent(mContext, FloatingTimerService.class);
        serviceIntent.putExtra("ACTION", "PAUSE");
        serviceIntent.putExtra("TASK_NAME", taskName);
        mContext.startService(serviceIntent);
    }

    // [v5.8.1] 恢复悬浮窗计时器
    @JavascriptInterface
    public void resumeFloatingTimer(String taskName) {
        Intent serviceIntent = new Intent(mContext, FloatingTimerService.class);
        serviceIntent.putExtra("ACTION", "RESUME");
        serviceIntent.putExtra("TASK_NAME", taskName);
        mContext.startService(serviceIntent);
    }

    // 悬浮窗权限检查
    @JavascriptInterface
    public boolean canDrawOverlays() {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) return true;
        return Settings.canDrawOverlays(mContext);
    }

    // 跳转到悬浮窗权限设置
    @JavascriptInterface
    public void openOverlaySettings() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            Intent intent = new Intent(Settings.ACTION_MANAGE_OVERLAY_PERMISSION,
                    Uri.parse("package:" + mContext.getPackageName()));
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            mContext.startActivity(intent);
        }
    }

    // [v7.9.9] 获取导航栏高度（用于底部栏避让）
    @JavascriptInterface
    public int getNavigationBarHeight() {
        try {
            int                                 resourceId                                =
mContext.getResources().getIdentifier("navigation_bar_height", "dimen", "android");
            if (resourceId > 0) {
                return mContext.getResources().getDimensionPixelSize(resourceId);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return 0;
    }

    // 通知权限检查 (Android 13+)
    @JavascriptInterface
    public boolean hasPostNotificationPermission() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
            return                          ContextCompat.checkSelfPermission(mContext,
Manifest.permission.POST_NOTIFICATIONS)
                    == PackageManager.PERMISSION_GRANTED;
        }
        return true;
    }

    // 打开应用通知设置
    @JavascriptInterface
    public void openAppNotificationSettings() {
        Intent intent;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            intent = new Intent(Settings.ACTION_APP_NOTIFICATION_SETTINGS)
                    .putExtra(Settings.EXTRA_APP_PACKAGE, mContext.getPackageName());
        } else {
            intent = new Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS)
                    .setData(Uri.parse("package:" + mContext.getPackageName()));
        }
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        mContext.startActivity(intent);
    }

    // 精准闹钟授权跳转 (Android 12+)
    @JavascriptInterface
    public void openExactAlarmSettings() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
            Intent intent = new Intent(Settings.ACTION_REQUEST_SCHEDULE_EXACT_ALARM);
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            mContext.startActivity(intent);
        }
    }

    // 电池优化白名单检查
    @JavascriptInterface
    public boolean isIgnoringBatteryOptimizations() {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) return true;
        PowerManager                pm                =                (PowerManager)
mContext.getSystemService(Context.POWER_SERVICE);
        if (pm == null) return false;
        return pm.isIgnoringBatteryOptimizations(mContext.getPackageName());
    }

    // 请求加入电池优化白名单
    @JavascriptInterface
    public void requestIgnoreBatteryOptimizations() {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) return;
        PowerManager                pm                =                (PowerManager)
mContext.getSystemService(Context.POWER_SERVICE);
        if (pm != null && pm.isIgnoringBatteryOptimizations(mContext.getPackageName()))
return;
        Intent                      intent                        =                new
Intent(Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS,
                Uri.parse("package:" + mContext.getPackageName()));
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        mContext.startActivity(intent);
    }

    // 原生闹钟接口：实现精准唤醒
    @JavascriptInterface
    public void scheduleAlarm(String title, String message, long delayMs) {
        try {
            android.app.AlarmManager    alarmManager    =    (android.app.AlarmManager)
mContext.getSystemService(Context.ALARM_SERVICE);

            Intent intent = new Intent(mContext, AlarmReceiver.class);
            intent.setAction("com.jianglicheng.timebank.ALARM_TRIGGER");
            intent.putExtra("title", title);
            intent.putExtra("message", message);

            int flags = android.app.PendingIntent.FLAG_UPDATE_CURRENT;
            if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M) {
                flags |= android.app.PendingIntent.FLAG_IMMUTABLE;
            }

            android.app.PendingIntent                 pendingIntent                 =
android.app.PendingIntent.getBroadcast(mContext, 0, intent, flags);

            long triggerTime = System.currentTimeMillis() + delayMs;

            if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M) {

alarmManager.setExactAndAllowWhileIdle(android.app.AlarmManager.RTC_WAKEUP,
triggerTime, pendingIntent);
            } else {
                alarmManager.setExact(android.app.AlarmManager.RTC_WAKEUP,
triggerTime, pendingIntent);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @JavascriptInterface
    public void cancelAlarm() {
        try {
            Intent intent = new Intent(mContext, AlarmReceiver.class);
            intent.setAction("com.jianglicheng.timebank.ALARM_TRIGGER");
            int flags = android.app.PendingIntent.FLAG_UPDATE_CURRENT;
            if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M) {
                flags |= android.app.PendingIntent.FLAG_IMMUTABLE;
            }
            android.app.PendingIntent                 pendingIntent                =
android.app.PendingIntent.getBroadcast(mContext, 0, intent, flags);
            android.app.AlarmManager   alarmManager   =   (android.app.AlarmManager)
mContext.getSystemService(Context.ALARM_SERVICE);
            alarmManager.cancel(pendingIntent);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // [v7.9.3] 设置带 ID 的闹钟（避免午睡和任务闹钟互相覆盖）
    @JavascriptInterface
    public void scheduleAlarmWithId(int alarmId, String title, String message, long
delayMs) {
        try {
            android.app.AlarmManager   alarmManager   =   (android.app.AlarmManager)
mContext.getSystemService(Context.ALARM_SERVICE);

            Intent intent = new Intent(mContext, AlarmReceiver.class);
            intent.setAction("com.jianglicheng.timebank.ALARM_TRIGGER_" + alarmId);
            intent.putExtra("title", title);
            intent.putExtra("message", message);
            intent.putExtra("alarmId", alarmId);

            int flags = android.app.PendingIntent.FLAG_UPDATE_CURRENT;
            if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M) {
                flags |= android.app.PendingIntent.FLAG_IMMUTABLE;
            }

            android.app.PendingIntent                 pendingIntent                =
android.app.PendingIntent.getBroadcast(mContext, alarmId, intent, flags);

            long triggerTime = System.currentTimeMillis() + delayMs;

            if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M) {
```

```
alarmManager.setExactAndAllowWhileIdle(android.app.AlarmManager.RTC_WAKEUP,
triggerTime, pendingIntent);
            } else {
                alarmManager.setExact(android.app.AlarmManager.RTC_WAKEUP,
triggerTime, pendingIntent);
            }
            android.util.Log.d("TimeBank", "Alarm scheduled with ID " + alarmId + ", delay:
" + delayMs + "ms");
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "scheduleAlarmWithId error", e);
        }
    }

    // [v7.9.3] 取消带 ID 的闹钟
    @JavascriptInterface
    public void cancelAlarmWithId(int alarmId) {
        try {
            Intent intent = new Intent(mContext, AlarmReceiver.class);
            intent.setAction("com.jianglicheng.timebank.ALARM_TRIGGER_" + alarmId);
            int flags = android.app.PendingIntent.FLAG_UPDATE_CURRENT;
            if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M) {
                flags |= android.app.PendingIntent.FLAG_IMMUTABLE;
            }
            android.app.PendingIntent                   pendingIntent                  =
android.app.PendingIntent.getBroadcast(mContext, alarmId, intent, flags);
            android.app.AlarmManager    alarmManager    =    (android.app.AlarmManager)
mContext.getSystemService(Context.ALARM_SERVICE);
            alarmManager.cancel(pendingIntent);
            android.util.Log.d("TimeBank", "Alarm cancelled with ID " + alarmId);
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "cancelAlarmWithId error", e);
        }
    }

    // [v7.9.3] 检查是否有精确闹钟权限
    @JavascriptInterface
    public boolean canScheduleExactAlarms() {
        if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.S) {
            android.app.AlarmManager    alarmManager    =    (android.app.AlarmManager)
mContext.getSystemService(Context.ALARM_SERVICE);
            return alarmManager.canScheduleExactAlarms();
        }
        // Android 12 以下不需要此权限
        return true;
    }

    // [v7.9.3] 跳转到闹钟权限设置页
    @JavascriptInterface
    public void openAlarmSettings() {
        try {
            if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.S) {
                Intent                    intent                    =                    new
Intent(Settings.ACTION_REQUEST_SCHEDULE_EXACT_ALARM);
                intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                mContext.startActivity(intent);
            } else {
                // 旧版本跳转到应用设置页
                Intent                    intent                    =                    new
Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
                intent.setData(Uri.parse("package:" + mContext.getPackageName()));
                intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                mContext.startActivity(intent);
            }
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "openAlarmSettings error", e);
        }
    }

    // [v4.10.0] 新增：启动外部应用
    @JavascriptInterface
    public void launchApp(String packageName) {
        try {
            PackageManager pm = mContext.getPackageManager();
            Intent intent = pm.getLaunchIntentForPackage(packageName);
            if (intent != null) {
                intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                mContext.startActivity(intent);
            } else {
                Toast.makeText(mContext,  " 未 安 装 该 应 用 ：  "  +  packageName,
Toast.LENGTH_SHORT).show();
            }
        } catch (Exception e) {
            e.printStackTrace();
            Toast.makeText(mContext,  " 启 动 应 用 失 败 ：  "  +  e.getMessage(),
Toast.LENGTH_SHORT).show();
        }
    }

    // ========== [v5.2.0] 屏幕时间管理接口 ==========

    /** 检查是否有使用情况访问权限 */
    @JavascriptInterface
    public boolean hasUsageStatsPermission() {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP) {
            return false;
        }
        AppOpsManager              appOps              =              (AppOpsManager)
mContext.getSystemService(Context.APP_OPS_SERVICE);
```

```
        int mode = appOps.checkOpNoThrow(AppOpsManager.OPSTR_GET_USAGE_STATS,
                android.os.Process.myUid(), mContext.getPackageName());
        return mode == AppOpsManager.MODE_ALLOWED;
    }

    /** 跳转到使用情况访问权限设置页 */
    @JavascriptInterface
    public void openUsageAccessSettings() {
        Intent intent = new Intent(Settings.ACTION_USAGE_ACCESS_SETTINGS);
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        mContext.startActivity(intent);
    }

    /**
     * 获取今日屏幕使用时间（毫秒）
     * @param excludedPackagesJson JSON 数组字符串， 如 ["com.example.app1",
"com.example.app2"]
     * @return 使用时间（毫秒），-1 表示无权限，-2 表示异常
     */
    @JavascriptInterface
    public long getTodayScreenTime(String excludedPackagesJson) {
        if (!hasUsageStatsPermission()) {
            return -1;
        }

        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP) {
            return -2;
        }

        try {
            UsageStatsManager usageStatsManager = (UsageStatsManager)
                    mContext.getSystemService(Context.USAGE_STATS_SERVICE);

            // 解析排除列表
            Set<String> excludedPackages = new HashSet<>();
            if (excludedPackagesJson != null && !excludedPackagesJson.isEmpty()) {
                JSONArray jsonArray = new JSONArray(excludedPackagesJson);
                for (int i = 0; i < jsonArray.length(); i++) {
                    excludedPackages.add(jsonArray.getString(i));
                }
            }

            // 今日零点到现在
            Calendar calendar = Calendar.getInstance();
            calendar.set(Calendar.HOUR_OF_DAY, 0);
            calendar.set(Calendar.MINUTE, 0);
            calendar.set(Calendar.SECOND, 0);
            calendar.set(Calendar.MILLISECOND, 0);
            long startTime = calendar.getTimeInMillis();
            long endTime = System.currentTimeMillis();

            // 查询使用统计
            List<UsageStats> stats = usageStatsManager.queryUsageStats(
                    UsageStatsManager.INTERVAL_DAILY, startTime, endTime);

            long totalTime = 0;
            if (stats != null) {
                for (UsageStats usageStats : stats) {
                    if (!excludedPackages.contains(usageStats.getPackageName())) {
                        totalTime += usageStats.getTotalTimeInForeground();
                    }
                }
            }
            return totalTime;
        } catch (Exception e) {
            e.printStackTrace();
            return -2;
        }
    }

    /** 获取已安装应用列表（用于白名单选择） */
    @JavascriptInterface
    public String getInstalledApps() {
        try {
            PackageManager pm = mContext.getPackageManager();
            List<ApplicationInfo> apps = pm.getInstalledApplications(0);

            JSONArray result = new JSONArray();
            for (ApplicationInfo app : apps) {
                // 只返回有启动器图标的应用（用户可见应用）
                if (pm.getLaunchIntentForPackage(app.packageName) != null) {
                    JSONObject obj = new JSONObject();
                    obj.put("packageName", app.packageName);
                    obj.put("appName", pm.getApplicationLabel(app).toString());
                    result.put(obj);
                }
            }
            return result.toString();
        } catch (Exception e) {
            e.printStackTrace();
            return "[]";
        }
    }

    /**
     * [v5.5.0] 获取今日各应用使用时长列表（按时长降序排列）
     * @param excludedPackagesJson 排除的应用包名 JSON 数组
     * @return JSON 数组字符串 [{packageName, appName, timeMs}, ...]，按时长降序
```

```
 */
@JavascriptInterface
public String getAppUsageList(String excludedPackagesJson) {
    if (!hasUsageStatsPermission()) {
        return "[]";
    }

    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP) {
        return "[]";
    }

    try {
        UsageStatsManager usageStatsManager = (UsageStatsManager)
                mContext.getSystemService(Context.USAGE_STATS_SERVICE);
        PackageManager pm = mContext.getPackageManager();

        // 解析排除列表
        Set<String> excludedPackages = new HashSet<>();
        if (excludedPackagesJson != null && !excludedPackagesJson.isEmpty()) {
            JSONArray jsonArray = new JSONArray(excludedPackagesJson);
            for (int i = 0; i < jsonArray.length(); i++) {
                excludedPackages.add(jsonArray.getString(i));
            }
        }

        // 今日零点到现在
        Calendar calendar = Calendar.getInstance();
        calendar.set(Calendar.HOUR_OF_DAY, 0);
        calendar.set(Calendar.MINUTE, 0);
        calendar.set(Calendar.SECOND, 0);
        calendar.set(Calendar.MILLISECOND, 0);
        long startTime = calendar.getTimeInMillis();
        long endTime = System.currentTimeMillis();

        // 查询使用统计
        List<UsageStats> stats = usageStatsManager.queryUsageStats(
                UsageStatsManager.INTERVAL_DAILY, startTime, endTime);

        // 收集有效数据并排序
        List<JSONObject> appUsageList = new ArrayList<>();
        if (stats != null) {
            for (UsageStats usageStats : stats) {
                String packageName = usageStats.getPackageName();
                long timeMs = usageStats.getTotalTimeInForeground();

                // 排除白名单应用和时长为 0 的应用
                if (excludedPackages.contains(packageName) || timeMs <= 0) {
                    continue;
                }
                // 获取应用名称
                String appName;
                try {
                    ApplicationInfo appInfo = pm.getApplicationInfo(packageName, 0);
                    appName = pm.getApplicationLabel(appInfo).toString();
                } catch (PackageManager.NameNotFoundException e) {
                    appName = packageName; // 找不到就用包名
                }

                JSONObject obj = new JSONObject();
                obj.put("packageName", packageName);
                obj.put("appName", appName);
                obj.put("timeMs", timeMs);
                appUsageList.add(obj);
            }
        }

        // 按时长降序排序
        Collections.sort(appUsageList, (a, b) -> {
            try {
                return Long.compare(b.getLong("timeMs"), a.getLong("timeMs"));
            } catch (Exception e) {
                return 0;
            }
        });

        // 转换为 JSONArray
        JSONArray result = new JSONArray();
        for (JSONObject obj : appUsageList) {
            result.put(obj);
        }
        return result.toString();
    } catch (Exception e) {
        e.printStackTrace();
        return "[]";
    }
}

/** 获取单个应用今日使用时间（毫秒） */
@JavascriptInterface
public long getAppScreenTime(String packageName) {
    if (!hasUsageStatsPermission()) {
        return -1;
    }

    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP) {
        return -2;
    }
    try {
        UsageStatsManager usageStatsManager = (UsageStatsManager)
                mContext.getSystemService(Context.USAGE_STATS_SERVICE);

        Calendar calendar = Calendar.getInstance();
        calendar.set(Calendar.HOUR_OF_DAY, 0);
        calendar.set(Calendar.MINUTE, 0);
        calendar.set(Calendar.SECOND, 0);
        calendar.set(Calendar.MILLISECOND, 0);
        long startTime = calendar.getTimeInMillis();
        long endTime = System.currentTimeMillis();

        List<UsageStats> stats = usageStatsManager.queryUsageStats(
                UsageStatsManager.INTERVAL_DAILY, startTime, endTime);

        if (stats != null) {
            for (UsageStats usageStats : stats) {
                if (packageName.equals(usageStats.getPackageName())) {
                    return usageStats.getTotalTimeInForeground();
                }
            }
        }
        return 0;
    } catch (Exception e) {
        e.printStackTrace();
        return -2;
    }
}

/**
 * [v5.2.0] 获取指定日期的屏幕使用时间（用于历史补结算）
 * @param dateString 日期字符串，格式 "YYYY-MM-DD"
 * @param excludedPackagesJson JSON 数组字符串
 * @return 使用时间（毫秒），-1 表示无权限，-2 表示异常
 */
@JavascriptInterface
public long getScreenTimeForDate(String dateString, String excludedPackagesJson) {
    if (!hasUsageStatsPermission()) {
        return -1;
    }

    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP) {
        return -2;
    }

    try {
        UsageStatsManager usageStatsManager = (UsageStatsManager)
                mContext.getSystemService(Context.USAGE_STATS_SERVICE);
```

```
// 解析日期字符串
String[] parts = dateString.split("-");
int year = Integer.parseInt(parts[0]);
int month = Integer.parseInt(parts[1]) - 1; // Calendar 月份从 0 开始
int day = Integer.parseInt(parts[2]);

// 解析排除列表
Set<String> excludedPackages = new HashSet<>();
if (excludedPackagesJson != null && !excludedPackagesJson.isEmpty()) {
    JSONArray jsonArray = new JSONArray(excludedPackagesJson);
    for (int i = 0; i < jsonArray.length(); i++) {
        excludedPackages.add(jsonArray.getString(i));
    }
}

// 指定日期的零点到次日零点
Calendar startCal = Calendar.getInstance();
startCal.set(year, month, day, 0, 0, 0);
startCal.set(Calendar.MILLISECOND, 0);
long startTime = startCal.getTimeInMillis();

Calendar endCal = Calendar.getInstance();
endCal.set(year, month, day, 23, 59, 59);
endCal.set(Calendar.MILLISECOND, 999);
long endTime = endCal.getTimeInMillis();

// 不能查询未来的日期
long now = System.currentTimeMillis();
if (startTime > now) {
    return 0;
}
if (endTime > now) {
    endTime = now;
}

// 查询使用统计
List<UsageStats> stats = usageStatsManager.queryUsageStats(
        UsageStatsManager.INTERVAL_DAILY, startTime, endTime);

long totalTime = 0;
if (stats != null) {
    for (UsageStats usageStats : stats) {
        if (!excludedPackages.contains(usageStats.getPackageName())) {
            totalTime += usageStats.getTotalTimeInForeground();
        }
    }
}
return totalTime;
```

```
    } catch (Exception e) {
        e.printStackTrace();
        return -2;
    }
}

/**
 * [v5.3.0] 获取指定应用在指定日期的使用时间
 * @param packageName 应用包名
 * @param dateString 日期字符串，格式 "YYYY-MM-DD"
 * @return 使用时间（毫秒），-1 表示无权限，-2 表示异常
 */
@JavascriptInterface
public long getAppScreenTimeForDate(String packageName, String dateString) {
    if (!hasUsageStatsPermission()) {
        return -1;
    }

    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP) {
        return -2;
    }

    try {
        UsageStatsManager usageStatsManager = (UsageStatsManager)
                mContext.getSystemService(Context.USAGE_STATS_SERVICE);

        // 解析日期字符串
        String[] parts = dateString.split("-");
        int year = Integer.parseInt(parts[0]);
        int month = Integer.parseInt(parts[1]) - 1;
        int day = Integer.parseInt(parts[2]);

        // 指定日期的零点到次日零点
        Calendar startCal = Calendar.getInstance();
        startCal.set(year, month, day, 0, 0, 0);
        startCal.set(Calendar.MILLISECOND, 0);
        long startTime = startCal.getTimeInMillis();

        Calendar endCal = Calendar.getInstance();
        endCal.set(year, month, day, 23, 59, 59);
        endCal.set(Calendar.MILLISECOND, 999);
        long endTime = endCal.getTimeInMillis();

        // 不能查询未来的日期
        long now = System.currentTimeMillis();
        if (startTime > now) {
            return 0;
        }
        if (endTime > now) {
                endTime = now;
            }

            // 查询使用统计
            List<UsageStats> stats = usageStatsManager.queryUsageStats(
                    UsageStatsManager.INTERVAL_DAILY, startTime, endTime);

            if (stats != null) {
                for (UsageStats usageStats : stats) {
                    if (usageStats.getPackageName().equals(packageName)) {
                        return usageStats.getTotalTimeInForeground();
                    }
                }
            }
            return 0;
        } catch (Exception e) {
            e.printStackTrace();
            return -2;
        }
    }

    // [v5.10.0] 更新桌面小组件数据
    @JavascriptInterface
    public void updateWidgets(long balanceSeconds, int dailyLimitMinutes, String
whitelistAppsJson) {
        try {
            android.util.Log.d("TimeBank", "updateWidgets called: balance=" +
balanceSeconds + ", limit=" + dailyLimitMinutes);

            // 保存数据到 SharedPreferences
            SharedPreferences prefs = mContext.getSharedPreferences("TimeBankWidget",
Context.MODE_PRIVATE);
            SharedPreferences.Editor editor = prefs.edit();
            editor.putLong("currentBalance", balanceSeconds);
            editor.putInt("dailyLimitMinutes", dailyLimitMinutes);
            editor.putString("whitelistApps", whitelistAppsJson);
            editor.commit(); // 使用 commit 确保同步写入

            // 通知所有小组件更新
            AppWidgetManager appWidgetManager =
AppWidgetManager.getInstance(mContext);

            // 更新余额小组件
            int[] balanceWidgetIds = appWidgetManager.getAppWidgetIds(
                    new ComponentName(mContext, BalanceWidgetProvider.class));
            for (int widgetId : balanceWidgetIds) {
                BalanceWidgetProvider.updateAppWidget(mContext, appWidgetManager,
widgetId);
            }
            // 更新屏幕时间经典小组件
            int[] classicWidgetIds = appWidgetManager.getAppWidgetIds(
                    new ComponentName(mContext, ScreenTimeWidgetProvider.class));
            for (int widgetId : classicWidgetIds) {
                ScreenTimeWidgetProvider.updateAppWidget(mContext,  appWidgetManager,
widgetId);
            }

            // 更新屏幕时间通透小组件
            int[] glassWidgetIds = appWidgetManager.getAppWidgetIds(
                    new ComponentName(mContext, ScreenTimeGlassWidgetProvider.class));
            for (int widgetId : glassWidgetIds) {
                // 设置为通透模式
                prefs.edit().putString("screenTimeStyle_" + widgetId, "glass").apply();
                ScreenTimeWidgetProvider.updateAppWidget(mContext,  appWidgetManager,
widgetId);
            }

            android.util.Log.d("TimeBank",   "Widgets   updated:   balance="   +
balanceWidgetIds.length +
                    ",  classic="  +  classicWidgetIds.length  +  ",  glass="  +
glassWidgetIds.length);
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "updateWidgets error", e);
        }
    }

    // [v7.8.3] 保存登录邮箱到 SharedPreferences（比 WebView localStorage 更可靠）
    @JavascriptInterface
    public void saveLoginEmail(String email) {
        try {
            SharedPreferences  prefs  =  mContext.getSharedPreferences("TimeBankAuth",
Context.MODE_PRIVATE);
            SharedPreferences.Editor editor = prefs.edit();
            editor.putString("loginEmail", email);
            editor.putLong("savedAt", System.currentTimeMillis());
            editor.apply();
            android.util.Log.d("TimeBank", "Login email saved: " + email);
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "saveLoginEmail error", e);
        }
    }

    // [v7.8.3] 读取保存的登录邮箱
    @JavascriptInterface
    public String getSavedLoginEmail() {
        try {
            SharedPreferences  prefs  =  mContext.getSharedPreferences("TimeBankAuth",
Context.MODE_PRIVATE);
            return prefs.getString("loginEmail", "");
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "getSavedLoginEmail error", e);
            return "";
        }
    }

    // [v7.8.3] 清除保存的登录邮箱（登出时调用）
    @JavascriptInterface
    public void clearSavedLoginEmail() {
        try {
            SharedPreferences  prefs  =  mContext.getSharedPreferences("TimeBankAuth",
Context.MODE_PRIVATE);
            prefs.edit().remove("loginEmail").remove("savedAt").apply();
            android.util.Log.d("TimeBank", "Login email cleared");
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "clearSavedLoginEmail error", e);
        }
    }

    // [v7.9.3] 保存期望登录状态标记（用于检测意外登出）
    @JavascriptInterface
    public void setExpectedLoggedIn(boolean isLoggedIn) {
        try {
            SharedPreferences  prefs  =  mContext.getSharedPreferences("TimeBankAuth",
Context.MODE_PRIVATE);
            prefs.edit().putBoolean("expectedLoggedIn", isLoggedIn).apply();
            android.util.Log.d("TimeBank", "Expected login state saved: " + isLoggedIn);
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "setExpectedLoggedIn error", e);
        }
    }

    // [v7.9.3] 读取期望登录状态标记
    @JavascriptInterface
    public boolean getExpectedLoggedIn() {
        try {
            SharedPreferences  prefs  =  mContext.getSharedPreferences("TimeBankAuth",
Context.MODE_PRIVATE);
            return prefs.getBoolean("expectedLoggedIn", false);
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "getExpectedLoggedIn error", e);
            return false;
        }
    }

    // [v7.9.4] 保存登录凭据（邮箱 + 加密密码）用于自动重新登录
    // 注意：这里使用 Base64 简单编码，主要是为了防止明文存储
    // 真正的安全性依赖于 Android SharedPreferences 的 MODE_PRIVATE
    @JavascriptInterface
    public void saveLoginCredentials(String email, String password) {
        try {
            SharedPreferences  prefs  =  mContext.getSharedPreferences("TimeBankAuth",
Context.MODE_PRIVATE);
            SharedPreferences.Editor editor = prefs.edit();
            editor.putString("loginEmail", email);
            // 使用 Base64 编码密码（简单混淆，防止明文存储）
            String encodedPassword = Base64.encodeToString(password.getBytes("UTF-8"),
Base64.NO_WRAP);
            editor.putString("loginPasswordEncoded", encodedPassword);
            editor.putLong("credentialsSavedAt", System.currentTimeMillis());
            editor.putBoolean("autoLoginEnabled", true);
            editor.apply();
            android.util.Log.d("TimeBank", "Login credentials saved for: " + email);
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "saveLoginCredentials error", e);
        }
    }

    // [v7.9.4] 读取保存的登录密码（解码）
    @JavascriptInterface
    public String getSavedLoginPassword() {
        try {
            SharedPreferences  prefs  =  mContext.getSharedPreferences("TimeBankAuth",
Context.MODE_PRIVATE);
            String encodedPassword = prefs.getString("loginPasswordEncoded", "");
            if (encodedPassword.isEmpty()) {
                return "";
            }
            byte[] decodedBytes = Base64.decode(encodedPassword, Base64.NO_WRAP);
            return new String(decodedBytes, "UTF-8");
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "getSavedLoginPassword error", e);
            return "";
        }
    }

    // [v7.9.4] 检查是否启用了自动登录
    @JavascriptInterface
    public boolean isAutoLoginEnabled() {
        try {
            SharedPreferences  prefs  =  mContext.getSharedPreferences("TimeBankAuth",
Context.MODE_PRIVATE);
            return prefs.getBoolean("autoLoginEnabled", false);
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "isAutoLoginEnabled error", e);
            return false;
        }
    }

    // [v7.9.4] 清除所有登录凭据（登出或禁用自动登录时调用）
    @JavascriptInterface
    public void clearLoginCredentials() {
        try {
            SharedPreferences  prefs  =  mContext.getSharedPreferences("TimeBankAuth",
Context.MODE_PRIVATE);
            SharedPreferences.Editor editor = prefs.edit();
            editor.remove("loginPasswordEncoded");
            editor.remove("credentialsSavedAt");
            editor.putBoolean("autoLoginEnabled", false);
            // 保留 loginEmail 用于自动填充
            editor.apply();
            android.util.Log.d("TimeBank",  "Login  credentials  cleared  (password
only)");
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "clearLoginCredentials error", e);
        }
    }

    // [v7.9.4] 设置是否启用自动登录
    @JavascriptInterface
    public void setAutoLoginEnabled(boolean enabled) {
        try {
            SharedPreferences  prefs  =  mContext.getSharedPreferences("TimeBankAuth",
Context.MODE_PRIVATE);
            prefs.edit().putBoolean("autoLoginEnabled", enabled).apply();
            android.util.Log.d("TimeBank", "Auto login enabled: " + enabled);
            // 如果禁用自动登录，同时清除密码
            if (!enabled) {
                clearLoginCredentials();
            }
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "setAutoLoginEnabled error", e);
        }
    }

    // ========== [v7.11.2] 设置持久化接口（解决 WebView localStorage 不可靠问题）
==========

    /**
     * 保存屏幕时间设置到 SharedPreferences（本地持久化）
     * @param settingsJson JSON 字符串
     */
    @JavascriptInterface
    public void saveScreenTimeSettingsNative(String settingsJson) {
        try {
            SharedPreferences prefs = mContext.getSharedPreferences("TimeBankSettings",
Context.MODE_PRIVATE);
            prefs.edit().putString("screenTimeSettings", settingsJson).commit(); // 使
用 commit 确保同步写入
            android.util.Log.d("TimeBank",  "[Native]  ScreenTime  settings  saved,
length=" + settingsJson.length());
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "saveScreenTimeSettingsNative error", e);
        }
    }

    /**
     * 读取屏幕时间设置
     * @return JSON 字符串，如果不存在返回空字符串
     */
    @JavascriptInterface
    public String getScreenTimeSettingsNative() {
        try {
            SharedPreferences prefs = mContext.getSharedPreferences("TimeBankSettings",
Context.MODE_PRIVATE);
            String result = prefs.getString("screenTimeSettings", "");
            android.util.Log.d("TimeBank",  "[Native]  ScreenTime  settings  loaded,
length=" + result.length());
            return result;
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "getScreenTimeSettingsNative error", e);
            return "";
        }
    }

    /**
     * 保存睡眠时间设置到 SharedPreferences（本地持久化）
     * @param settingsJson JSON 字符串
     */
    @JavascriptInterface
    public void saveSleepSettingsNative(String settingsJson) {
        try {
            SharedPreferences prefs = mContext.getSharedPreferences("TimeBankSettings",
Context.MODE_PRIVATE);
            prefs.edit().putString("sleepSettings", settingsJson).commit();
            android.util.Log.d("TimeBank", "[Native] Sleep settings saved, length=" +
settingsJson.length());
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "saveSleepSettingsNative error", e);
        }
    }

    /**
     * 读取睡眠时间设置
     * @return JSON 字符串，如果不存在返回空字符串
     */
    @JavascriptInterface
    public String getSleepSettingsNative() {
        try {
            SharedPreferences prefs = mContext.getSharedPreferences("TimeBankSettings",
Context.MODE_PRIVATE);
            String result = prefs.getString("sleepSettings", "");
            android.util.Log.d("TimeBank", "[Native] Sleep settings loaded, length=" +
result.length());
            return result;
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "getSleepSettingsNative error", e);
            return "";
        }
    }

    /**
     * 保存睡眠状态到 SharedPreferences
     * @param stateJson JSON 字符串
     */
    @JavascriptInterface
    public void saveSleepStateNative(String stateJson) {
        try {
            SharedPreferences prefs = mContext.getSharedPreferences("TimeBankSettings",
Context.MODE_PRIVATE);
            prefs.edit().putString("sleepState", stateJson).commit();
            android.util.Log.d("TimeBank", "[Native] Sleep state saved");
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "saveSleepStateNative error", e);
        }
    }

    /**
     * 读取睡眠状态
     * @return JSON 字符串
     */
    @JavascriptInterface
    public String getSleepStateNative() {
        try {
            SharedPreferences prefs = mContext.getSharedPreferences("TimeBankSettings",
Context.MODE_PRIVATE);
            return prefs.getString("sleepState", "");
        } catch (Exception e) {
            android.util.Log.e("TimeBank", "getSleepStateNative error", e);
            return "";
        }
    }
    /**
     * [v7.11.2] 原生日志输出（用于调试）
     * @param tag 日志标签
     * @param message 日志内容
     */
    @JavascriptInterface
    public void nativeLog(String tag, String message) {
        android.util.Log.d("TimeBank-" + tag, message);
    }
}
package com.jianglicheng.timebank;

import android.Manifest;
import android.app.DownloadManager;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.res.Configuration;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.provider.Settings;
import android.webkit.DownloadListener;
import android.webkit.URLUtil;
import android.webkit.ValueCallback;
import android.webkit.WebChromeClient;
import android.webkit.WebResourceRequest;
import android.webkit.WebResourceResponse;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.webkit.WebViewAssetLoader;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Base64;

public class MainActivity extends AppCompatActivity {

    private WebView myWebView;
    private ValueCallback<Uri[]> mUploadMessage;
    public static final int FILECHOOSER_RESULTCODE = 1;
    private WebViewAssetLoader assetLoader;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 1. 动态申请通知权限 (Android 13+)
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
            if                              (ContextCompat.checkSelfPermission(this,
Manifest.permission.POST_NOTIFICATIONS) != PackageManager.PERMISSION_GRANTED) {
                ActivityCompat.requestPermissions(this,                          new
String[]{Manifest.permission.POST_NOTIFICATIONS}, 101);
            }
        }

        // 2. 申请闹钟权限 (Android 12+)
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
            android.app.AlarmManager   alarmManager   =   (android.app.AlarmManager)
getSystemService(ALARM_SERVICE);
            if (!alarmManager.canScheduleExactAlarms()) {
                Intent                 intent                 =                 new
Intent(Settings.ACTION_REQUEST_SCHEDULE_EXACT_ALARM);
                startActivity(intent);
            }
        }

        myWebView = new WebView(this);
        setContentView(myWebView);


        WebSettings webSettings = myWebView.getSettings();
        webSettings.setJavaScriptEnabled(true);
        webSettings.setDomStorageEnabled(true);
        webSettings.setDatabaseEnabled(true);
        webSettings.setAllowFileAccess(true);

        // [v7.3.4] 设置 WebView 数据持久化路径，防止重启后登录状态丢失
        String     databasePath     =     getApplicationContext().getDir("webviewdb",
MODE_PRIVATE).getPath();
        webSettings.setDatabasePath(databasePath);
        // 设置缓存模式为优先使用缓存
        webSettings.setCacheMode(WebSettings.LOAD_DEFAULT);

        // 3. 暗色模式适配：强制 WebView 跟随系统
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
            int    nightModeFlags    =    getResources().getConfiguration().uiMode    &
Configuration.UI_MODE_NIGHT_MASK;
```

```
            if (nightModeFlags == Configuration.UI_MODE_NIGHT_YES) {
                webSettings.setForceDark(WebSettings.FORCE_DARK_ON);
            } else {
                webSettings.setForceDark(WebSettings.FORCE_DARK_OFF);
            }
        }

        // 4. 注入 JS 接口
        myWebView.addJavascriptInterface(new WebAppInterface(this), "Android");

        // [v7.9.9] 监听系统导航栏高度变化（适配三键导航栏）
        ViewCompat.setOnApplyWindowInsetsListener(myWebView, (v, insets) -> {
            Insets                              navInsets                              =
insets.getInsets(WindowInsetsCompat.Type.navigationBars());
            int bottom = navInsets != null ? navInsets.bottom : 0;
            myWebView.post(() -> myWebView.evaluateJavascript(
                "window.__setAndroidNavBarHeight && window.__setAndroidNavBarHeight(" +
bottom + ");",
                null
            ));
            return insets;
        });
        ViewCompat.requestApplyInsets(myWebView);

        // 5. 使用 WebViewAssetLoader 将本地资源映射到虚拟 HTTPS 域名
        // 这样 CloudBase SDK 才能正确识别域名
        assetLoader = new WebViewAssetLoader.Builder()
                .setDomain("timebank.local")  // 虚拟域名
                .addPathHandler("/assets/",                                        new
WebViewAssetLoader.AssetsPathHandler(this))
                .build();

        myWebView.setWebViewClient(new WebViewClient() {
            @Override
            public    WebResourceResponse    shouldInterceptRequest(WebView    view,
WebResourceRequest request) {
                return assetLoader.shouldInterceptRequest(request.getUrl());
            }
        });

        // 5. 文件选择支持 (导入/导出数据)
        myWebView.setWebChromeClient(new WebChromeClient() {
            @Override
            public  boolean  onShowFileChooser(WebView  webView,  ValueCallback<Uri[]>
filePathCallback, FileChooserParams fileChooserParams) {
                if (mUploadMessage != null) {
                    mUploadMessage.onReceiveValue(null);
                }
                mUploadMessage = filePathCallback;
                Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
                intent.addCategory(Intent.CATEGORY_OPENABLE);
                intent.setType("*/*");
                startActivityForResult(Intent.createChooser(intent, "选择备份文件"),
FILECHOOSER_RESULTCODE);
                return true;
            }
        });

        // 6. 下载支持 (处理 blob: URL 和普通下载)
        myWebView.setDownloadListener(new DownloadListener() {
            @Override
            public void onDownloadStart(String url, String userAgent, String
contentDisposition, String mimetype, long contentLength) {
                if (url.startsWith("blob:")) {
                    // blob URL 需要通过 JS 获取内容
                    myWebView.evaluateJavascript(
                        "(function() {" +
                        "  var xhr = new XMLHttpRequest();" +
                        "  xhr.open('GET', '" + url + "', true);" +
                        "  xhr.responseType = 'blob';" +
                        "  xhr.onload = function() {" +
                        "    var reader = new FileReader();" +
                        "    reader.onloadend = function() {" +
                        "                  Android.saveFile(reader.result, '" +
URLUtil.guessFileName(url, contentDisposition, mimetype) + "');" +
                        "    };" +
                        "    reader.readAsDataURL(xhr.response);" +
                        "  };" +
                        "  xhr.send();" +
                        "})();", null);
                } else {
                    // 普通 URL 使用系统下载管理器
                    DownloadManager.Request request = new
DownloadManager.Request(Uri.parse(url));
                    request.setMimeType(mimetype);
                    request.addRequestHeader("User-Agent", userAgent);
                    request.setDescription("正在下载文件...");
                    request.setTitle(URLUtil.guessFileName(url, contentDisposition,
mimetype));

request.setNotificationVisibility(DownloadManager.Request.VISIBILITY_VISIBLE_NOTIFY
_COMPLETED);

request.setDestinationInExternalPublicDir(Environment.DIRECTORY_DOWNLOADS,
URLUtil.guessFileName(url, contentDisposition, mimetype));
                    DownloadManager dm = (DownloadManager)
getSystemService(Context.DOWNLOAD_SERVICE);
                    dm.enqueue(request);
                    Toast.makeText(getApplicationContext(),  " 文 件 开 始 下 载 ...",
Toast.LENGTH_SHORT).show();
                }
            }
        });

        // 加载网页 - 使用虚拟 HTTPS 域名
        myWebView.loadUrl("https://timebank.local/assets/www/index.html");
    }

    @Override
    public void onBackPressed() {
        if (myWebView.canGoBack()) {
            myWebView.goBack();
        } else {
            super.onBackPressed();
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
        super.onActivityResult(requestCode, resultCode, intent);
        if (requestCode == FILECHOOSER_RESULTCODE) {
            if (mUploadMessage == null) return;
            Uri[] results = null;
            if (resultCode == AppCompatActivity.RESULT_OK && intent != null) {
                String dataString = intent.getDataString();
                if (dataString != null) {
                    results = new Uri[]{Uri.parse(dataString)};
                }
            }
            mUploadMessage.onReceiveValue(results);
            mUploadMessage = null;
        }
    }
}
```