Android application:
Smart Stopwatch

Name: Fuxing Liu
Student ID: 5100309301

Name: Meidan Zhao
Student ID: 5100309543

Contents

Abstract

In order to time quickly and accurately, we urgently need a reliable timing tool with a certain function. However, the current timer is either too simple - it can do nothing except timing, or too expensive - not suitable or necessary in some smaller application occasions. To make it possible to calculate and record individual result automatically, we have Android system to expand the cell phones timer function, making the timer function more powerful, inexpensive, and universally applied. At the moment, the new timer can not only record the time, but also store additional information and organize them. This report is a design illustration of an Android application named "Smart Stopwatch", which is based on the actual requirements of an automatical stopwatch.

Chapter 1

Application background

1.1 Two application scenarios

The value of an application is measured by its practicability, i.e., whether it is useful in certain circumstances. Before the designing illustration part, let us look back two scenarios in our daily life.

Scenario 1: In the outdoor stadium, there are more than 20 girls who are ready to take the 4×400 meters relay race. As the track is only 400 meters, each girl has to finish one lap. The most unlucky one would be the timekeeper. He must stand at the finish line along the whole race. Whenever there is a girl running over the finish line, the timekeeper must press the traditional stopwatch, and shout a number representing the ranking. After the race, all the girls must report their ranking number, in order to register each result. But that is not the end of timekeeper's work. He still has to make some effort to calculate each girl's time and sort them in order. For example, the personal result of No.102 athlete is the total time when No.103 running by minus the total time when No.102 running by. This procedure is both complex and vulnerable. What if a girl lied about her ranking by purpose? What if once the timekeeper presses the stop-watch not hard enough and the time is not recorded by the watch? Every small mistake in this procedure can result in a fatal error of the whole timing system.

Scenario 2: Same thing happens in university sports festival-a 4×100 meters relay race. We not only want to record each groups performance, but also want to record every team members individual result. Every 100 meters there will be a timekeeper, who must record the individual number, the group number and the time of an athlete when he running through this stopwatch. With the help of pen and paper. it seems not so complex. However, since the requirements of timing accuracy and speed becomes higher and higher, this is no longer a easy task. What is more, if you want to get a whole ranking of this race, you have to synthesize all the records from 4 timekeepers and make some effort in calculating. Why not develop an application on cell phone to realize this stop-watch function?

1.2 The importance of an Android-based stopwatch

From the above two scenarios we know, one thing is very important in such races: dependable timing system. The runners, or the athletes, they care much about their own performance as well as others'. Besides, the accuracy of the race result relates to fairness.

In a running race which is not such professional(as scenario 2 shows, a relay race held in a university), a traditional method is widely used, that is, one timekeeper with a few stopwatches and a result board. However, this timing system has fatal weakness: the procedure is complex and vulnerable as scenario 1 shows. What is worse, when a group of people run over the finish line at the same time, it is hardly possible to record everyone's time correctly and accurately. So this kind of timing system is out of date.

Someone may argue that there is already a very good solution: RFID(Radio Frequency Identification) timing carpet. This carpet does have lots of advantages, as you know, it can record one's time automatically as the man runs over it. However, only one factor prevents it from widely use, that is, the price. An RFID timing carpet deserves thousands of dollars. Not every organizing committee could afford it or is willing to pay for it.

Thus, we urgently need a reliable timing tool with certain characteristics–powerful, inexpensive, and universally applied. This is our starting point of the Android application–Smart Stopwatch.

Chapter 2

Design principles

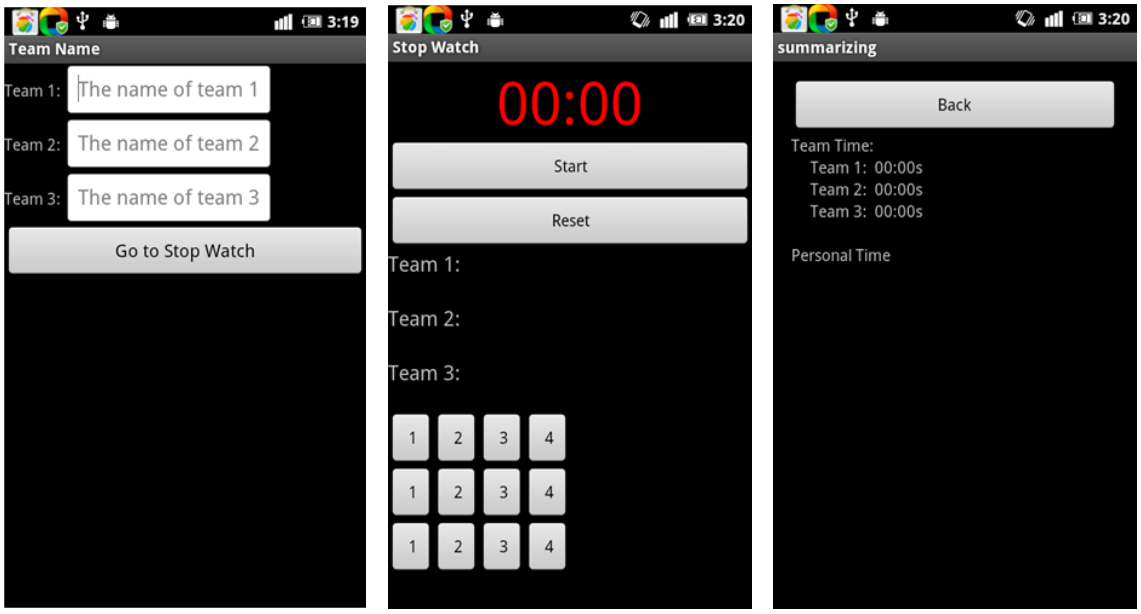
2.1 User interface

UI(User Interface) allows interacts between software and users, i.e., it is regarded as the connection part. This part is so significant that we put it precedence over other parts. The details will be shown in next two sections, here we just give the brief instructions about three UIs in Smart Stopwatch.

In the middle(Figure 2.1(b)) is the main interface. After the stopwatch is initialized, each time an athlete runs over the finishing line, you can press the corresponding button then his/her time is recorded here. Note that in a normal race, pause is obviously not allowed. So the pause button does not exist. There are only start and reset buttons. The left one(Figure 2.1(a)) is the customization interface, and the team name you type in will appear in next interfaces. After you stop the timer, a summarize interface is shown in Figure 2.1(c). The time used by each person is shown from the fastest to the slowest.

2.2 Basic ideas about Smart Stopwatch

An application based on cell phone provide the benefits of portability, efficiency and low-consumption. The Android platform also has its own advantages: it is an open-source operating system and developers can look over other's design for reference. After combined the existing application with own ideas, a more powerful application is born. In fact, the basic stopwatch function is synthesized in Android developing toolkits. That is to say, we do not have to re-write the basic function from the bottom layer. We should focus on the ideas.



(a) the team name UI (b) the main UI (c) the summarizing UI

Figure 2.1: UI of Smart Stopwatch

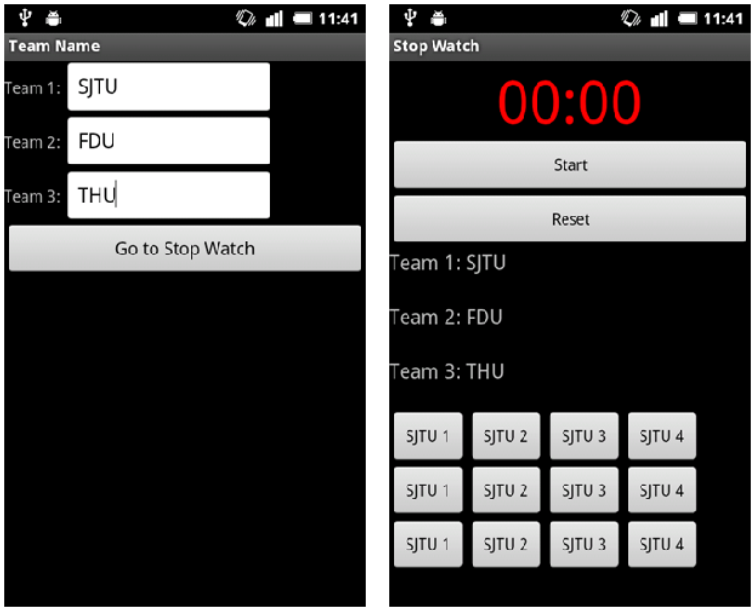
After discussing the shortcomings of current timing systems, we stretch out these descriptions as the basic rules of Smart Stopwatch:

- Easy to manipulate
- Calculate everyone's result automatically
- Can be customized
- Directly perceived

And we will give a concise description one by one.

2.2.1 Easy to manipulate

Traditional stopwatch, no matter mechanical or electronic, requires a rather complex procedure to manipulate it. In general, each button on the traditional stopwatch has multiplex functions, and sometimes you have to look up the instruction to get used to it. This problem can be solved easily with the help of Android platform. Here we place several buttons on the screen, and each button represents one racer(for example, the button in row 2, column 3 and signed with number 3 represents the third racer in team 2). In other word, there is a one-to-one relationship between racers and buttons. The timekeeper just need to press the corresponding button when a racer running over the finishing line. In this way, the manipulation is visibly simplified.



(a) customization in the team name UI

(b) results in the main UI

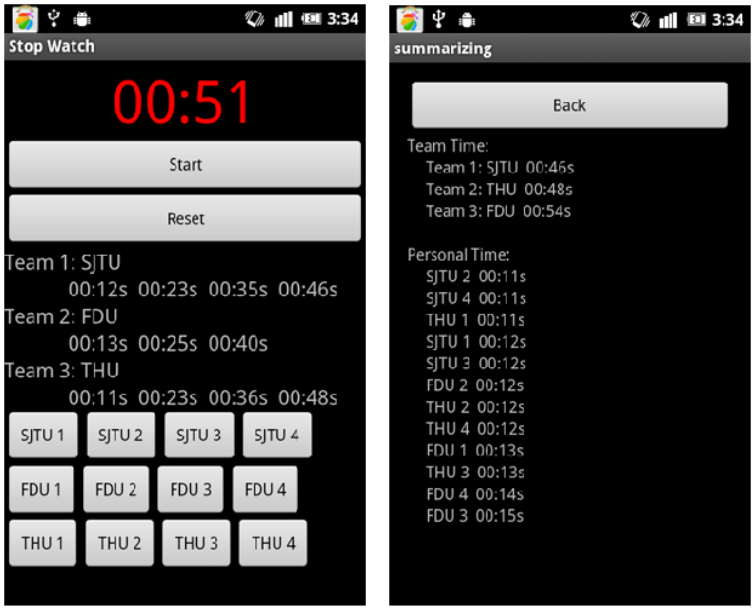
Figure 2.2: Customization

2.2.2 Calculate everyone's result automatically

Another complex procedure in traditional timing system is calculating. Each racer's result must be calculated by substraction, so lots of time is wasted. Besides, a small mistake can cause ripple effects and all other results go wrong. However, in Smart Stopwatch, this is no longer a stubborn problem. This software can calculate each racer's result automatically and immediately. After you pressed a certain button, you can check the result of the corresponding racer in the summarizing UI, and it is in real-time.

2.2.3 Can be customized

If there are more than five teams at the same time, one single screen cannot contain all the buttons and textboxes, and the timekeeper must scroll the screen to reach all buttons. Thus, the misoperation rate will be higher without a clear sign about button information. Among the three UIs, the team name UI is used for customization. After you typed the team's name in the textbox, the buttons and result information in next two UIs will be updated immediately. From Figure 2.2 we can see that every button is distinguished with each other, thus the potential misoperation is avoided. Because team names are shown in buttons and results, no matter how many teams or how many racers exist, this application can handle all of them.



(a) results in the main UI (b) results in the summarizing UI

Figure 2.3: Directly perceived

2.2.4 Directly perceived

Another benefit of customization is that the results are directly perceived. We still use the example in Figure 2.2, a race between three universities. Figure 2.3 shows the results during the race. From Figure 2.3(a), we can see the total time when every racer is running by is recorded in sequence. We can easily get the information that SJTU team got first and only FDU team has not finished the race. And in Figure 2.3(b) we can see the summarizing results after the last one returns to finishing line. The result of every racer is calculated and sorted in order. In fact, this is exactly the final result, the ranking of teams and racers are shown. So the Smart Stopwatch is directly perceived.

2.3 Function hierarchy

In this Smart Stopwatch application, we have accomplished the basic timing function: each button corresponds to one racer, calculate the result of each racer automatically, team name can be customized, the final ranking is directly perceived, etc. However, this is not all of the design. These functions are all based on the traditional stopwatch, and we want something different. After measurement and calculation, what is our next goal to achieve in this application?

Let us review the scenario 2 in previous chapter. In that a 4×100 meters relay race, for the handover happens in different places, if you want to get the result of each stage, you still have to arrange 4 timekeepers and synthesize all the records from those four

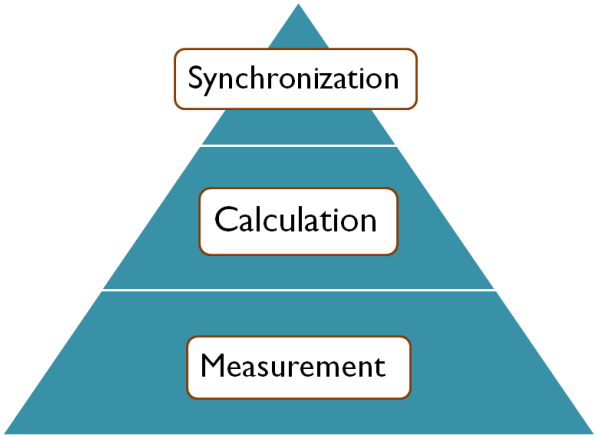


Figure 2.4: The function hierarchy

after the race, and some manual operation is still necessary. Why not let those terminals work simultaneously? This is the final function of Smart Stopwatch: synchronization. The whole function hierarchy is shown in Figure 2.4, and details will be discussed in next chapter.

Chapter 3

Software realization

3.1 Timing component: Chronometer

Chronometer is an timing component provided by Android Developing Toolkit. This component can display a period of time in *CharSequence* format, but it is not related to the current time. Chronometer only cares about how long has it been since the starting point. Apparently, Chronometer is exactly suitable for this Smart Stopwatch application.

When the start button is pressed, the Chronometer is triggered and started working. In normal circumstances, the base time is set to zero. The main challenge appears in the stopping part. Actually, the Chronometer should never stop in the whole process, for a real race never pauses. So after one button was pressed, the Chronometer only record the current time and continues working. We define several TickListeners bounded to the Chronometer, as long as the corresponding button was pressed, the current time is recorded in a char sequence and can be calculated after processing. The processed record is saved in integer format, and can be subtracted or sorted to meet different demands. Some critical source codes are shown below:

```
        start.setOnClickListener(new OnClickListener()
                {
                        @Override
                        public void onClick(View source)
                        {
                                ch.setBase(SystemClock.
                                   elapsedRealtime());
                                ch.start();
                        }
                });


                reset.setOnClickListener(new OnClickListener()
                {
                        public void onClick(View source)
                        {
```

```
                                ch.setBase(SystemClock.
                                    elapsedRealtime());
                                ch.stop();
                        }
                });

            int score11;

                st11.setOnClickListener(new OnClickListener()
                {
                        public void onClick(View source)
                        {
                                CharSequence temp = ch.getText();
                                String str=temp.toString();
                            int score11=Integer.parseInt(str,0)*600+
                                Integer.parseInt(str,1)*60
                                                +Integer.parseInt(
                                                    str,3)*10+Integer
                                                    .parseInt(str,4);
                                s11.setText("␣"+temp+"s␣");
                        }
                });
```

3.2 Use Bundle to connect between Activities

Activity is a significant concept and major component in Android development. An application usually contains several Activities, and each Activity shows different operating interface to the users. A proper configuration of Activities is necessary to a well-designed application. After configuration, the data sharing between Activities is the major challenge. In this Smart Stopwatch application, the data sharing is essential to customization function–the team names that user typed in in the first Activity must be transmitted to other two Activities, and configure the names of buttons and textboxes. Besides, the team name should be revised in real time.

In our application, we use Bundle to carry information(called Intent in Android) between Activities. In order to put certain data into Bundle, we must do some encryption first. Briefly speaking, establish a class to save the information(in this application is the team name). After configuration of the class, each team name can be put in an object of this class, and can be transmitted to other Activities by Bundle. Part of source codes are shown below, only team 1 is given for simplicity.

```
package org.crazyt.model;
import java.io.Serializable;
public class Person implements Serializable
{
        private static final long serialVersionUID = 1L;

        private String team1;
```

```
        public Person()
        {
        }

        public Person(String team1)
        {
                this.team1 = team1;
        }

        public String getName1()
        {
                return team1;
        }
        public void setName1(String name1)
        {
                this.team1 = name1;
        }
}

public void onCreate(Bundle savedInstanceState)
{
                super.onCreate(savedInstanceState);
                setContentView(R.layout.result);
                TextView name1 = (TextView)findViewById(R.id.name1);

                Intent intent = getIntent();
                Bundle data = intent.getExtras();
                Person p = (Person)data.getSerializable("person");
                name1.setText("Team␣1:␣" + p.getName1() + "␣");

                final Chronometer ch = (Chronometer)findViewById(R.
                   id.test);
                Button start = (Button)findViewById(R.id.start);
                Button reset = (Button)findViewById(R.id.reset);

                Button st11 = (Button)findViewById(R.id.st11);
                Button st12 = (Button)findViewById(R.id.st12);
                Button st13 = (Button)findViewById(R.id.st13);
                Button st14 = (Button)findViewById(R.id.st14);

                st11.setText("␣" + p.getName1() + "␣1␣");
                st12.setText("␣" + p.getName1() + "␣2␣");
                st13.setText("␣" + p.getName1() + "␣3␣");
                st14.setText("␣" + p.getName1() + "␣4␣");

                final TextView s11 = (TextView)findViewById(R.id.s11
                   );
                final TextView s12 = (TextView)findViewById(R.id.s12
                   );
                final TextView s13 = (TextView)findViewById(R.id.s13
                   );
                final TextView s14 = (TextView)findViewById(R.id.s14
                   );
}
```

3.3 Synchronization

To meet the specific demands of Smart Stopwatch, we finally choose WLAN direct to realize the synchronization function. WLAN direct is a new method to build connections in local area. This method is first called WLAN P2P, for a terminal which support WLAN service can invite other terminal to join in the WLAN. But the data can share by all the group.

As we know, there are several mature method to establish connection between two mobile terminals, such as Bluetooth, mobile hotspot and cloud server. Among these methods, Why we choose the WLAN direct to achieve the synchronization function? Here we make a comparison between WLAN direct and Bluetooth.

	WLAN Direct	**Bluetooth**
Category	WLAN	WPAN
Cover range	$<$ 100m	$<$ 10m
Transmission rate	100Mbps	1Mbps
Service	Multiple terminals	P2P

From the table above, we can see that WLAN direct is a wireless local area network method, it uses RF(radio frequency) to make connection among terminals in the same area, so it is a widely-used technology to transmit data. Bluetooth, at the same time, belongs to WPAN, which is abbreviation of wireless personal area network. PAN is the last procedure in wireless communication, and is seen as the solution of last one meter in data transmission. So we can know that the cover range of Bluetooth is relatively small, only several meters, which is much smaller than the cover range of WLAN direct. Besides, the transmission rate of Bluetooth is rather slow, approximately 1/100 times of that of WLAN direct. What is more, Bluetooth is a P2P(person-to-person) service, the data is only shared by two terminals each time. But WLAN direct can support many terminals to share same data at one time. Here WLAN direct shows plenty of advantages over Bluetooth.

However, for WLAN direct is not a mature technology, the function is still in development and not all terminals support this service. To test our assumption, we have to use the help of a third party application called “ShanChuan”. Figure 3.1 is the UI of“ShanChuan” when we manipulate three terminals to share same data. Using WLAN direct and “ShanChuan”, we can share results between two terminals. Each time a terminal records a time, a same record appears in the other terminal and the two terminals keep synchronous as long as the connection exists. This is very useful when more than one timekeepers are needed in the race.

Figure 3.1: The UI of a third party application “ShanChuan”

Chapter 4

Further discussion

4.1 Flaws to be improved

For this is the first time we use Java language and Android development toolkit, some details in this application are not so fine. For example, we have not considered the screen rotation yet. If the screen is rotated during the application working, all the recorded data will be erased and the timer will restart from zero. And if Smart Stopwatch is running in background for a rather long time, this application will be stopped by Android system and all the data you have recorded will be lost. There is still one detail to be improved: an Undo button should be added in the main UI. For the buttons are close to each other, it is common to press the wrong button. If the corresponding time has not recorded yet, then you can still press the button again to refresh the record time. But if the time has been recorded, then the original record will be lost. An Undo button is the best solution to avoid this kind of misoperation.

4.2 Outlook of Smart Stopwatch

There are many other extensions of this application. First is the totally replacement of traditional stopwatch. Since smart mobilephone is held by everyone today, this application can be applied in many cases. When you are jogging, you can record the time lap by lap and check your current speed by Smart Stopwatch, in order to keep a constant velocity. It also works in the training of athletes, coaches can get the real-time information about athletes without expensive equipments, a smart mobilephone can do all of this.

Smart Stopwatch also can be used for speed detection. Suppose there are two terminals beside a expressway, they are near to each other and are synchronized by WLAN direct. When a car drives by, the difference between the two record times can tell us about the speed. Besides, it may bring the revolution of timing carpet. Combined with

the newly developed NFC(Near Field Communication) technology, Smart Stopwatch can even replace traditional timing carpet with this auto-identification timing carpet.

References

[1] Crazy Android lectures(2nd Edition), Li Gang, Publishing house of electronics industry.

[2] http://www.android-study.com

[3] http://www.developer.android.com