## **کلاس MainActivity**

### این کد مربوط به **فعالیت اصلی** برنامه‌ هست که شامل نمایش زمان استفاده از اپلیکیشن‌ها، دریافت زمان خواب و بیداری کاربر، و نمایش نمودار و اطلاعات مرتبط می‌باشد.

### در ادامه جزئیات بخش‌های مختلف و کلاس ها رو می تونید ببینید.

متغیرهای زیر در کلاس تعریف شده‌اند:

* totalUsageTextView: برای نمایش مجموع زمان استفاده از اپلیکیشن‌ها.
* socialAppChart: نمودار دایره‌ای برای نمایش مدت زمان استفاده از اپلیکیشن‌ شبکه های اجتماعی.
* sleepWakeTextView: برای نمایش زمان خواب و بیداری کاربر.
* socialAppsList: یک لیست ویو برای نمایش جزئیات اپلیکیشن‌های شبکه های اجتماعی.
* ثابت‌های PREFS\_NAME و KEY\_SLEEP\_WAKE\_INFO برای مدیریت ذخیره‌ داده ها.

### **2. متد onCreate**

داخل این متد وظایف زیر را انجام می‌دهد:

* **مقداردهی ویجت‌ها:** شامل TextViewها، نمودارها و دکمه‌ها.
* **بررسی مجوز دسترسی به آمار استفاده از برنامه ها:** از طریق hasUsageStatsPermission و در صورت عدم وجود دسترسی به مجوز، درخواست آن با requestUsageStatsPermission.
* **فراخوانی متدهای اصلی:** شامل initializeAppFeatures برای مقداردهی ویژگی‌های برنامه و getWakeSleepTime برای مدیریت اطلاعات خواب و بیداری.

### **3. متد initializeAppFeatures**

وظیفه این متد، مقداردهی ویژگی‌های اصلی برنامه است:

* **نمایش زمان استفاده کلی:** با استفاده از UsageTracker.getTotalUsage و فرمت زمان با formatTimeLong.
* **نمایش نمودار اپلیکیشن‌های اجتماعی:** از طریق populatePieChart با استفاده از داده‌های آمار اپلیکیشن‌های اجتماعی که از UsageTracker.getSocialAppUsage دریافت می‌شود.

### **4. متد populatePieChart**

این متد، داده‌های اپلیکیشن‌های اجتماعی را در نمودار دایره‌ای نمایش می‌دهد:

* دریافت داده‌ها و تبدیل آن‌ها به لیستی از ورودی‌های PieEntry.
* افزودن رنگ‌ها به نمودار بر اساس نوع اپلیکیشن (مانند واتساپ، اینستاگرام، تلگرام).
* تنظیم ویژگی‌های ظاهری نمودار از جمله نمایش درصدها، حذف توضیحات اضافی، و بازخوانی اطلاعات.

### **5. متد displaySocialApps**

این متد، لیست اپلیکیشن‌های اجتماعی و زمان استفاده از آن‌ها را نمایش می‌دهد:

* **پاک‌کردن اطلاعات قبلی:** حذف تمام ویوهای موجود در socialAppsList.
* **ایجاد ویو جدید برای هر اپلیکیشن:** شامل نام، آیکون و زمان استفاده.
* **مدیریت کلیک:** انتقال به صفحه جزئیات اپلیکیشن از طریق openAppDetailsActivity.

### **6. مدیریت زمان خواب و بیداری (getWakeSleepTime)**

این متد اطلاعات مربوط به زمان آخرین قفل و بازگشایی دستگاه را مدیریت می‌کند:

* **دریافت اطلاعات قفل/بازگشایی:** با استفاده از LockUnlockTimeFetcher.
* **فرمت زمان‌ها:** تبدیل به فرمت خوانا (HH:mm:ss) با استفاده از SimpleDateFormat.
* **بررسی اختلاف زمان:** بررسی اینکه آیا زمان بازگشایی حداقل دو ساعت پس از زمان قفل است یا خیر.
* **ذخیره و بارگذاری اطلاعات:** در SharedPreferences برای ذخیره یا بازیابی اطلاعات.

### **7. متدهای ذخیره و بارگذاری اطلاعات (SharedPreferences)**

دو متد برای مدیریت اطلاعات خواب و بیداری استفاده شده‌اند:

* **saveSleepWakeInfo:** ذخیره اطلاعات در SharedPreferences.
* **loadSleepWakeInfo:** بازیابی اطلاعات ذخیره‌شده.

### **8. مدیریت مجوزها**

دو متد زیر برای بررسی و درخواست مجوز دسترسی به آمار استفاده استفاده می‌شوند:

* **hasUsageStatsPermission:** بررسی اینکه آیا مجوز استفاده از آمار دستگاه داده شده است یا خیر.
* **requestUsageStatsPermission:** درخواست مجوز از کاربر در صورت عدم وجود.

### **9. فرمت زمان**

دو متد برای فرمت کردن زمان استفاده شده‌اند:

* **formatTime:** فرمت زمان به ساعت، دقیقه و ثانیه.
* **formatTimeLong:** فرمت زمان به روز، ساعت، دقیقه و ثانیه.

### **10. اتصال به سایر اکتیویتی‌ها**

متد openAppDetailsActivity اطلاعات مربوط به اپلیکیشن انتخاب‌شده را به صفحه جزئیات ارسال می‌کند.

## **کلاس AppDetailsActivity**

## این کلاس وظیفه نمایش جزئیات مربوط به یک اپلیکیشن خاص و نمودار استفاده آن در ۷ روز گذشته را دارد.

### **ویژگی‌های اصلی**

#### **۱. نمایش جزئیات اپلیکیشن**

* **آیکون و نام اپلیکیشن:**
  + آیکون و نام اپلیکیشن با استفاده از packageName که از طریق Intent ارسال شده، دریافت می‌شوند.
  + این اطلاعات در رابط کاربری در ویوهای مربوطه (appIconView و appNameTextView) نمایش داده می‌شوند.
* **مصرف روزانه و کل:**
  + از کلاس UsageTracker برای دریافت میزان مصرف روزانه و کل اپلیکیشن استفاده شده است.
  + اطلاعات مصرف به صورت فرمت مناسب نمایش داده می‌شود و از رشته‌های منابع (R.string.today\_usage و R.string.total\_usage) استفاده شده است.

#### **۲. نمایش نمودار مصرف هفتگی**

* **داده‌ها:**
  + با استفاده از متد UsageTracker.getAppUsageForLast7Days، میزان مصرف اپلیکیشن در ۷ روز گذشته دریافت می‌شود.
  + داده‌ها به صورت Entry در آرایه‌ای ذخیره می‌شوند تا در نمودار استفاده شوند.
* **پیکربندی نمودار:**
  + از کتابخانه MPAndroidChart برای رسم نمودار استفاده شده است.
  + رنگ‌های مورد نیاز از تم فعلی برنامه (accentColor و primaryColor) دریافت می‌شوند.
  + نمودار خطی با استفاده از LineDataSet و داده‌های جمع‌آوری شده رسم می‌شود.
  + محور X به گونه‌ای تنظیم شده که مقادیر به ترتیب روزهای هفته (از ۱ تا ۷) نمایش داده شوند.

### **متدهای مهم در کد**

#### **۱. متد displayAppDetails**

* وظیفه نمایش جزئیات اپلیکیشن (آیکون، نام، مصرف روزانه و کل) را برعهده دارد.
* با استفاده از PackageManager آیکون و نام اپلیکیشن دریافت می‌شوند.
* استفاده روزانه و کل با فرمت مناسب نمایش داده می‌شود.

#### **۲. متد displayUsageChart**

* داده‌های استفاده اپلیکیشن در ۷ روز گذشته دربافت و در نمودار خطی نمایش داده می‌شوند.

#### **۳. متد formatTime**

* وظیفه تبدیل میلی‌ثانیه به فرمت ساعت، دقیقه و ثانیه را برعهده دارد. این متد در چندین جای برنامه استفاده می‌شود.

#### **۴. متد onSupportNavigateUp**

* دکمه بازگشت در Toolbar را فعال می‌کند و کاربر را به فعالیت قبلی بازمی‌گرداند.

## **کلاس AppUtils**

این کلاس شامل دو متد عمومی برای کار با اپلیکیشن‌ها و دریافت رنگ از تم در اندروید است.

### **متدها**

#### **۱. متد getAppNameFromPackage**

* **هدف:**
  + این متد برای دریافت نام اپلیکیشن از طریق **نام پکیج (Package Name)** طراحی شده است.
* **عملکرد:**
  + با استفاده از **PackageManager**، اطلاعات مربوط به اپلیکیشن (ApplicationInfo) از طریق نام پکیج دریافت می‌شود.
  + اگر اطلاعات اپلیکیشن موجود باشد، نام آن با استفاده از **getApplicationLabel** بازگردانده می‌شود.
  + در صورتی که اپلیکیشن مورد نظر پیدا نشود (برای مثال، نام پکیج اشتباه باشد)، استثنای **NameNotFoundException** مدیریت شده و به جای نام اپلیکیشن، نام پکیج بازگردانده می‌شود.

#### **۲. متد getColorFromTheme**

* **هدف:**
  + دریافت یک رنگ خاص از تم جاری اپلیکیشن.
* **عملکرد:**
  + از کلاس **TypedValue** برای دریافت مقادیر مربوط به خصوصیات رنگ در تم استفاده می‌شود.
  + متد **resolveAttribute**، مقدار داده شده را از تم جاری دریافت می‌کند.
* **کاربرد:**
  + برای مثال می‌توان رنگی مثل **colorPrimary** یا **colorAccent** را از تم دریافت کرد.

## **کلاس LockUnlockTimeFetcher**

این کلاس برای بازیابی آخرین زمان‌های مربوط به قفل و باز کردن صفحه‌نمایش دستگاه طراحی شده است.

این کار با استفاده از **UsageStatsManager** و تحلیل رویدادهای دستگاه انجام می‌شود.

### **متدها**

#### **۱. getLastUnlockTime**

* **هدف:**
  + بازگرداندن زمان آخرین باز کردن قفل صفحه (Screen Unlock).
* **عملکرد:**
  + این متد به صورت داخلی از متد عمومی **getLastEventTime** استفاده می‌کند و نوع رویداد (UsageEvents.Event.SCREEN\_INTERACTIVE) را برای جستجوی رویدادهای مربوط به باز کردن صفحه ارسال می‌کند.
* **نتیجه:**
  + زمان (به میلی‌ثانیه) مربوط به آخرین رویداد باز کردن قفل صفحه.

#### **۲. getLastLockTime**

* **هدف:**
  + بازگرداندن زمان آخرین قفل شدن صفحه (Screen Lock).
* **عملکرد:**
  + مشابه متد قبلی، با این تفاوت که اینجا نوع رویداد برای قفل شدن صفحه (UsageEvents.Event.SCREEN\_NON\_INTERACTIVE) ارسال می‌شود.
* **نتیجه:**
  + زمان (به میلی‌ثانیه) مربوط به آخرین رویداد قفل شدن صفحه.

#### **۳. getLastEventTime**

* **هدف:**
  + متد عمومی برای جستجوی آخرین زمان وقوع یک رویداد مشخص.
* **عملکرد:**
  + با استفاده از **UsageStatsManager.queryEvents**، تمام رویدادها در بازه زمانی تعیین‌شده (یک روز گذشته) دریافت می‌شوند.
  + رویدادها با حلقه بررسی می‌شوند و اگر نوع رویداد با مقدار **eventType** یکسان باشد، زمان آن ذخیره می‌شود.
  + پس از پایان حلقه، زمان آخرین رویداد بازگردانده می‌شود.
* **نتیجه:**
  + زمان آخرین وقوع رویداد (به میلی‌ثانیه) یا صفر اگر رویدادی پیدا نشود.

## **کلاس OtherAppsActivity**

این کلاس برای نمایش لیست سایر اپلیکیشن‌ها و میزان زمان استفاده از آن‌ها طراحی شده است.

این صفحه از طریق تحلیل داده‌های **UsageTracker** اطلاعات اپلیکیشن‌های دیگر را جمع‌آوری کرده و به کاربر نمایش می‌دهد.

#### **1. displayOtherApps**

* **هدف:**
  1. نمایش اطلاعات اپلیکیشن‌های دیگر (نام، آیکون، زمان استفاده).
* **عملکرد:**
  1. متد **UsageTracker.getOtherAppUsage** داده‌های مربوط به استفاده از اپلیکیشن‌ها را بازیابی می‌کند.
  2. با استفاده از حلقه، اطلاعات هر اپلیکیشن (نام و آیکون) از طریق **PackageManager** گرفته می‌شود.
  3. برای هر اپلیکیشن، یک ویو سفارشی با استفاده از **social\_app\_item.xml** ساخته می‌شود که شامل موارد زیر است:
     + **نام اپلیکیشن**
     + **زمان استفاده**
     + **آیکون اپلیکیشن**
     + **آیکون بعدی برای دسترسی به جزئیات اپلیکیشن**
  4. هر آیتم به **LinearLayout** اضافه می‌شود.

#### **2. openAppDetailsActivity**

* **هدف:**
  + باز کردن صفحه جزئیات اپلیکیشن (Activity دیگر) برای یک اپلیکیشن خاص.
* **پارامترها:**
  + **packageName**: نام پکیج اپلیکیشن.
  + **appName**: نام اپلیکیشن.
* **عملکرد:**
  + با استفاده از **Intent**، اطلاعات اپلیکیشن (نام و پکیج) به کلاس **AppDetailsActivity** ارسال می‌شود.
  + صفحه **AppDetailsActivity** نمایش داده می‌شود.

#### **3. formatTime**

* **هدف:**
  + تبدیل زمان استفاده به فرمت خوانا (ساعت:دقیقه:ثانیه).
* **عملکرد:**
  + زمان به میلی‌ثانیه گرفته می‌شود و به مقادیر ساعت، دقیقه و ثانیه تقسیم می‌شود.

## **کلاس UsageTracker**

این کلاس برای تحلیل داده‌های استفاده از اپلیکیشن‌ها بر اساس اطلاعاتی که از **UsageStatsManager** جمع‌آوری می‌کند می باشد.

این کلاس روش‌های مختلفی برای بازیابی زمان استفاده از اپلیکیشن‌ها (اعم از شبکه‌های اجتماعی، سایر اپلیکیشن‌ها، و همچنین کل زمان استفاده) فراهم می‌کند.

### **عملکرد متدها**

#### **۱. getTotalAppUsage**

* **هدف:**
  + جمع‌آوری زمان استفاده از تمامی اپلیکیشن‌ها در ۲۴ ساعت گذشته.
* **عملکرد:**
  + از متد **queryUsageStats** برای دریافت داده‌های استفاده از تمامی اپلیکیشن‌ها استفاده می‌کند.
  + زمان کل استفاده از هر اپلیکیشن (مقدار **getTotalTimeInForeground**) را در یک **Map** ذخیره می‌کند.

#### **۲. getSocialAppUsage**

* **هدف:**
  + دریافت زمان استفاده از اپلیکیشن‌های شبکه‌های اجتماعی.
* **عملکرد:**
  + با استفاده از آرایه **SOCIAL\_APPS** (که شامل پکیج‌نیم‌های شبکه‌های اجتماعی است) داده‌های استفاده را فیلتر کرده و تنها موارد مرتبط با این اپلیکیشن‌ها را بازمی‌گرداند.

#### **۳. getOtherAppUsage**

* **هدف:**
  + دریافت زمان استفاده از اپلیکیشن‌هایی که در دسته شبکه‌های اجتماعی قرار ندارند.
* **عملکرد:**
  + تمامی اپلیکیشن‌ها را از **getTotalAppUsage** دریافت می‌کند.
  + با استفاده از متد **isSocialApp**، اپلیکیشن‌هایی که شبکه اجتماعی نیستند فیلتر شده و در یک **Map** جداگانه ذخیره می‌شوند.

#### **۴. getTodayUsageForApp**

* **هدف:**
  + دریافت زمان استفاده از یک اپلیکیشن خاص در طول روز جاری.
* **عملکرد:**
  + با تنظیم ساعت شروع (نیمه‌شب) و پایان (زمان جاری)، اطلاعات مربوط به آن بازه زمانی را با **queryUsageStats** بازیابی می‌کند.
  + با بررسی **PackageName**، مقدار زمان استفاده از اپلیکیشن موردنظر محاسبه و بازگردانده می‌شود.

#### **۵. getTotalUsageForApp**

* **هدف:**
  + دریافت کل زمان استفاده از یک اپلیکیشن خاص در یک سال گذشته.
* **عملکرد:**
  + زمان شروع و پایان یک سال گذشته را تنظیم می‌کند.
  + داده‌های مربوط به آن بازه را بازیابی کرده و زمان استفاده از اپلیکیشن موردنظر را از مجموع **getTotalTimeInForeground** محاسبه می‌کند.

#### **۶. getAppUsageForLast7Days**

* **هدف:**
  + دریافت زمان استفاده از یک اپلیکیشن خاص در ۷ روز گذشته.
* **عملکرد:**
  + به صورت حلقه برای هر روز بازه زمانی ۲۴ ساعت مشخص کرده و داده‌های استفاده را بازیابی می‌کند.
  + زمان استفاده هر روز را با **Day X** به‌عنوان کلید ذخیره می‌کند.

#### **۷. getTotalUsage**

* **هدف:**
  + محاسبه کل زمان استفاده از تمامی اپلیکیشن‌ها در ۲۴ ساعت گذشته.
* **عملکرد:**
  + با جمع‌کردن تمامی مقادیر موجود در **Map** بازگردانده‌شده توسط **getTotalAppUsage**، کل زمان استفاده محاسبه می‌شود.

#### **۸. isSocialApp**

* **هدف:**
  + بررسی اینکه یک اپلیکیشن در دسته شبکه‌های اجتماعی قرار دارد یا خیر.
* **عملکرد:**
  + نام پکیج اپلیکیشن را با مقادیر موجود در آرایه **SOCIAL\_APPS** مقایسه کرده و نتیجه **true/false** بازمی‌گرداند.