**Design document for “PCorrector” project.**

Madi Uristen

Aidarov Shokhan

Ordabayev Akylbek

Shalkarbai Nurzhigit

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Change |
| 0.1 | 27/03/16 | Upper Team | Initial document |
| 0.2 | 02/04/16 | Upper Team | Complete document |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of contents**

**1 Introduction** ........................................................................................................................................3

**1.1 Purpose** ............................................................................................................................................3

**1.2 Scope** ................................................................................................................................................3

**1.3 Definitions, Acronyms, Abbreviations** …..........................................................................................3

**1.4 Design goals** .....................................................................................................................................3

**2** **References ………………………………………………………………………………………………………………………………….**3

**3 Decomposition Description …………………………………………………………………………………………………………**4

**4** **Dependency Description ……………………………………………………………………………………………………………..**6

**5 Design Issue ………………………………………………………………………………………………………………………………..**8

## 1 Introduction

* 1. **Purpose**

The purpose of this document is to explain the design and architecture of the “PCorrector” Application.

**1.2 Scope**

This document covers system decomposition, interfaces, and dependencies, as well as design

rationale.

**1.3 Definitions, Acronyms, Abbreviations**

**1.4 Design goals**

1. Reliability : The server must work continually and fully consistent.

2. Maintainability: no

3. Extensibility: Our application must be easy extensible, future upgrade must take place without any difficulties.

4. User Privacy: User can do absolutely all things which he have access.

2. References

[NONE]

**3. Decomposition Description**

**3.1 Data Decomposition**

**3.1.1 Database for storage a photo data that was already corrected.**

**3.1.1.1** **‘Ready photo’ table in database.**

photo\_id INTEGER AUTO\_INCREMENT

photo\_source\_name STRING

**3.1.1.2** **‘Send Photo’ table in database.**

photo\_id INTEGER AUTO\_INCREMENT

photo\_source\_name STRING

**3.1.2 Space in mobile phone for save photo there.**

**3.1.1.1 “PCorrector” folder in phone.**

**3.1.1.2 ‘Ready Photo’ folder in “PCorrector” folder.**

**3.1.1.3 ‘Raw photo’ folder in “PCorrector” folder where photo from camera or gallery will be saved.**

**3.1.1.4 User can add any folder and save photo.**

**3.1.3 RImage class.**

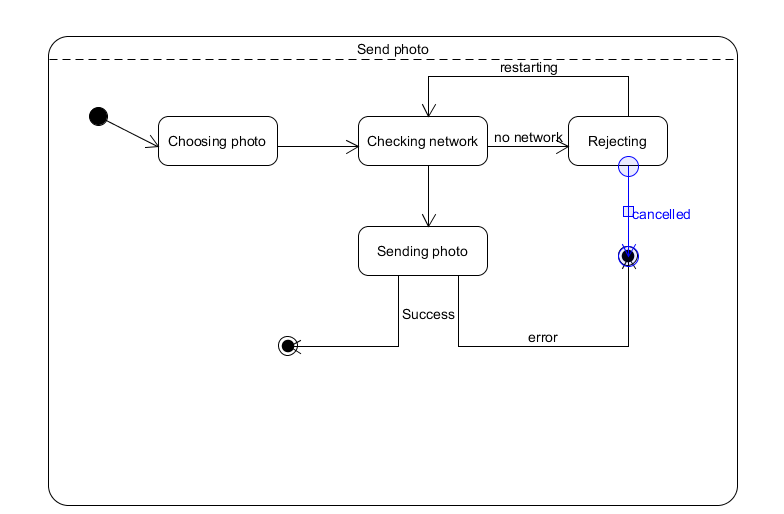
**RImage class will be store photo bitmap , folder name where photo is stored.**

**Bitmap photob;**

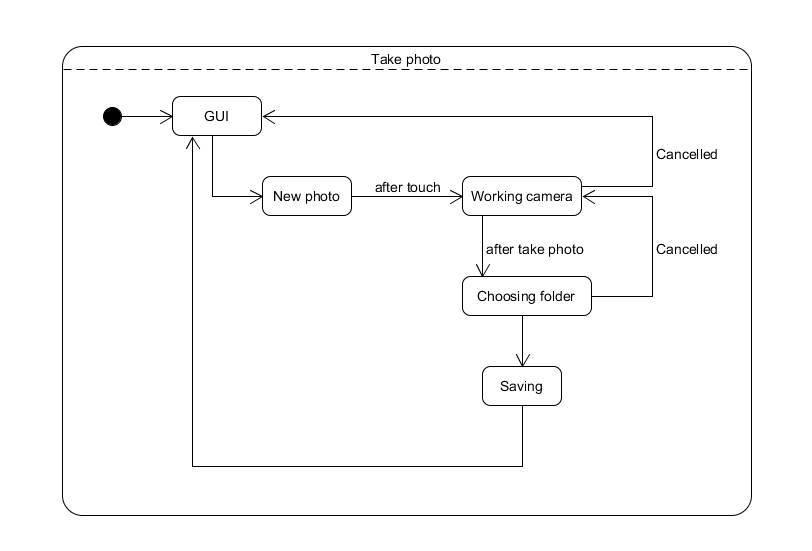
**String psource;**

**3.2 State Decomposition**

**3.2.2 Sending photo**

****

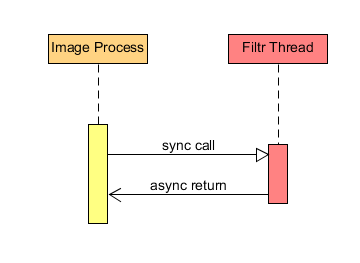
**3.2.2 Taking photo**

****

**4 Dependency Description.**

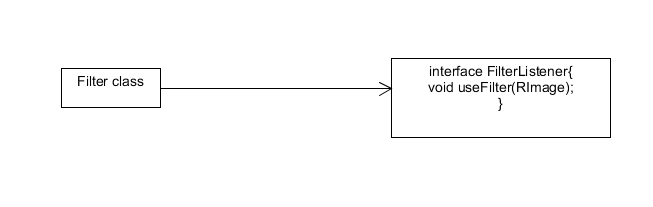
**4.1 Process-Thread communication.**

When we use some filter the process create thread which take photo bitmap and do some operation with bitmap and send it to process.

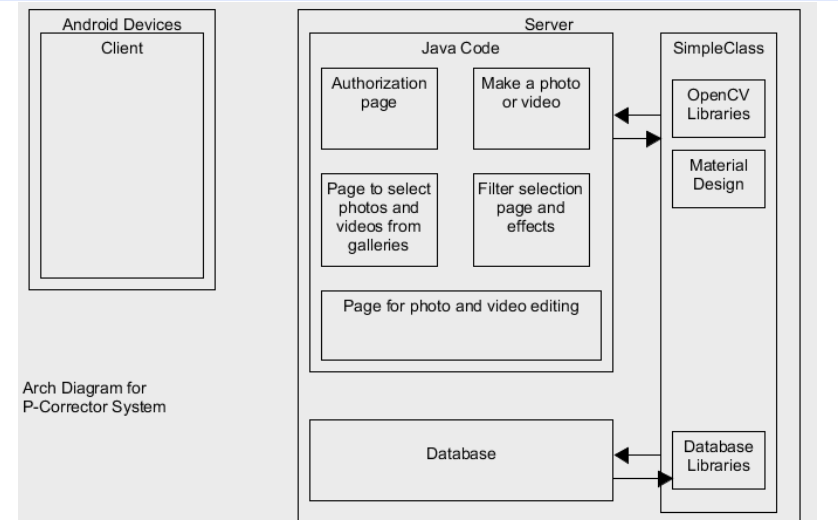
****

**4.2 Filter-FilterListener relationship.**

All filter will extends from Filter that implements from FilterListener . When user use filter the system will call useFilter() function.

****

**4.3 Inter-module dependency.**



**5. Design Issue**

**5.1 Use Bitmap**

5.1.1 Description

Our application should read and write image bitmap.

5.1.2 Factors affecting the issue

5.1.2.1 Our application should use some filter to change photo bitmap.

5.1.3 Alternatives and their pros and cons

5.1.3.1 OpenCV is also good alternative.

5.1.4 Resolution of the Issue

We decided to use standard ‘Bitmap’ android library to read and write image.

**5.2 Use SqlLite**

5.2.1 Description

Our application should use one Database.

5.2.2 Factors affecting the issue

5.2.2.1 Our application should store some information

5.2.3 Alternatives and their pros and cons

5.2.3.1 MySQL. It’s like SqlLite.

5.2.4 Resolution of the Issue

We decided to use SqlLite because SqlLite easy to integrate with Android.

**5.3 AsynTask**

5.3.1 Description

With thread we can do some action parallel.

5.3.2 Factors affecting the issue

5.3.2.1 Image processing is long process.

5.3.3 Alternatives and their pros and cons

5.3.3.1 Thread and Runnable interface. Also works like AsynTask.

5.3.4 Resolution of the Issue

We decided to use AsynTask because this class have two function .First onPreExicute() which calls when we start our AsynTask and onPastExicute() calls when AsynTask finished.

**5.4 External Storage**

5.2.1 Description

To store photos in phone.

5.2.2 Factors affecting the issue

5.2.2.1 Our application should photo in mobile phone

5.2.3 Alternatives and their pros and cons

5.2.3.1 Internal Storage.

5.2.3.1.1 Internal Storage also can store photos but when user delete our application all photos also will be deleted.

5.2.4 Resolution of the Issue

We decided to use External Storage because in this case user can use all photos for any application.