프로젝트 보고서

**프로젝트명 : One Touch Task Manager**

팀 명 : AndrorMetaCon

팀 원 :

**20063450 홍길동**  **20063478 성춘향**

[**1. 프로젝트 개요** 4](#_Toc311796440)

[1.1 프로젝트 명 4](#_Toc311796441)

[1.2 개발 동기 4](#_Toc311796442)

[1.3 개발 목적 4](#_Toc311796443)

[1.4 향후 발전 방향 4](#_Toc311796444)

[1.5 개발환경 및 기기 5](#_Toc311796445)

[**2. 배경 지식** 5](#_Toc311796446)

[2.1 Android 5](#_Toc311796447)

[2.2 Android의 특징 6](#_Toc311796448)

[2.3 Android 라이브러리 계층 7](#_Toc311796449)

[**3. 유사 서비스** 8](#_Toc311796450)

[3.1 Android Assistant 8](#_Toc311796451)

[3.1.1 분석 11](#_Toc311796452)

[3.2 Advanced Task Killer 11](#_Toc311796453)

[3.2.1 분석 12](#_Toc311796454)

[**4. 개발 내용** 13](#_Toc311796455)

[4.1 메인 화면 13](#_Toc311796456)

[4.2 기본 화면 13](#_Toc311796457)

[4.3 User App Kill 14](#_Toc311796458)

[4.4 Memory overseer 14](#_Toc311796459)

[4.5 Cache Clear 15](#_Toc311796460)

[4.6 Ignore List 15](#_Toc311796461)

[4.7 Done 16](#_Toc311796462)

[**5.개발 일정** 17](#_Toc311796463)

[**6. 체크포인트 리스트(Check Point List)** 18](#_Toc311796464)

[6.1 User App Kill 18](#_Toc311796465)

[6.2 Memory Overseer 18](#_Toc311796466)

[6.3 Cache Clear 18](#_Toc311796467)

[6.4 Ignore List 18](#_Toc311796468)

[6.5 Done(Self-Kill) 19](#_Toc311796469)

[**7. 멘토링 보고서** 19](#_Toc311796470)

[**8. 자료 및 참고 문헌 출처** 20](#_Toc311796471)

[[목차 1.5 개발환경 및 기기] 20](#_Toc311796472)

[[목차 2.1 Android] 20](#_Toc311796473)

[[목차 2.2 Android의 특징] 20](#_Toc311796474)

[[목차 2.3 Android 라이브러리 계층] 20](#_Toc311796475)

[[목차 3. 유사어플] 21](#_Toc311796476)

[[목차 4. 개발내용] 21](#_Toc311796477)

**1. 프로젝트 개요**

## **1.1 프로젝트 명**

One Touch Task Manager

## **1.2 개발 동기**

Android 2.1(이하 이클레어) 이전의 Android 운영체제에서는 OS가 효율적으로 메모리 관리를 하지 못했기 때문에 다소 많은 단점들이 존재하였지만 필수적으로 Task Manager가 필요했었다. 하지만 Android 2.2(이하 프로요) 이후의 Android 운영체제 에서는 OS가 자체적으로 효율적인 메모리 관리를 할 수 있도록 개선되었고 자연스럽게 Task Manager의 단점들이 부각되기 시작하면서 “과연 Task Manager가 필요한가?” 라는 문제점이 제기 되기 시작하였다. 실제로 Task Manager는 많은 기능과 편리함을 제공하는 동시에 몇몇의 치명적이지는 않지만 사용자들이 직접적으로 체감 할 수 있는 문제점들이 있는데 이를 개선하고자 한다.

## **1.3 개발 목적**

Android 프로요 이후의 상황을 보면, Task Manager의 가장 큰 문제점인 배터리 소모, 많은 자원 점유, 강제적인 종료로 인한 Home UI Lack 현상, 실제로 확보되는 메모리량 등이 있는데, 이런 점을 개선하여, 위젯만을 백그라운드에 남겨두고, 위젯을 누르게 되면 앱을 Active시켜 지정된 앱을 Kill하고, 모든 앱의 Kill이 끝나면 스스로도 Kill하여 백그라운드에만 남겨두게 하여 배터리 소모, 자원 점유율을 내리는 등 단점들을 개선 하고자 한다.

## **1.4 향후 발전 방향**

Andoird OS가 발전함에 현재 Main-Stream OS인 프로요, 진저브레드의 메모리 관리 능력보다 더 향상 될 것은 자명한 일이며, 따라서 Task Manager의 입지는 점점 좁아지게 될 것이다. 이런 악조건 에서도 이런 종류의 앱이 살아남기 위해서는 SU(Super User)권한과 연계되는 쪽으로 발전되어야 할 것이다. 사실 국내에서 스마트폰이 유행하기 시작 한 것은 프로요 버전 이후이기 때문에 딱히 Task Manager가 필요하지 않았던 것이 사실이다. 하지만 Task Manager가 많은 관심을 끌었던 이유는 바로 국내 스마트폰에는 SU권한이 없으면 삭제 할 수 없는 제휴 또는 통신사 자체제작 번들웨어가 항시 메모리에 상주해 있기 때문이다. 국내 통신사에서는 개인 사용자가 SU권한을 얻는 것을 비 추천하며 실제 AS시에 불이익을 안겨주곤 있다. 하지만 Android는 엄연히 오픈 컨텐츠이다. 이것들을 연관시킬 수 있다면 이후에도 널리 쓰이지 않을까 생각해본다.

## **1.5 개발환경 및 기기**

OS : Microsoft Windows7 Home Premium, GPL v2(Android 2.3)

Language : Java SDK, Android NDK, C

개발기기 : Optimus MACH(Android 2.3 Ginger Bread)

|  |  |
| --- | --- |
| 프로세서 | CPU : MDM6600  GPU : Ti OMAP 3600 |
| RAM | 483 MB |
| 저장공간 | 내부 : 336 MB  외부 : SD card(최대 32GB) |
| 디스플레이 | 3.8 인치 TFT-LCD  480 x 800 픽셀 WVGA |

[그림 1.1] 개발기기 Optimus Mach

# **2. 배경 지식**

## **2.1 Android**

안드로이드(Android)는 휴대 전화를 비롯한 휴대용 장치를 위한 운영 체제와 미들웨어,사용자 인터페이스 그리고 표준 응용 프로그램(웹 브라우저, 이메일 클라이언트, 단문 메시지 서비스(SMS), 멀티미디어 메시지 서비스(MMS)등)을 포함하고 있는 소프트웨어 스택이다. 안드로이드는 개발자들이 자바 언어로 응용 프로그램을 작성할 수 있게 하였으며, 컴파일된 바이트코드를 구동할 수 있는 런타임 라이브러리를 제공한다. 또한 안드로이드 소프트웨어 개발 키트(SDK:Software Development Kit)를 통해 응용 프로그램을 개발하기 위해 필요한 각종 도구들과 응용 프로그램 프로그래밍 인터페이스(API)를 제공한다.

안드로이드는 리눅스 커널 위에서 동작하며, 다양한 안드로이드 시스템 구성 요소에서 사용되는 C/C++ 라이브러리들을 포함하고 있다. 안드로이드는 기존의 자바 가상 머신과는 다른 가상 머신인 달빅 가상 머신을 통해 자바로 작성된 응용 프로그램을 별도의 프로세스에서 실행하는 구조로 되어 있다.

2005년에 안드로이드 사를 구글에서 인수한 후 2007년 11월에 안드로이드 플랫폼을 휴대용 장치 운영 체제로서 무료 공개한다고 발표한 후 48개의 하드웨어, 소프트웨어, 통신 회사가 모여 만든 오픈 핸드셋 얼라이언스(Open Handset Aliance,OHA)에서 공개 표준을 위해 개발하고 있다. 구글은 안드로이드의 모든 소스 코드를 오픈 소스 라이선스인 아파치 v2 라이선스로 배포하고 있어 기업이나 사용자는 각자 안드로이드 프로그램을 독자적으로 개발을 해서 탑재할 수 있다. 또한 응용 프로그램을 사고 팔수 있는 구글 안드로이드 마켓을 제공하고 있으며, 이와 동시에 각 제조사 혹은 통신사별 응용 프로그램 마켓이 함께 운영되고 있다. 마켓에서는 유료 및 무료 응용 프로그램이 제공되고 있다.

## **2.2 Android의 특징**

안드로이드는 다른 스마트폰용 플랫폼과 특징이 매우 다른데 그 이유는 오픈 소스 소프트웨어기반에서 개발되었기 때문이다. 또한 구글이라는 광고 검색 시장 1위 기업이 만들어낸 것이라 구글의 생각이 그대로 반영된 제품이기도 하다 .

안드로이드의 구성 요소를 살펴보면 애플리케이션 Framework, Dalvik VirtualMachine, 통합 브라우저, 그래픽 최적화, SQLite, 미디어 지원, 그 외 하드웨어 의존적인 기술들과 유용한 개발 환경 지원이 있다.

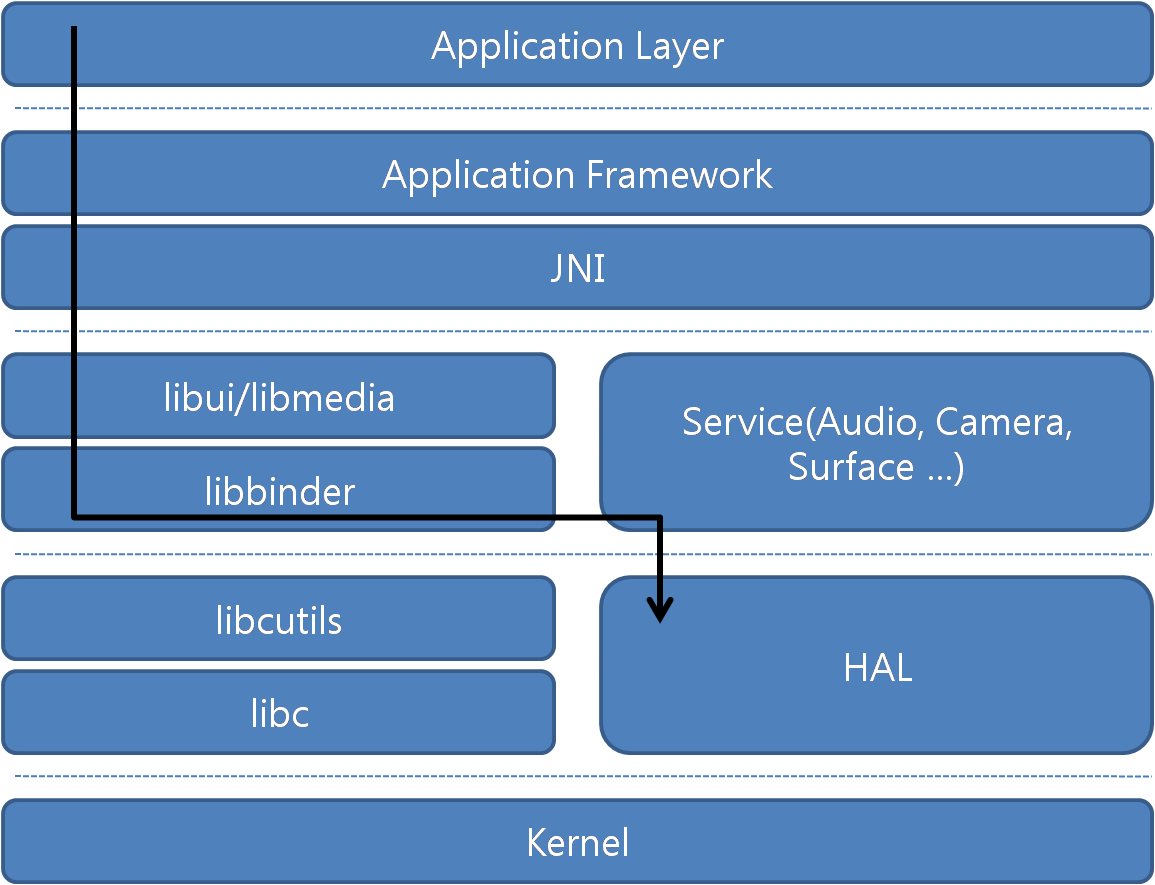
Android의 구조는 크게 안드로이드용으로 패치 된 리눅스 커널, C/C++로 작성된 라이브러리(Native라고도 표현함), 애플리케이션 Framework(Java API에 해당함), 애플리케이션 (Java로 작성된 실행 프로그램) 이렇게 4 부분으로 나뉜다.

안드로이드는 여러 가지 장점 때문에 리눅스 커널을 기반으로 하지만 시스템이 아닌 커널 부분만을 리눅스로 사용한다. 이유는 안드로이드의 목표가 바로 스마트 폰이기 때문이다. 현재 안드로이드 관련 리눅스 커널은 2.6.35 버전이다. 이 커널은 구글에서 공식 리눅스 커널 사이트에서 커널을 가져다가 안드로이드용으로 패치 한 것이다. 패치한 부분은 Binder, Alarm, Ashmem, Low memory-killer, debugger, logger컴포넌트 등이다. 안드로이드를 컴파일 할 때는 EABI(Embedded Application Binary Interface) 툴체인을 사용한다. 이전에는 일반적인 ARM GCC를 사용했지만, 최근(리눅스 커널 2.6.18 이후)에는 EABI 툴체인을 사용하도록 되어있다.

## **2.3 Android 라이브러리 계층**

라이브러리 계층은 실제로는 안드로이드 이전에 개발된 라이브러리 혹은 외부에서 기존 프로젝트에서 사용하던 라이브러리(주로 external 디렉터리), 안드로이드 Framework 부분(frameworks/base 디렉터리 중 C++로 작성된 것- JNI는 제외), 안드로이드의 HAL부분 이 세 가지로 나눌 수 있다.

HAL 부분은 frameworks/base/libs의 여러 디렉터리에 특성에 알맞게 분산되어 있거나, hardware 디렉터리 혹은 vendor디렉터리 등으로 분산되어 있다. 디렉터리로 HAL 계층을 정확하게 분리하지 못하는 이유도 여기에 있다고 불 수 있다. 기존의 리눅스 관련 라이브러리는 거의 C로 구성되어 있기 때문에 C++로 구현된 내용은 대부분 안드로이드의 핵심 코드다. 이런 이유로 frameworks 디렉터리 밑의 C++코드는 안드로이드 그 자체라고 생각하는 게 낫다. 여기서 C++라고 하는 부분은 문법은 C++를 쓰지만, 실제로는 안드로이드 전용 C++라고 생각하는 것이 더 맞다. 이 부분의 내용은 주로 Java와 인터페이스 하기 위해 존재, Surface Manager -UI Windows 제어를 담당, 2D/3D 그래픽 라이브러리를 포함, 멀티미디어 코덱, SQLie DB엔진, WWW엔진을 포함 과 같은 내용이다. 안드로이드 플랫폼의 핵심인 Surface Manager와 Media Framework 등의 루틴을 포함하고 있다. 왜 이런 루틴이 C++ 레벨로 있는지에 대한 것을 생각해 보면 간단한 얘기다. 고속으로 처리해야 하는 부분과 대용량 메모리를 사용하는 루틴은 Java로 처리하지 않고 라이브러리 형태로 C++레벨에서 처리하는 것이다. 이 계층을 다시 자세한 라이브러리 형태의 그림으로 그려본다면 다음과 같다.

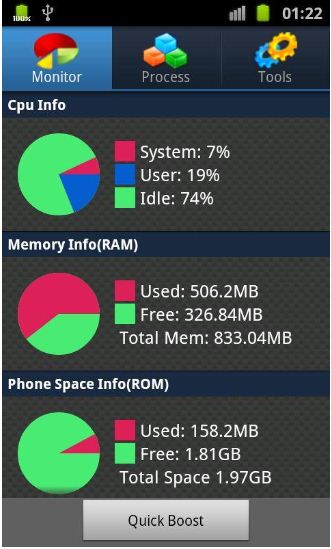


[그림 2.1] 안드로이드 계층

그림을 보면 안드로이드의 가장 중요한 부분이 나와 있다. 안드로이드의 핵심은 Binder를 이용한 소프트웨어 모듈화인데, Java에서 JNI를 통해서 C++의 진입점인 libui와 libmedia를 통하여 직접 하드웨어 등을 제어하는 게 아니라 Binder를 통해서 “각종 서비스(예: Surface flinger, audioflinger 등)로 접속해서 모든 하드웨어 제어와 여러 가지 핵심 기능을 처리하도록 설계되어 있다. 이렇게 구성한 이유는 여러 가지가 있겠지만 크게 UI에서 직접 하드웨어 혹은 여러 가지 기능을 처리할 경우 유닉스 특유의 blocking 방식의 함수를 처리할 방법이 없다는 것과 안드로이드의 구성상 안드로이드는 응용프로그램에 대해서 시스템으로 접근 할 수 있는 권한을 주지 않는 것이다.

# **3. 유사 서비스**

## **3.1 Android Assistant**



[그림 3.1] 모니터 탭 화면

CPU상태, 메모리상태(RAM), Phone Space Info(ROM), Sd\_card Space Info, 배터리 상태 확인



[그림 3.2] 자세히 탭 화면

여러 개의 실행중인 파일을 선택해서 종료 시킬 수 있다.



[그림 3.3] 도구 텝

다양한 기능들을 지원한다.



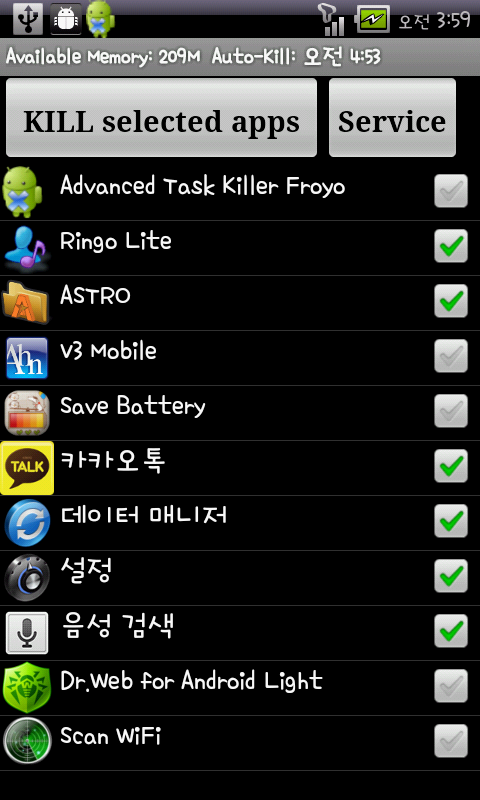
[그림 3.4] 위젯

### 3.1.1 분석

비교적 깔끔하고 한글로 되어 있어서 사용하는데 굉장히 편하고 위젯 기능도 지원한다.

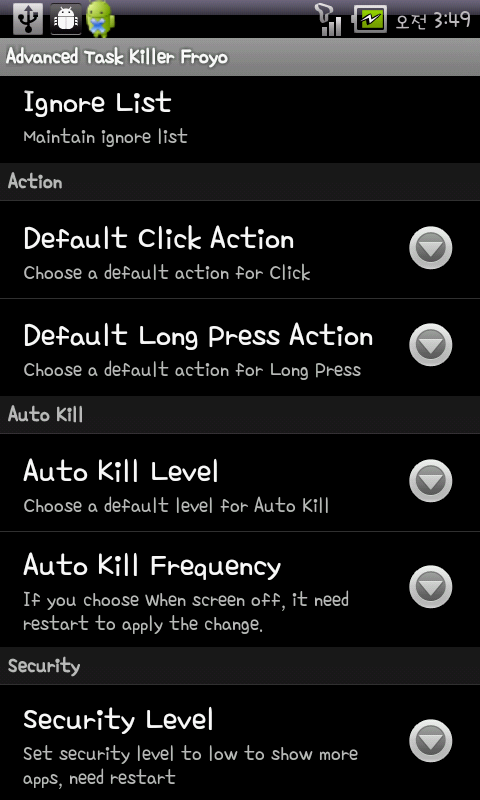
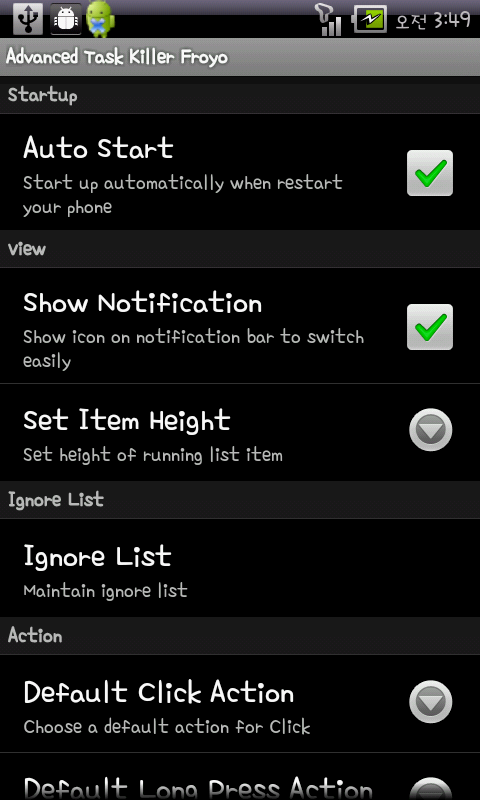
주기적으로 종료를 눌러 줘야 하고 종료해도 종료를 시켜도 기본 app은 다시 실행되며 강제종료로 인해 메모리가 전체적으로 한 순간에 초기화되어 시스템의 끊김 현상이 생기는 단점이 있다.

## **3.2 Advanced Task Killer**



[그림 3.5]실행화면

마찬가지로 여러 개의 실행중인 파일을 선택해서 종료 시킬 수 있다.



[그림 3.6]기능화면

다양한 기능들을 보여주고 있다.



[그림 3.7]위젯

위젯 아이콘을 한번 클릭하면 밑에 종료된 app 개수와 확보한 메모리량이 표시된다.

### 3.2.1 분석

현재 마켓에서 가장 많이 다운된 메모리 관리 app이다.(2011년 11월 30일 기준)

우선 영어로 표시되어 있어 사용하는데 조금의 불편함이 있다. 어플 실행 시 휴대폰속도가 느려지고 이 어플 자체가 메모리와 배터리를 많이 사용하는 단점이 있다.

# **4. 개발 내용**

## **4.1 메인 화면**



[그림 4.1 메인 화면]

## **4.2 기본 화면**



[그림 4.2 실행 시 기본 화면]

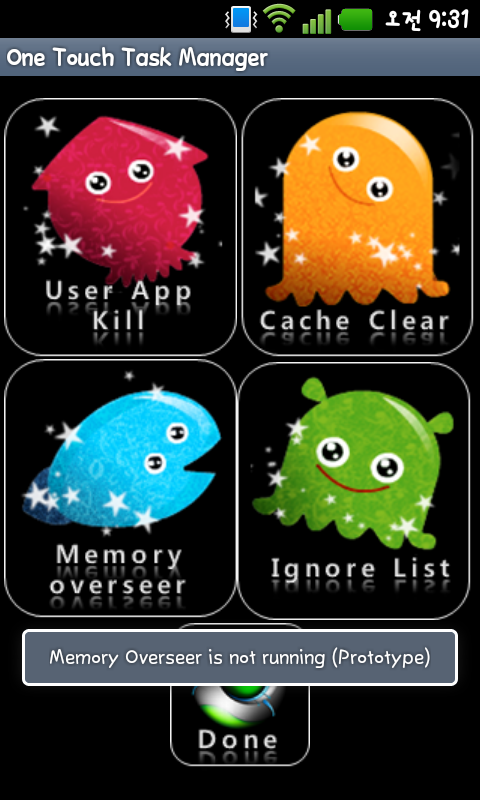
## **4.3 User App Kill**



[그림 4.3 User App Kill 실행 화면]

원 터치로 현재 실행중인 어플리케이션을 종료 시킨다.

## **4.4 Memory overseer**

****

[그림 4.4 Memory Overseer 실행 화면]

사용 가능한 RAM량과 실행중인 App 목록을 보여준다.

개별적인 프로세스 종료 기능이 있다.

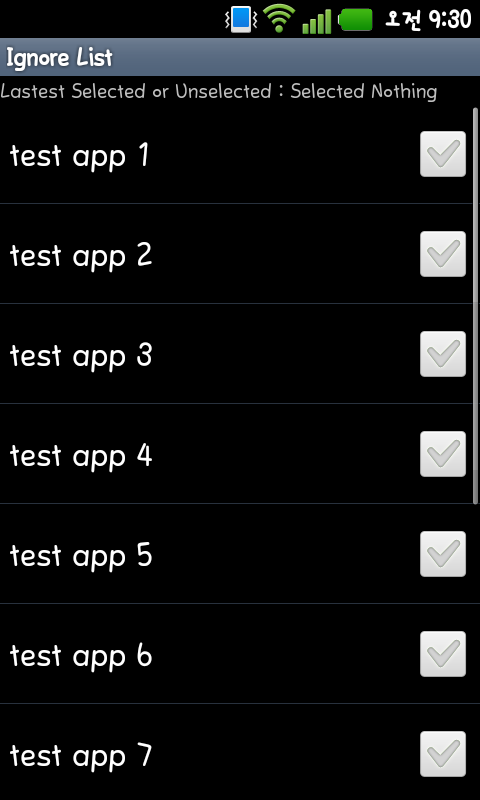
## **4.5 Cache Clear**

****

[그림 4.5 Cache Clear 실행화면]

내부 및 외부 메모리의 Cache 정리 해준다.

## **4.6 Ignore List**

****

[그림 4.6 Ignore List 실행화면]

필수 어플리케이션과User App Kill에서 제외할 목록 지정한다.

## **4.7 Done**



[그림 4.7 Done 실행화면]

App 사용이 끝난 후에 실행하면 자기 스스로 종료를 시키는 Self Kill 기능이다.

Manager가 잡아먹는 시스템 자원을 해결할 수 있다.

# **5.개발 일정**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **순서** | **기간**  **작업내용** | **11** | | | | **12** | | | | **1** | | | | **2** | | | | **3** | | | | **4** | | | |
| **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** |
| **1** | **프로젝트 일정 조정** | **A** | **A** | **A** | **A** | **A** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** | **프로젝트 물품 준비** |  |  | **A** | **A** | **A** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **3** | **전체 구성요소 설계** | **A** | **A** | **A** | **A** | **A** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **4** | **UI 설계** |  |  |  | **C** | **C** | **C** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **5** | **UI, 위젯 구현** |  |  |  |  | **C** | **C** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **6** | **User App Kill 설계** |  |  |  |  |  |  | **B** | **B** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **7** | **Memory Overseer 설계** |  |  |  |  |  |  | **C** | **C** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **8** | **User App Kill 구현** |  |  |  |  |  |  |  | **B** | **B** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **9** | **Memory Overseer 구현** |  |  |  |  |  |  |  | **C** | **C** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **10** | **User App Kill,**  **Memory Overseer**  **테스트 및 디버깅** |  |  |  |  |  |  |  |  | **A** | **A** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **11** | **Cache Clear 설계** |  |  |  |  |  |  |  |  |  |  | **C** | **C** |  |  |  |  |  |  |  |  |  |  |  |  |
| **12** | **Cache Clear 구현** |  |  |  |  |  |  |  |  |  |  |  | **C** | **C** |  |  |  |  |  |  |  |  |  |  |  |
| **13** | **Cache Clear**  **테스트 및 디버깅** |  |  |  |  |  |  |  |  |  |  |  |  |  | **A** | **A** |  |  |  |  |  |  |  |  |  |
| **14** | **Ignore List**  **설계 및 구현** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **B** | **B** |  |  |  |  |  |  |  |
| **15** | **Done 기능 설계 및 구현** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **C** | **C** |  |  |  |  |  |  |  |
| **16** | **Ignore List, Done**  **테스트 및 디버깅** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A** | **A** | **A** |  |  |  |  |  |
| **17** | **문서화 작업** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **B** |  |  |  |  |
| **18** | **1차 버그 수정 업데이트** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A** | **A** |  |  |  |
| **19** | **문서화 작업** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **C** |  |  |
| **20** | **2차 버그 수정 업데이트** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A** | **A** | **A** |
| **21** | **최종 발표 준비 및 발표** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A** | **A** |

**A : 공통 B : 김진열 C : 권준오**

# **6. 체크포인트 리스트(Check Point List)**

## **6.1 User App Kill**

System App과 User App을 구분 지어 둘 것이다. 이는 App 목록이 많은 점과 System App중에는 항시 유지되어야 하는 목록들이 많다는 점이 이와 같이 구분 하도록 결정하는데 기여를 했다.

## **6.2 Memory Overseer**

사용 가능한 RAM량과 실행중인 App 목록을 보여준다. 전체 메모리 대비 사용 메모리를 비교하여 일정 비율마다 표시하는 색깔을 다르게 보여준다. 실행중인 App 목록에서는 각 개별 App 종료 기능을 둔다.

## **6.3 Cache Clear**

Cache의 삭제는 RAM 관리보다 좀 더 Task Manager의 장점을 부각시켜준다. Android OS는 순차적으로 오래된 Cache 기록부터 삭제를 한다는 점에서 RAM 관리과 동일한데, 문제는 Cache의 유출이 개인정보 유출과 직접적으로 관련이 되기 때문이다. OS가 직접 삭제하기를 기다리기 보단, 사용자가 주기적으로 먼저 삭제 해준다면, Cache 관리도 되며, 개인정보의 관리 또한 용이 해진다.

## **6.4 Ignore List**

OS구동에 필수적인 System App, 데이터, 문자, 통화 사용량, 또는 사용자가 지정해두어 항시 가동 되어야 하는 App을 Task Manager가 건드리지 않도록 하는 Ignore 기능이다. 이 기능이 없을 경우, 일시에 모든 App을 Kill하게 되어 System App과 일부 User App이 한 번에 메모리에 로드 되게 되는 경우, 순간적으로 엄청난 Lack 현상을 일으키게 된다. 이를 미연에 방지 하고자 Ignore에 등록 해두어야 할 것이다.

## **6.5 Done(Self-Kill)**

Done 기능은 단어 뜻 그대로 모두 다 수행을 하였으며, 이제 이 Task Manager의 고질적인 문제인 Manager가 잡아먹는 시스템 자원을 조금이나마 해결하고자, 모든 메모리, Cache 관리를 끝내고 Done을 누를 경우, manager 스스로도 Kill하여 메모리에 여유를 두는 Self-Kill 기능이다.

# **8. 자료 및 참고 문헌 출처**

## **[목차 1.5 개발환경 및 기기]**

[그림 1.1] Optimus Mach

<http://www.lgmobile.co.kr/index.jsp> (LG Mobile)

## **[목차 2.1 Android]**

<http://ko.wikipedia.org/wiki/%EA%B5%AC%EA%B8%80_%EC%95%88%EB%93%9C%EB%A1%9C%EC%9D%B4%EB%93%9C> (Google Wiki백과)

## **[목차 2.2 Android의 특징]**

참고문헌 p36 ~ p42 (한빛미디어 “안드로이드의 모든 것 분석과 포팅“ 고현철 유형목 지음)

## **[목차 2.3 Android 라이브러리 계층]**

참고문헌 p45 ~ p47 ((한빛미디어 “안드로이드의 모든 것 분석과 포팅“ 고현철 유형목 지음)

[그림 2.1] 자체 제작

## **[목차 3. 유사어플]**

[그림 3.1], [그림 3.3]

<http://blog.naver.com/ncnscs?Redirect=Log&logNo=10120657240>

[그림 3.2]

<http://blog.naver.com/redieye?Redirect=Log&logNo=10113728740>

[그림 3.4]

<http://cafe.naver.com/dollfin.cafe?iframe_url=/ArticleRead.nhn%3Farticleid=537701>

[그림 3.5], [그림 3.6], [그림 3.7]

<http://blog.naver.com/many1976?Redirect=Log&logNo=50101854342>

## **[목차 4. 개발내용]**

[그림 4.1], [그림 4.2], [그림 4.3], [그림 4.4], [그림 4.5], [그림 4.6] : 자체 제작