Institiúid Teicneolaíochta Cheatharlach
INSTITUTE of TECHNOLOGY CARLOW
At the Heart of South Leinster

Student ID card Barcode Recognition for Android Mobile Phone

Design Manual

Student Name: Long Long

Student ID: C00131028

Supervisor: Christophe Meudec

Date: 18 Jan 2010

Table of Contents

1. Introduction

This Mobile Barcode Recognition(MBR) is an Android platform application which focuses on 1D barcode(Code 39) image processing. The aim of it is to develop an application using built-in camera on mobile phones to process and decode barcodes on the device locally.

Once it reads the barcode on student card and decodes it successfully, an option to get detail information of the card holder from web server via network is available for users. For some security reasons, we cannot access college student database or its relevant web server, an analogical web server and student database is including in this project. So instead of connecting to college web server for details of card holders, the application(MBR) will connect to the analogical web server which is also a part of the whole project.

This MBR application is a foreground activity. That means it's only useful when it's in the foreground and is effectively suspended when it's not visible. The Activity life cycle needs to be considered carefully so that the Activity switches seamlessly between the foreground and the background. To achieve this goal, state of the application needs to be saved when the Activity becomes invisible. When returns to the foreground, it presents the exact same state. Additionally, "games and map mashups are common examples".[1]

There are two parts in the domain, one is the local MBR application and the other one is web server. As the major work of the project is building application on Android, the web server design will be less detailed in this manual and it's no more than a general web server as those now running and working with databases.

2. Design Theory

It's seriously necessary to follow these guidelines during Android software design:

Well behaved
"Start by ensuring that your Activities suspend when they're not in the foreground. Android triggers event handlers when your Activity is suspended or resumed so you can pause UI updates and network lookups when your application isn't visible — there's no point updating your UI if no one can see it. If you need to continue updating or processing in the background, Android provides a Service class designed to run in the background without the UI overheads." [2]

Switches seamlessly from the background to the foreground
"With the multitasking nature of mobile devices, it's very likely that your applications will regularly switch into and out of the background. When this happens, it's important that they "come to life" quickly and seamlessly. Android's nondeterministic process management means that if your application is in the background, there's every chance it will get killed to free up resources. This should be invisible to the user. You can ensure this by saving the application state and queuing updates so that your users don't notice a difference between restarting and resuming your application. Switching back to it should be seamless with users being shown the exact UI and application state they last saw."[2]

Polite to other activities
"Your application should never steal focus or interrupt a user's current activity. Use Notifi cations and Toasts (detailed in Chapter 8) instead to inform or remind users that their attention is requested if your application isn't in the foreground. There are several ways for mobile devices to alert users. For example, when a call is coming in, your phone rings; when you have unread messages, the LED fl ashes; and when you have new voice mail, a small "mail" icon appears in your status bar. All these techniques and more are available through the notifi - cation mechanism."[2]

Presents a consistent user interface
"Your application is likely to be one of several in use at any time, so it's important that the UI you present is easy to use. Don't force users to interpret and relearn your application every time they load it. Using it should be simple, easy, and obvious — particularly given the limited screen space and distracting user environment."[2]

Responsive
"Responsiveness is one of the most important design considerations on a mobile device. You've no doubt experienced the frustration of a "frozen" piece of software; the multifunction nature of a mobile makes it even more annoying. With possible delays due to slow and unreliable data connections, it's important that your

application use worker threads and background services to keep your activities responsive and, more importantly, stop them from preventing other applications from responding in a timely manner."[2]

3 Application Design

Some name changes from previous documents:

- Edge Trimming now has been changed to Barcode Detect(see class *BarcodeDetector*)

3.1 Graphic User Interface

3.1.1 Capture UI:

This UI contains the grey box and all stuff inside the grey box. The grey box(actually it will be semi-transparent so that it's able to display captures) is where to display all camera captures. While the black box area is where effective capture taking place. It is supposed to be the only area that the *CaptureActivity* captures and transfers to other classes. The red line helps users with positioning barcode parallel with camera x-axis(the horizontal axis).The idea(the red horizontal line) comes from a similar application on iPhone, whose presentation was put on YouTube.com as a video fragment.[3]

The point of this red line is to ensure that as less work as it could give to the application and it really helps in improving process speed (even more effective than a better algorithm does).
I individually think it's more than a kind of auxiliary locator on that iPhone application. It turns green when a barcode is recognized and it has such functions like indicating effective barcode edge and the recognize level. But even without these functions and just a simple auxiliary locator, it will help a lot in improving speed.

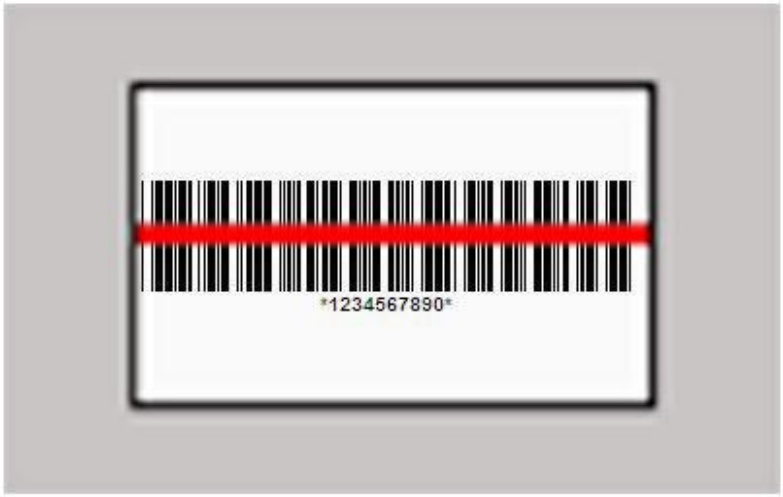

Figure 1: Capture UI

3.1.2 Result and Connect to Web server UI

In this UI, recognized barcodes were displayed(string “1234567890” in the picture). While it’s also the “connect to web server” UI. User name and password are required to connect to and get details from web server, or a failed connection happens. The reason of requiring user name and password is, only illegal college staff members are able to access this very serious information of students’ database.

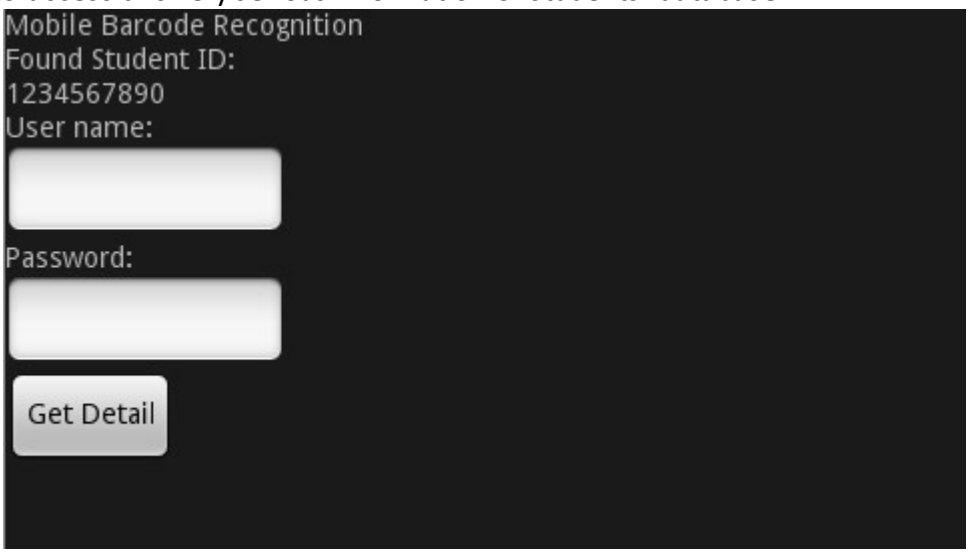

Figure 2: Result and Connect Web server UI

3.1.3 Details display UI

Details of student will be displayed in this (or similar) way.

Figure 3: Details display UI

3.1.4 Menu

When pressing “Menu” button on the phone, an option menu will provide. There will be some components in the menu:

- Setting
  Set general options like warning tone.
- History back view
  View the previous recognized barcodes
- Help
  Give some guidelines to user
- About
  Some version information

3.2 Use Case Diagram & Use Cases

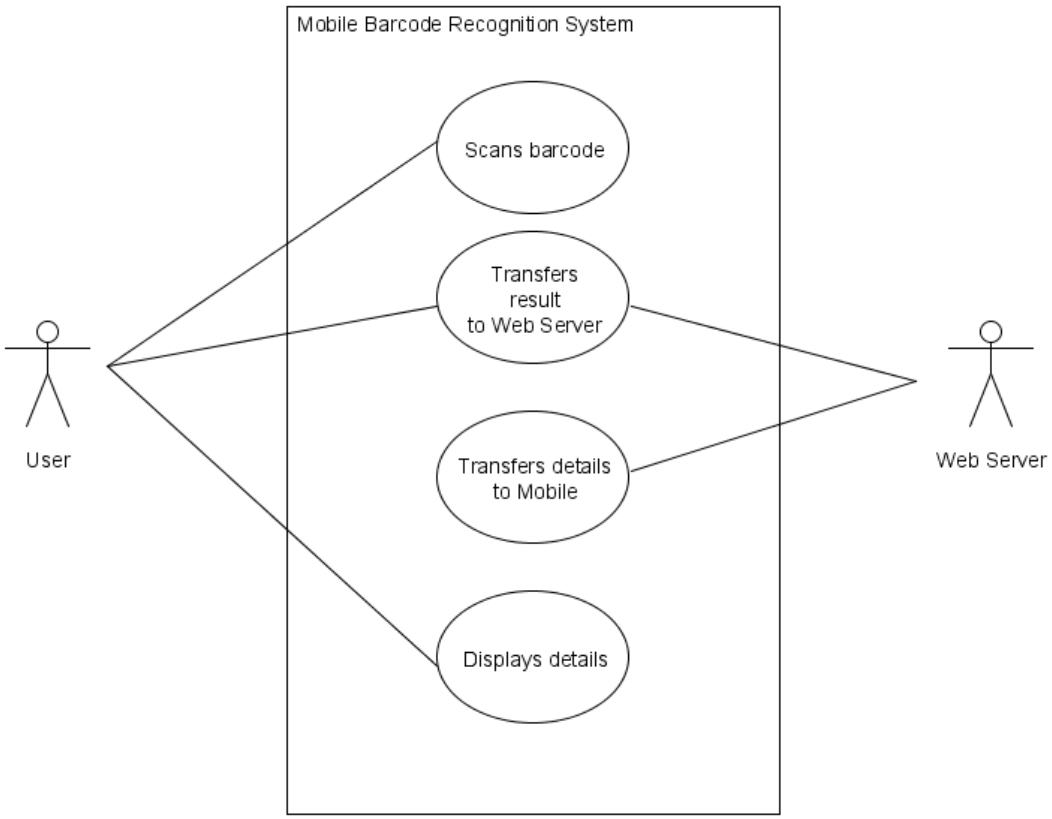

Figure 4: Use case diagram

3.2.1 Use Case 1

Use Case: Scans barcode

Actors: User

Description: User runs the Mobile Barcode Recognition program on the Android mobile, scans the barcode area of student ID card. The program takes charge of the scanned images, analysis and decodes the image of barcode. And when worked out, it displays the result onto the screen. User gets the result of decoded barcode info and stops scanning.

Main Success Scenario:

1. User runs Mobile Barcode Recognition System and scans barcode area.
2. System analysis the image and work out it
3. System displays the result onto the screen.

Alternatives:

1.1 If the barcode image cannot be recognized by System, User needs to re-scan the barcode area.

3.2.2 Use Case 2

Use Case: Transfers result to Web Server

Actors: User, Web Server

Description: User presses transfer button to transfer decoded barcode info to the Server via wireless network. User is asked for login name and password to log in. User inputs his valid login name and password and logs in to the Server. After a success login, System transfers to barcode info to Server.

Main Success Scenario:

1. User inputs login name and password
2. User presses get detail button (on the screen)
3. System connects to Web Server with username and password
4. System transfers result to Web Server

Alternatives:

3.1 If an invalid login name or password has been input, User has up to 3 times to retry it or the Web Server will ban this transferring.

3.2.3 Use Case 3

Use Case: Transfers details to Mobile

Actor: Web Server

Description: Web Server looks up Database for corresponding details of the decoded barcode info of the student card holder. When working out the corresponding details, Web Server transfers it to Mobile Barcode Recognition System on the request mobile.

Main Success Scenario:

1. Web Server works out the corresponding details of the barcode image info
2. Web Server transfers the details to Mobile Barcode Recognition System

3.2.4 Use Case 4

Use Case: Displays details

Actor: User

Description: When getting the details, System displays it onto the screen.

Main Success Scenario:

1. System displays the details onto the screen

3.3 Domain Model

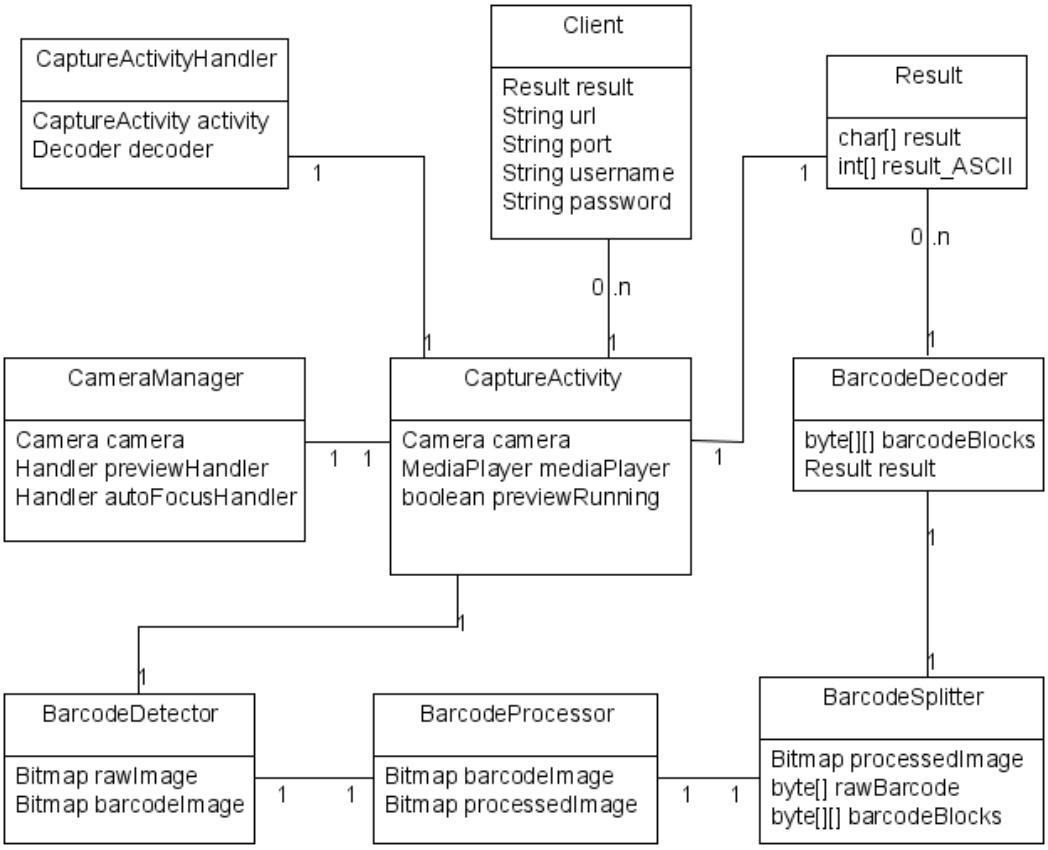

Figure 5: Domain model

CaptureActivity

This is the one which the main function(onCreate()) appears. The core functions are described as below:

CaptureActivityHandler

This class is designed mainly to help *CaptureActivity* handle messaging. The CaptureActivityHandler schedules messages to be executed as some point in the future and to enqueue an action to be performed on a different thread. [4]
Message may be used as a parameter for states switching.

CameraManager

The central work of this CameraManager class is managing camera service.

BarcodeDetector

Once images were captured, barcode area needs to be detected. This class is designed to detect the area which barcode locates. It actually is edge trimming of Digital Signal Processor in previous document(Specification Manual). Why separate it from Digital Signal Processor and make it an individual class? Because if not, there

will be so many works left for barcode converter and processor. The speed will be slow down and it costs more resource to complete the same amount of work. It doesn't follow the theory of mobile software design.

BarcodeProcessor
It includes all image processing and output the processed image.

BarcodeSplitter
Usage of this class is to split processed barcodes into barcode blocks. So that *Decoder* is able to decode them.

BarcodeDecoder
This class charges barcode decoding.

Result
The result from *BarcodeDecoder* stores in objects of class *Result*.

Client
This class is used to connect to web server and transferring data between local application and web server.

3.4 System Sequence Diagram

3.4.1 SSD for Scans barcode(Use case 1)

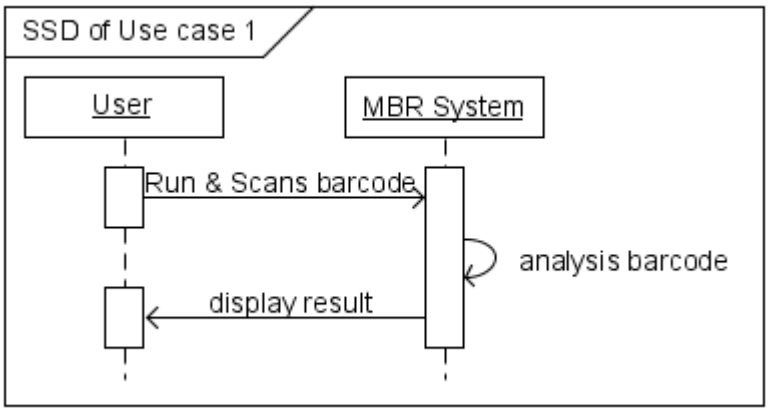

Figure 6: System Sequence Diagram – Use case 1

3.4.2 SSD for Transfers result to Web Server(Use case 2)

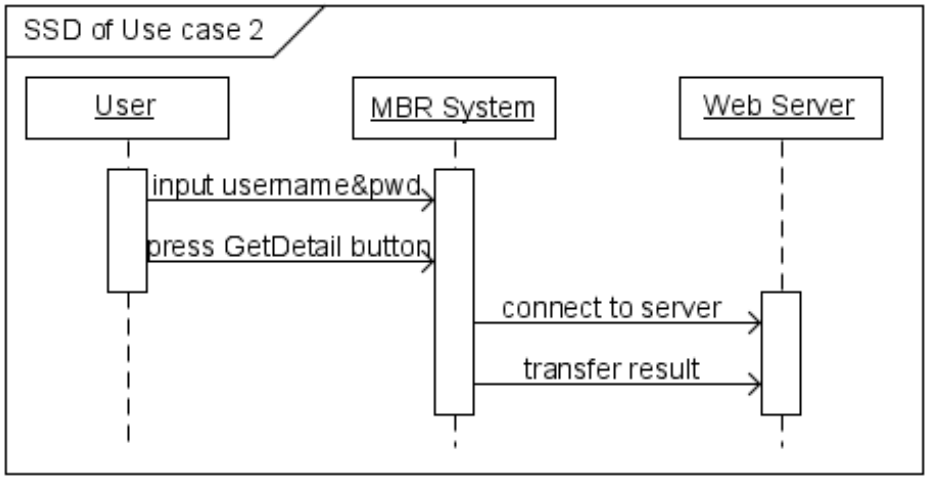

Figure 7: System Sequence Diagram – Use case 2

3.4.3 SSD for Transfers details to Mobile(Use case 3)

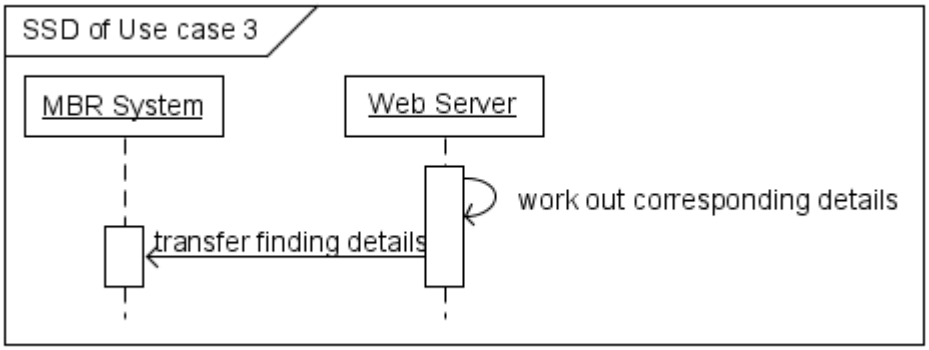

Figure 8: System Sequence Diagram – Use case 3

3.4.4 SSD for Displays details(Use case 4)

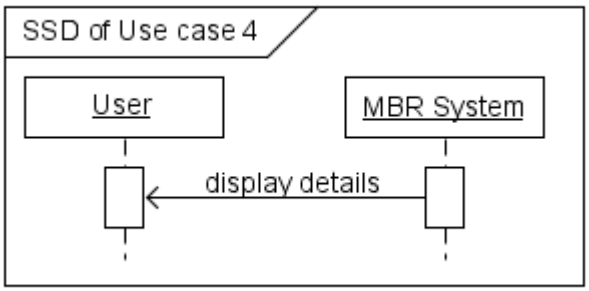

Figure 9: System Sequence Diagram – Use case 4

3.5 Class Diagram & Classes

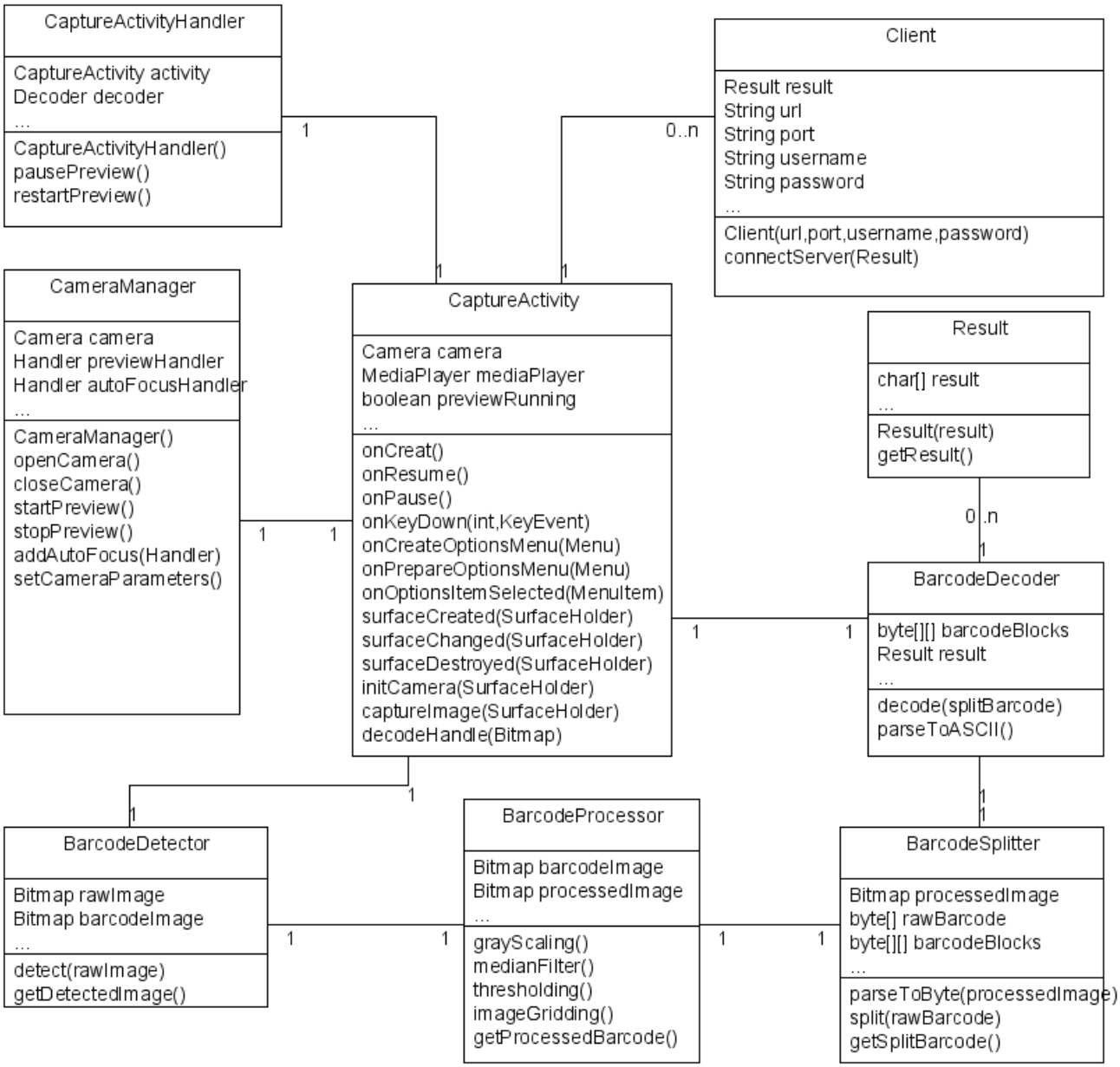

Figure 10: Class Diagram

3.5.1 CaptureActivity

Main functions:

- Built UIs
- Initialize camera, camera manager and handlers
- Add event listeners
- Deal surface event and views

3.5.2 CaptureActivityHandler

Main functions:

- Handle messages between classes
- Restart preview when needed

3.5.3 CameraManager

Main functions:

- All camera driver dealings
- Camera preview start and stop
- Auto focus appending
- Set camera parameters and get screen resolution.

3.5.4 BarcodeDetector

Main functions:

- detect(rawImage)
  detect the barcode area and return barcodeimage: Bitmap
- getDetectedImage()
  return barcodeImage(detected)

3.5.5 BarcodeProcessor

Main functions:

- grayScaling()
  gray-scale the image
- medianFilter()
  do median filter
- Thresholding()
  do thresholding
- imageGridding()
  grid images

- getProcessedBarcode()
  a function for external class getting processed barcode

3.5.6 BarcodeSplitter

Main functions:

- parseToByte(barcodeImage)
  parse Bitmap barcodeImage to byte[] format
- split(rawBarcode)
  split raw barcodes, return byte[][] barcodeBlocks
- getSplitBarcode()
  return barcodeBlocks

3.5.7 BarcodeDecoder

Main functions:

- decode(splitBarcode)
  decode the splitted barcode
- parseToASCII()
  parse the decoded barcode to ASCII format for further use.

3.5.8 Result

Main functions:

- Result(result)
  use to build object of *Result*
- getResult()
  a function for external class getting result.

3.5.9 Client

Main functions:

- Client (url,port,username,password)
  build objects of *Client*
- connectServer(result)
  connect and transfer result to web server

3.6 Interactions of Main Use Cases

Use case 1 “Scans barcode” is the most core use case among all use cases. In here below it will be detailed. The second scenario in Use case 1 contains almost all works that the application does. Here a detailed description is provided by using interaction diagram.

Additionally, use case 3 and use case 4 are less important and really simple, so no need to be detailed here.

3.6.1 Use Case 1: Scans barcode

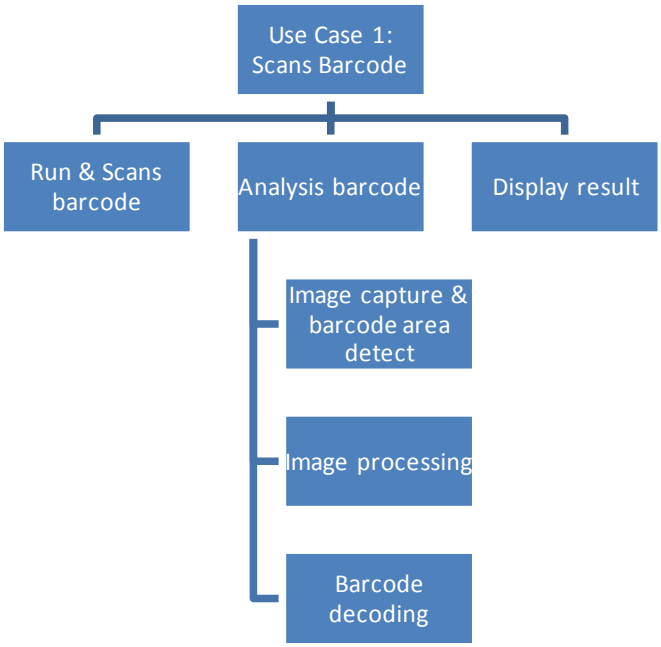

Figure 11: Hierarchy of Use Case 1

Contract 1: Run & Scans barcode (Omitted)

Contract 2: analysis barcode

- CO1: Image capture and barcode area detect

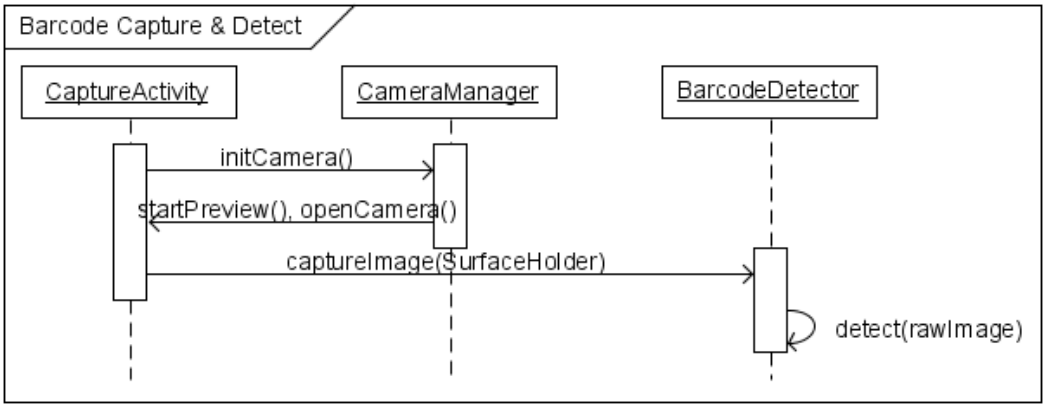

Figure 12: Barcode capture & detect

- CO2: Image processing

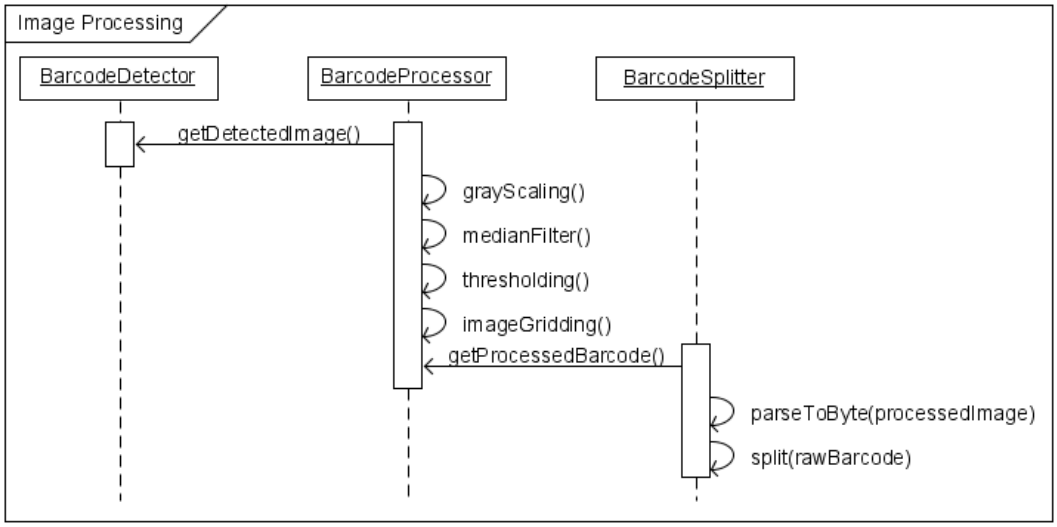

Figure 13: Image processing

Contract 3: Display result (Omitted)

- CO3: Barcode decoding

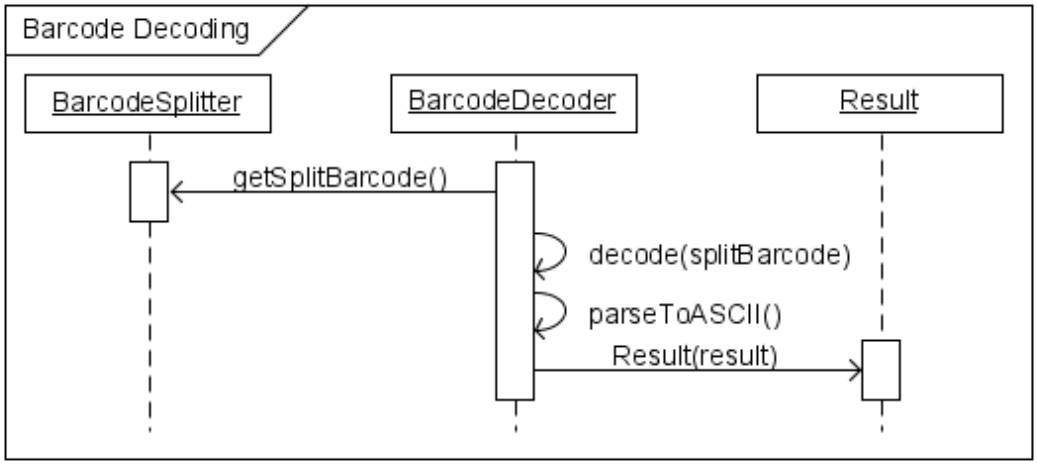

Figure 14: Barcode decoding

Contract 3: Display result (Omitted)

3.6.2 Use Case 2: Transfers result to Server

Contract 1: Input username & password (Omitted)
Contract 2: Connect to Server

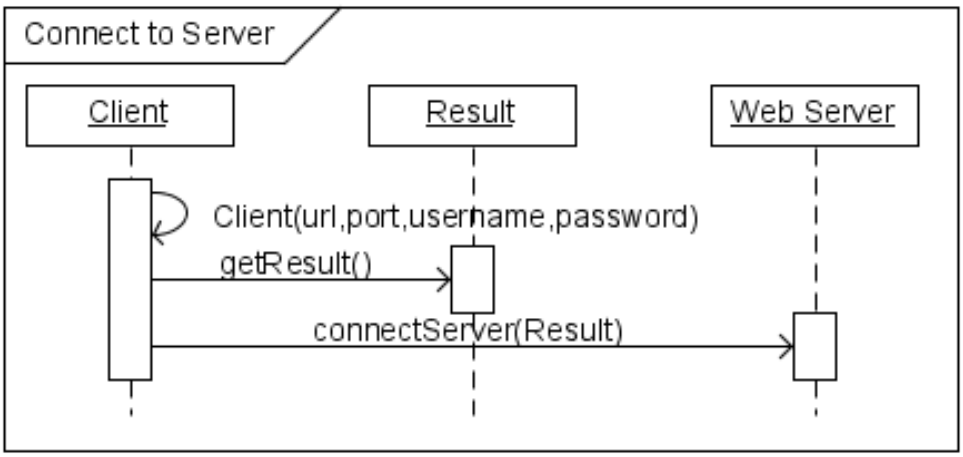

Figure 15: Connect to server

4 Web Server Design

4.1 Dummy Database Table

Field	Data Type
StudentID	Varchar(10)
Surname	Varchar(20)
Firstname	Varchar(20)
Gender	Varchar(6)
HomeAddress	Varchar(50)
DateOfBirth	Date
Tel.No.	Varchar(20)
MobileNo.	Varchar(20)
Email	Varchar(30)
Course	Varchar(30)
CourseCode	Varchar(10)
EmploymentRecord	Varchar(50)

Table 1: student's database structure

4.2 Web-based Database Manager

A web based database manager(management system) will be built in order to manage dummy student's database.

5. Reference

1. “*Professional Android Application Development”*, ISBN: 0470344717, Reto Meier, pp29
2. “*Professional Android Application Development”*, ISBN: 0470344717, Reto Meier, pp34

3. “*iPhone live barcode scanner by Vision Smarts*”, benoit872(14-Sep-2009),
Web: http://www.youtube.com/watch?v=jNDu0Rb0vRY&NR=1

4. “*Android SDK API/android.os.Handler/Class Overview”,* Android Developers, Google.com,
Web: http://developer.android.com/intl/zh-CN/reference/android/os/Handler.html