Design Document on

Mobile Application Manager

# Authors

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | First Name | Last Name | Email |
| 1 | Surya Kiran | Akula | Surya.akula1@marist.edu |
| 2 | Vishal | Koosuri | Vishal.Koosuri1@marist.edu |
| 3 | Mitin | Sharma | Mitin.Sharma1@marist.edu |
| 4 | Bhargavi | Madunala | Bhargavi.madunala1@marist.edu |

# Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.no | Author Name | Modified Date | Version | Comments |
| 1 | Surya Akula | 03/01/2017 | V01.00 | Initial document |
|  |  |  |  |  |

# Target Audience

This document is intended to make the development team to understand the customer’s requirements and develop working code. Design Document could be seen a part of the Plan on what a developer does.

Table of Contents

[Authors 1](#_Toc476588102)

[Revision History 1](#_Toc476588103)

[Target Audience 1](#_Toc476588104)

[Objective & Approach 3](#_Toc476588105)

[External Design 3](#_Toc476588106)

[Internal Design 4](#_Toc476588107)

[Software Flow 6](#_Toc476588108)

[Error handling 1](#_Toc476588109)

[Test environment 1](#_Toc476588110)

[PACKAGING 2](#_Toc476588111)

[SECURITY 2](#_Toc476588112)

[Accessibility 2](#_Toc476588113)

[Globalization 2](#_Toc476588114)

[Supporting Material 3](#_Toc476588115)

[Risks and Dependencies 4](#_Toc476588116)

# Objective & Approach

With the advent of things going smart and mobile, these days; it is important for an application to provide an ease of access to the user, along with security. Today, there are numerous number of applications that are available in the Mobile market, and the number keeps growing. But, there is one thing every user wants to know, whether the application causes trouble to the system or not.

Considering this, it is very important for the user to keep an eye on the apps which they’re using, to want to know the permissions the app is trying to access, amount of data it’s consuming, the applications uptime, etc. Knowing this, gives the user much more confidence towards using the particular application and hence, a process manager (or metrics manager or task manager) would be really helpful to keep track of the system.

So, the main objective is to design a system metrics manager application, which shows all the various processes running in a mobile. Using this manager, a user can view the information about the application(s)’ data, usage, uptime, and also the cumulative processes running on the system and also can kill it.

# External Design

API - 18+

Libraries:

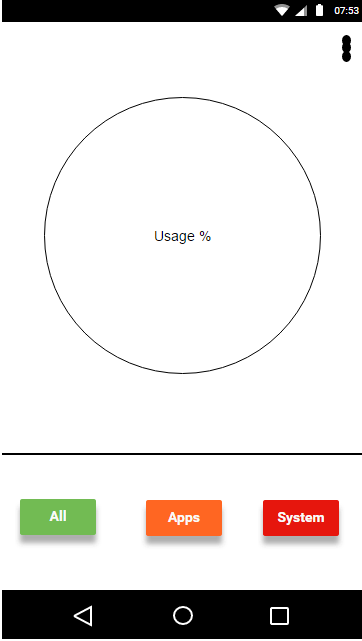
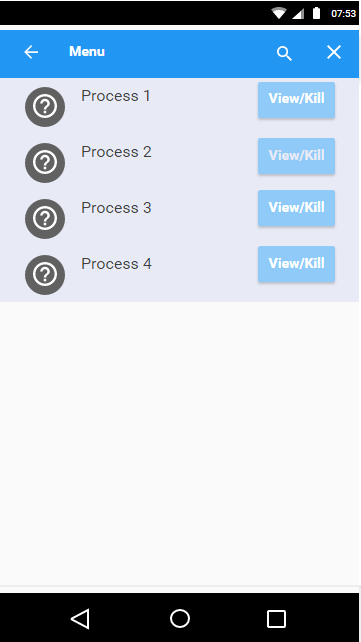
Activity Manager:

This class gives information about, and interacts with, activities, services, and the containing process.

ActivityManager.AppTask – to manage application’s task

.MemoryInfo – information regarding the available memory

.RunningAppProcessInfo – info about running processes

1.0 Main screen 1.1 working Processes display

# Internal Design

Hardware Resources:

Min Requirements for Desktop/ Laptop

RAM – 1 GB

Processor – Intel Core 2 Duo

HDD – 80GB

OS – Windows 7

Mobile

RAM – 512MB

HDD – 8GB

Processor –

OS – Android 4.0

**Development Standards:**

For this project, we decided that we would stick to the Agile development model. This is because of the various advantages that are included in this model considering the group size we are operating.

Since, we are building this project for an Android platform(currently) Java is going to be our go-to language. We might even use Cordova framework for the development purposes which would really helpful if we are planning to use any CSS3, HTML5 and JavaScript.

**Software License:**

This application is going to be a free licensing software. In other words, it is going to be open-source due to the fact that the number of advantages it has over proprietary.

**Development Environment:**

Android Studio – would be our go-to IDE for this project.

GitHub – would be source code repository as we are already familiar with it.

# Software Flow

Database storage

Metrics collection

UI data extraction

Data aggregation

Initialization

Termination

Metrics presentation to user

# Error handling

Well-written applications will dual-purpose logs and activity traces for audit and monitoring, and make it easy to track a transaction without excessive effort or access to the system. They should possess the ability to easily track or identify potential fraud or anomalies end-to-end.

* Fail safe – do not fail open
* Dual purpose logs
* Audit logs are legally protected – protect them
* Reports and search logs using a read-only copy or complete replica

The error handling can be managed by the following library.

ActivityManager.ProcessErrorStateInfo.

# Test environment

We are using JUnits as the testing environment.

JUnit is a Java library to help you perform unit testing. Unit testing is the process of examining a small "unit" of software (usually a single class) to verify that it meets its expectations or specification.

**Writing a test case**

Each unit test method in your JUnit test case file should test a particular small aspect of the behavior of the "class under test." For example, an ArrayIntListTest might have one testing method to see whether elements can be added to the list and then retrieved. Another test might check to make sure that the list's size is correct after various manipulations. And so on. Each testing method should be short and should test only one specific aspect of the class under test.

JUnit testing methods utilize assertions, which are statements that check whether a given condition is true or false. If the condition is false, the test method fails. If all assertions' conditions in the test method are true, the test method passes. You use assertions to state things that you expect to always be true, such as assertEquals (3, list.size()); if you expect the array list to contain exactly 3 elements at that point in the code.

**System Testing**

Traditionally we will be using IDE’s default AVD manager through which we emulate nexus device.

# PACKAGING

This is done automatically with Android Studio as it builds a signed “.apk” file as soon as we run the project with the configurations that are customized accordingly.

# SECURITY

Since, this is a standalone application, there are no permissions that the user has to set. The user just needs to grant access to the application so that it could access all the tasks/processes running inside the device.

This has single role the user(who is Admin himself) who is actually registered with the device.

# Accessibility

The application will be quite easy to operate even for visually impaired users. We will be including the Interactive voice response (IVR) is a technology that allows a computer to interact with people.

# Globalization

English language will be used for the displaying the messages.

There will be translations. This App will have documentation that tells user how to use this app. Project Documentation is an important part of project management. It is substantiated by the essential two functions of documentation: to make sure that project requirements are fulfilled and to establish traceability with regard to what has been done, who has done it, and when it has been done. Documentation will also have Design information, issue tracking, work plan and user manual.

# Supporting Material

**Glossary**

.apk file

Android application package file. Each Android application is compiled and packaged in a single file tha

Activity

A single screen in an application, with supporting Java code, derived from the Activity class.

adb

Android Debug Bridge, a command-line debugging application included with the SDK. Application

Canvas

A drawing surface that handles compositing of the actual bits against a Bitmap or Surface object.

Content Provider

A data-abstraction layer that you can use to safely expose your application's data to other applications.

Dialog

A floating window that acts as a lightweight form. A dialog can have button controls only and is intended to perform a simple action (such as button choice) and perhaps return a value.

View

An object that draws to a rectangular area on the screen and handles click, keystroke, and other interaction events. A view is a base class for most layout components of an Activity or Dialog screen (text boxes, windows, and so on).

Ram Usage: Ram used by specific app

Total number of processes: Total number of running processes

Threads: Threads use by an application at one time

Data usage: data used by app that is stored on ram

Storage: data consumption by app

Instance running: tells no of instance running against the app

Battery: battery used by particular app

**References:**

Web sites

https://developer.android.com/reference/java/lang/ref/Reference.html

Books

App Development & Programming Guide: Learn In A Day!

# Risks and Dependencies

Some of the methods can be deprecated with the advancements in the API levels. And, also it’s a standalone project which depends on core libraries of the android OS.