CHAPTER 5

IMPLEMENTATION AND TESTING

This chapter will discuss about the implementation strategy of proposed solution that the author use. The result of proposed solution testing is also described briefly in this chapter.

The expected result and actual result will be written clearly along with the error reason. Testing of the application developed by the author is needed in order to check whether the application can deliver its main functionality or not.

5.1 System Specification

5.1.1 Software

The software needed and used by the author in development process of the proposed solution is described below:

- Microsoft Windows 7 (64 Bit)
- Eclipse (Indigo Release Version)
- Google Chrome 17.0.963.2 (Windows)
- FeedBurner(Web site)

5.1.2 Computer Specification

The hardware used by the author in development process of the application is described below:

- Processor: 2,67 GHz Intel Core i5
- Memory: 4GB 1066 MHz DDR3
- Hard Disk: 640 GB 5400rpm

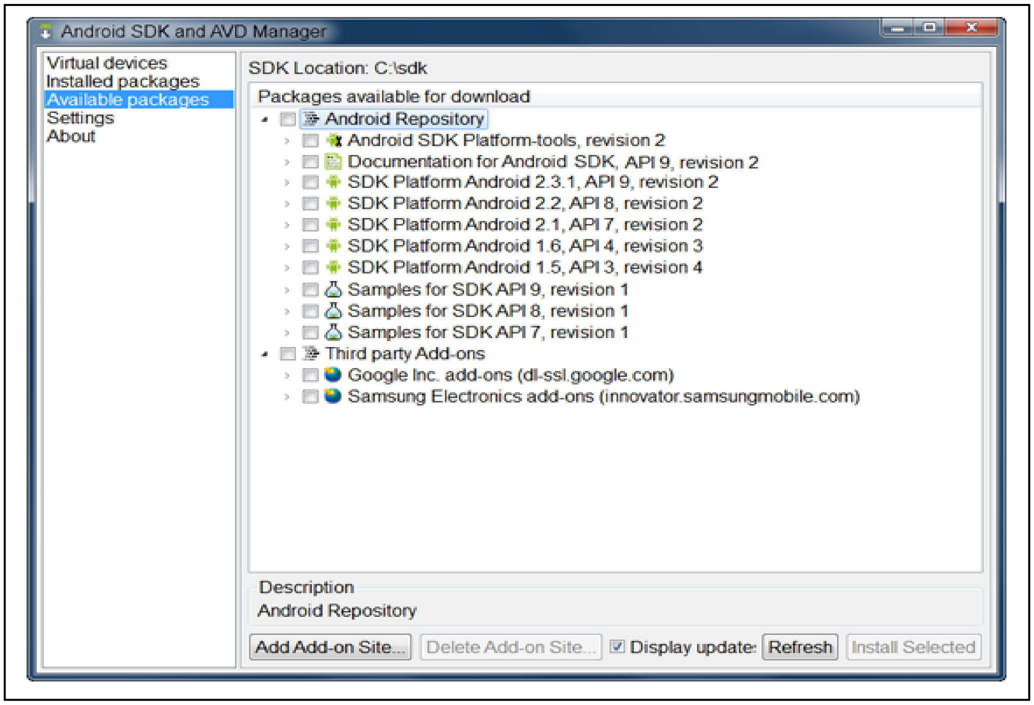
5.2 Operational Procedures

In order to implement the proposed system correctly, the author needs to configure and manage the environment. The steps done by the author are described as follow.

1. Download and install Eclipse IDE
2. Download JDK which is required for compile Java Source Code and for advanced development scenarios
3. Download Android SDK and follow the installation steps

4. Install and run a plugin called Android Development Tools. ADT provides powerful tools and extensions that will ease the process of creating, debugging, and running the Android application.

5. Adding platforms and other components



SDK Location: C:\sdk

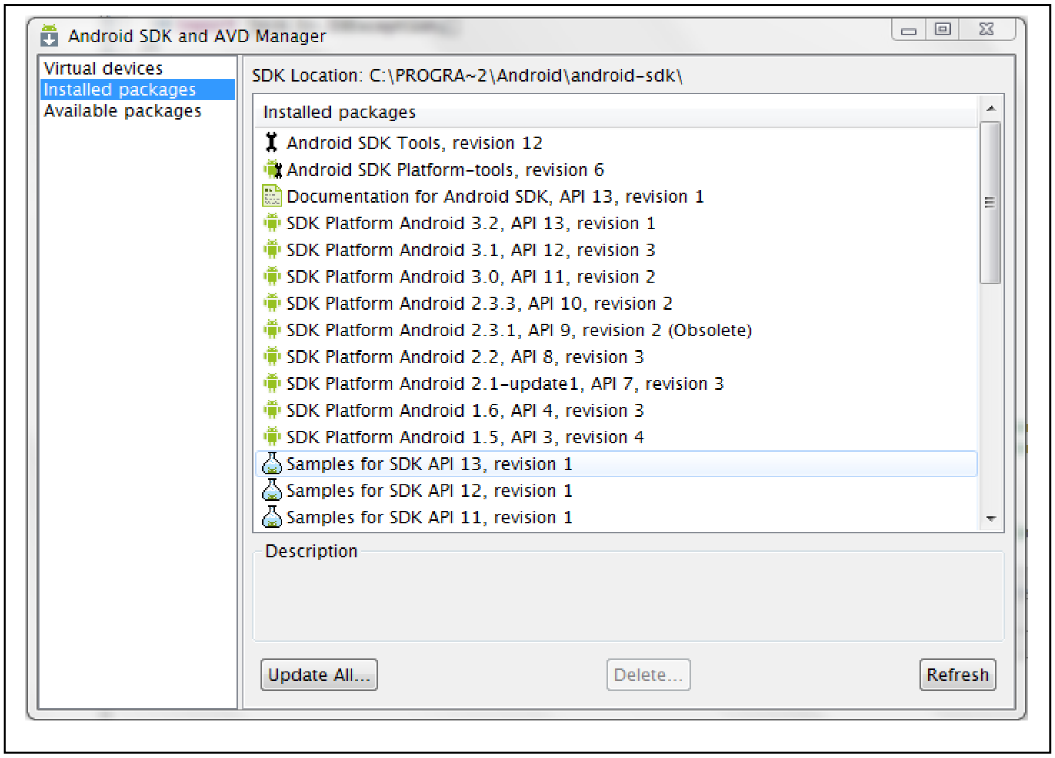
Packages available for download

- □ Android Repository
  - □ Android SDK Platform-tools, revision 2
  - □ Documentation for Android SDK, API 9, revision 2
  - □ SDK Platform Android 2.3.1, API 9, revision 2
  - □ SDK Platform Android 2.2, API 8, revision 2
  - □ SDK Platform Android 2.1, API 7, revision 2
  - □ SDK Platform Android 1.6, API 4, revision 3
  - □ SDK Platform Android 1.5, API 3, revision 4
  - □ Samples for SDK API 9, revision 1
  - □ Samples for SDK API 8, revision 1
  - □ Samples for SDK API 7, revision 1
- □ Third party Add-ons
  - □ Google Inc. add-ons (dl-ssl.google.com)
  - □ Samsung Electronics add-ons (innovator.samsungmobile.com)

Description
Android Repository

Figure 5.1 Available packages in the Android SDK and AVD manager

Figure shown above is the available packages in the Android SDK and AVD Manager. Components available in the Available packages tab must be downloaded first before used.



SDK Location: C:\PROGRA~2\Android\android-sdk\

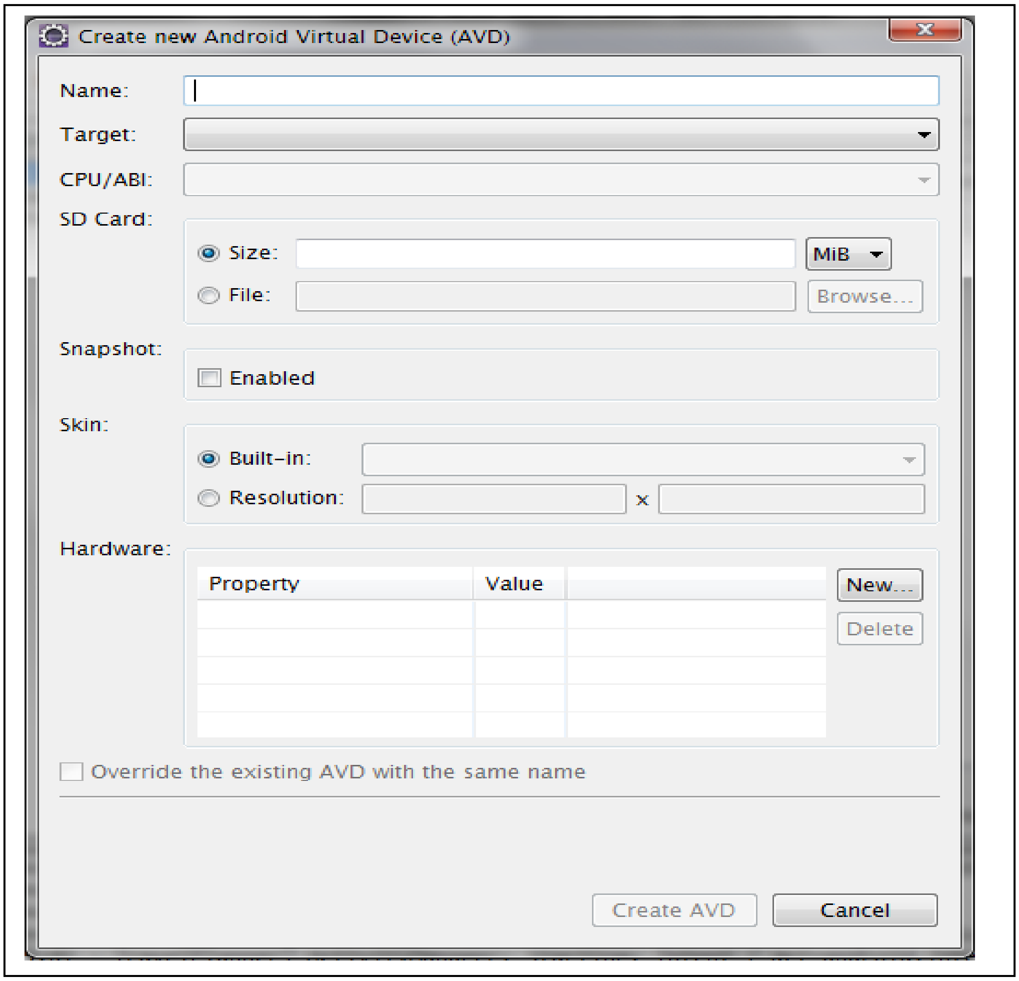
Installed packages

- Android SDK Tools, revision 12
- Android SDK Platform-tools, revision 6
- Documentation for Android SDK, API 13, revision 1
- SDK Platform Android 3.2, API 13, revision 1
- SDK Platform Android 3.1, API 12, revision 3
- SDK Platform Android 3.0, API 11, revision 2
- SDK Platform Android 2.3.3, API 10, revision 2
- SDK Platform Android 2.3.1, API 9, revision 2 (Obsolete)
- SDK Platform Android 2.2, API 8, revision 3
- SDK Platform Android 2.1-update1, API 7, revision 3
- SDK Platform Android 1.6, API 4, revision 3
- SDK Platform Android 1.5, API 3, revision 4
- Samples for SDK API 13, revision 1
- Samples for SDK API 12, revision 1
- Samples for SDK API 11, revision 1

Figure 5.2 Installed packages in the Android SDK and AVD manager

In order to create new AVD in the Eclipse IDE, the author is required to install packages available first before an AVD can be used. AVD stands for Android Virtual Device is a configured emulator which allows users to model the actual device of Android to be emulated by the Android emulator.

6. Create new AVD



Create new Android Virtual Device (AVD)

Name:

Target:

CPU/ABI:

SD Card:

● Size: MiB

○ File: Browse...

Snapshot:

□ Enabled

Skin:

● Built-in:

○ Resolution: x

Hardware:

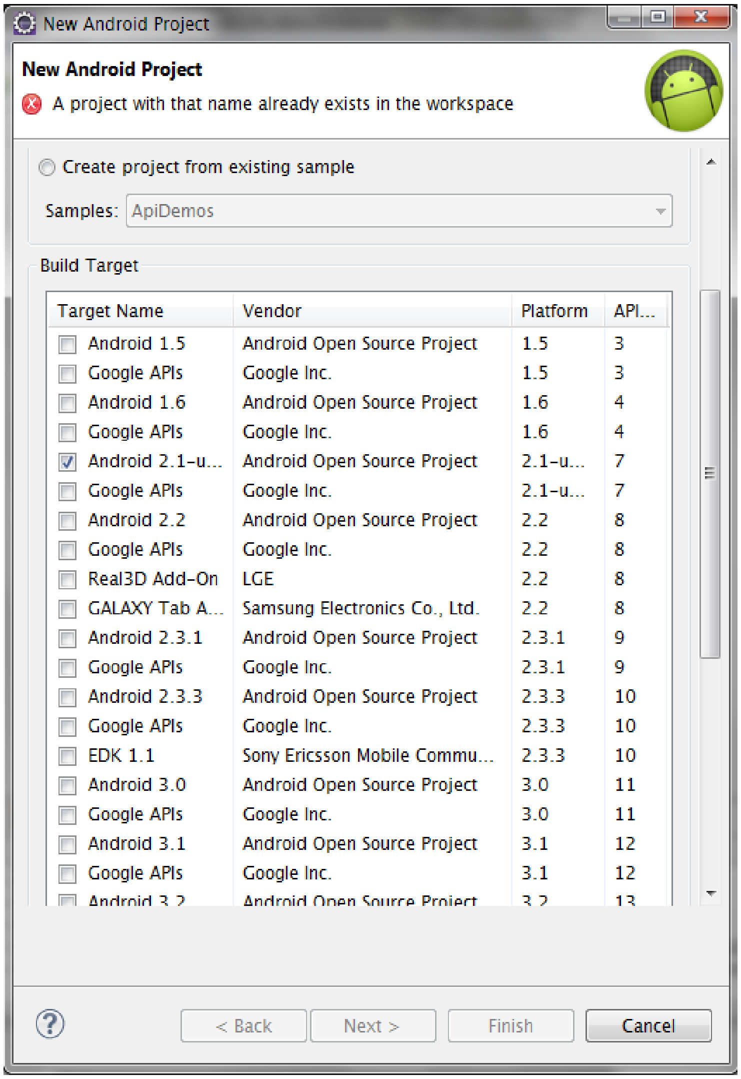
Property	Value	

□ Override the existing AVD with the same name

Figure 5.3 Create new Android Virtual Device before create new project

In developing the proposed solution, the author chooses to use Android 2.1-update 1 as the Virtual Device.

7. After the installation and setting of Eclipse IDE finished, then the author is ready to codes the proposed solution



New Android Project

☒ A project with that name already exists in the workspace

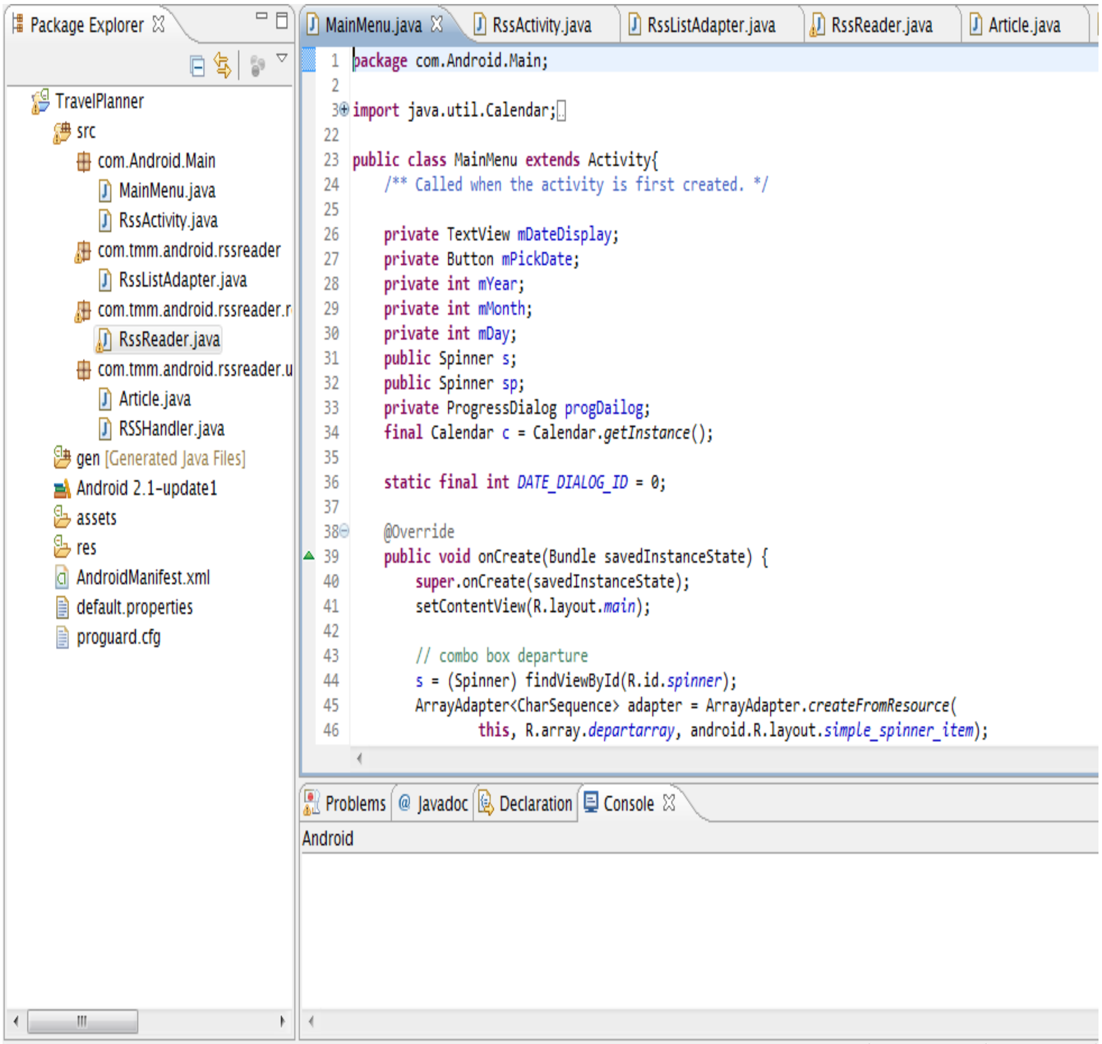
○ Create project from existing sample

Samples: ApiDemos

Build Target

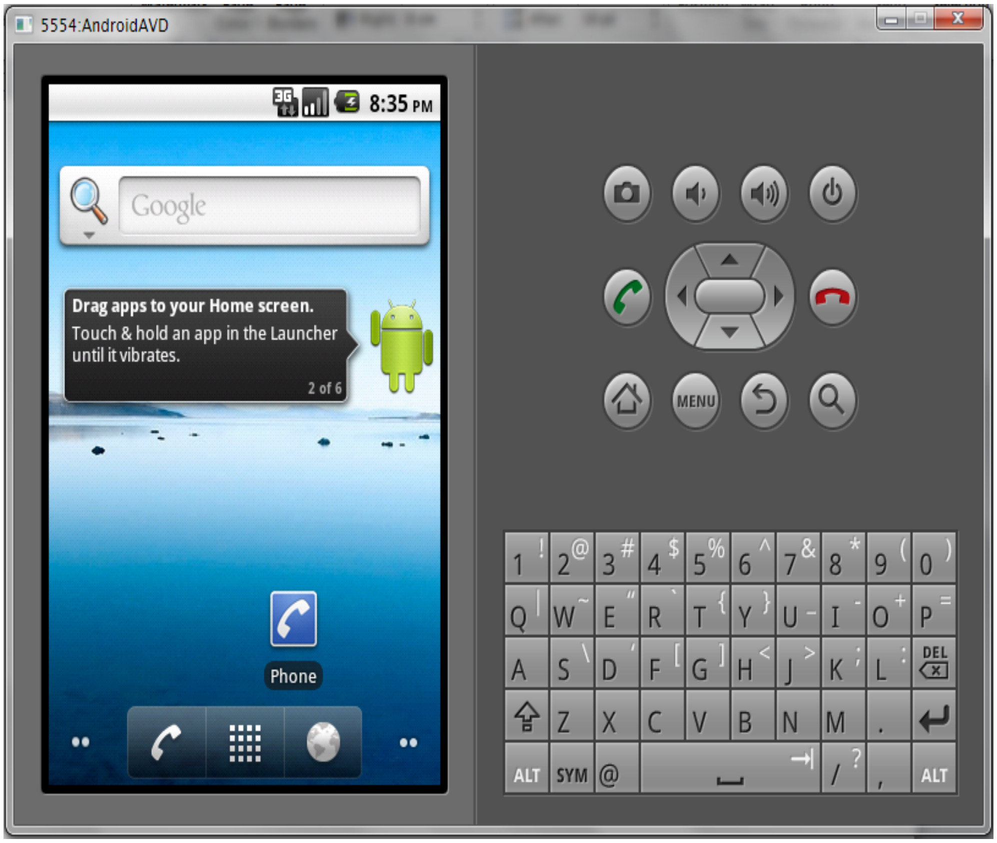
	Target Name	Vendor	Platform	API...
☐	Android 1.5	Android Open Source Project	1.5	3
☐	Google APIs	Google Inc.	1.5	3
☐	Android 1.6	Android Open Source Project	1.6	4
☐	Google APIs	Google Inc.	1.6	4
☑	Android 2.1-u...	Android Open Source Project	2.1-u...	7
☐	Google APIs	Google Inc.	2.1-u...	7
☐	Android 2.2	Android Open Source Project	2.2	8
☐	Google APIs	Google Inc.	2.2	8
☐	Real3D Add-On	LGE	2.2	8
☐	GALAXY Tab A...	Samsung Electronics Co., Ltd.	2.2	8
☐	Android 2.3.1	Android Open Source Project	2.3.1	9
☐	Google APIs	Google Inc.	2.3.1	9
☐	Android 2.3.3	Android Open Source Project	2.3.3	10
☐	Google APIs	Google Inc.	2.3.3	10
☐	EDK 1.1	Sony Ericsson Mobile Commu...	2.3.3	10
☐	Android 3.0	Android Open Source Project	3.0	11
☐	Google APIs	Google Inc.	3.0	11
☐	Android 3.1	Android Open Source Project	3.1	12
☐	Google APIs	Google Inc.	3.1	12
☐	Android 3.2	Android Open Source Project	3.2	13

Figure 5.4 Create new Android project



```
package com.Android.Main;

import java.util.Calendar;

public class MainMenu extends Activity{
    /** Called when the activity is first created. */

    private TextView mDateDisplay;
    private Button mPickDate;
    private int mYear;
    private int mMonth;
    private int mDay;
    public Spinner s;
    public Spinner sp;
    private ProgressDialog progDailog;
    final Calendar c = Calendar.getInstance();

    static final int DATE_DIALOG_ID = 0;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // combo box departure
        s = (Spinner) findViewById(R.id.spinner);
        ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
                this, R.array.departarray, android.R.layout.simple_spinner_item);
```

Figure 5.5 Code developed by author



Drag apps to your Home screen.

Touch & hold an app in the Launcher until it vibrates.

2 of 6

Figure 5.6 Android 2.1 –update 1 appearance

5.3 Implementation Strategy

In this project, the author would like to test whether the application could successfully deliver result of the requested parameters or not. The author will act as a user who would like to request status of flight within Indonesia region. The author will be using Android Virtual Device for requesting flight status from the provider.

However, the feed URL given by the provider must be compressed first by using FeedBurner in order to minimize the page redirect that produced by feed URL given by the provider. Moreover, the test will be about connecting the Android Virtual Device with the service offered by FlightStats. The author will try to request Flight status from Android Virtual Device and shows the response on it.

5.4 Testing Plan

The Feed URL must have been compressed by FeedBurner in order to be used in the testing. FeedBurner is very useful in this program because it could reduce the page redirect produced by the original Feed URL given by FlightStats. In order to make the request valid, FlightStats GUID must be owned first before creating a query or request.

Moreover, the response from the request asked by Author will be shown in the Android Emulator with List View as the type of interface. The reason why Author chooses List View is because it’s simple and easy to be understood.

5.5 Application Testing

Module testing is conducted in order to test whether the application have brought the feature expected or not. This application testing will test the functionality of the prototype.

1. Running the application

Date	9 December 2011
Test Case	The application icon shown in the AVD
Pre-condition	The application's AVD has already set
Steps	1. Run the application from Eclipse IDE
Expected Result:	Application icon should appear in the AVD
Status:	**OK**

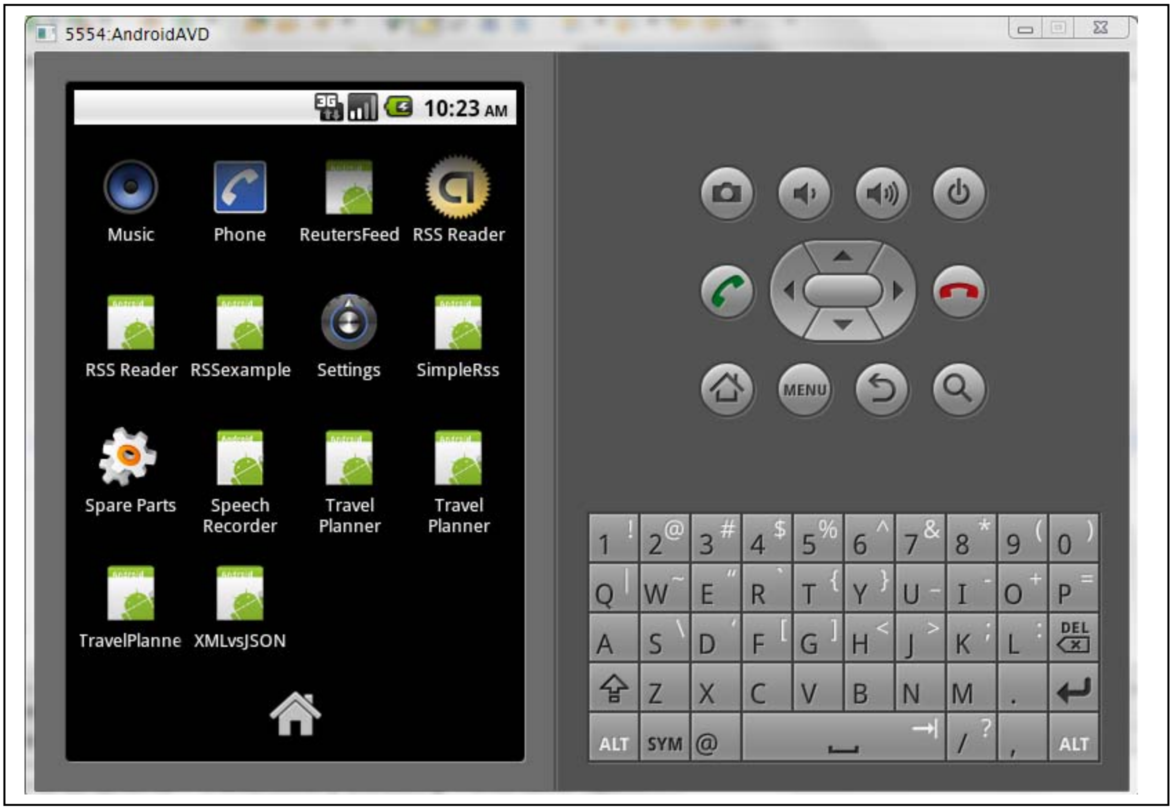
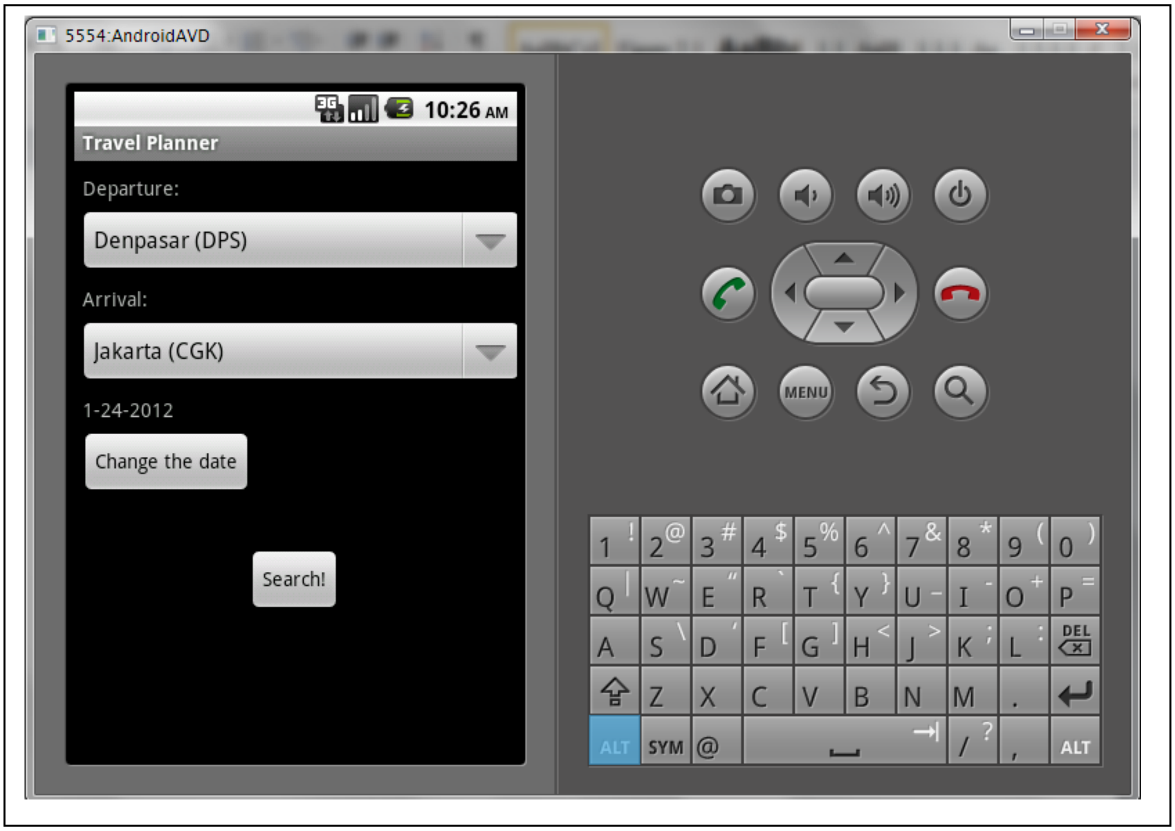


Figure 5.7 Travel Planner icon in the Home Screen

2. Display the interface

Date	9December 2011
Test Case	The application shows the 1st screen
Pre-condition	AVD already running Application’s icon tapped
Steps	1. Run the application from Eclipse IDE
	2. Choose the application’s icon from the menu
Expected Result:	Application 1st screen appear in the AVD
Status:	**OK**



5554:AndroidAVD

Travel Planner

Departure: Denpasar (DPS)

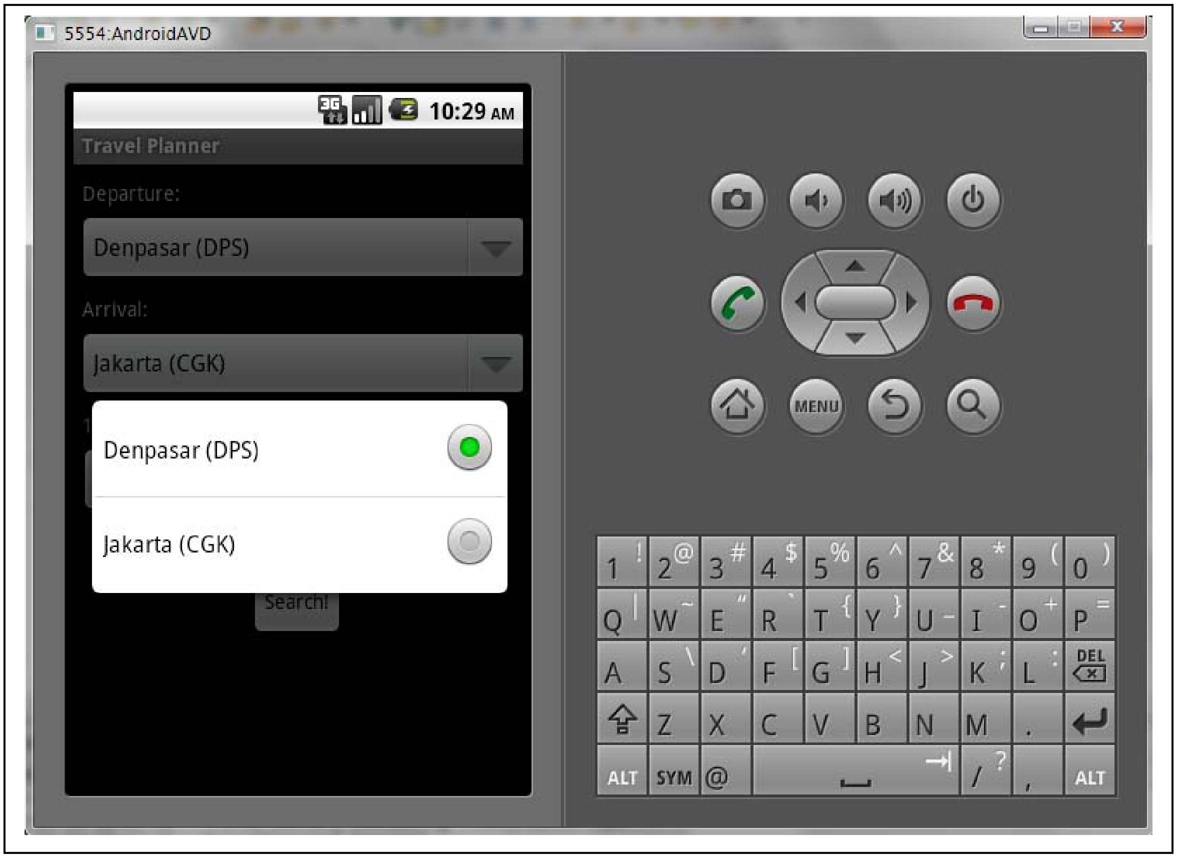
Arrival: Jakarta (CGK)

1-24-2012

Figure 5.8 1st screen of the application

3. Departure dropdown box

Date	9 December 2011
Test Case	Departure Spinner shows items
Pre-condition	AVD already running Application's icon tapped Application interface design shown
Steps	1. Run the application from Eclipse IDE
	2. Choose the application's icon from the menu
Expected Result:	Dropdown box showing departure point appear when arrow clicked
Status:	**OK**



Departure: Denpasar (DPS)

Arrival: Jakarta (CGK)

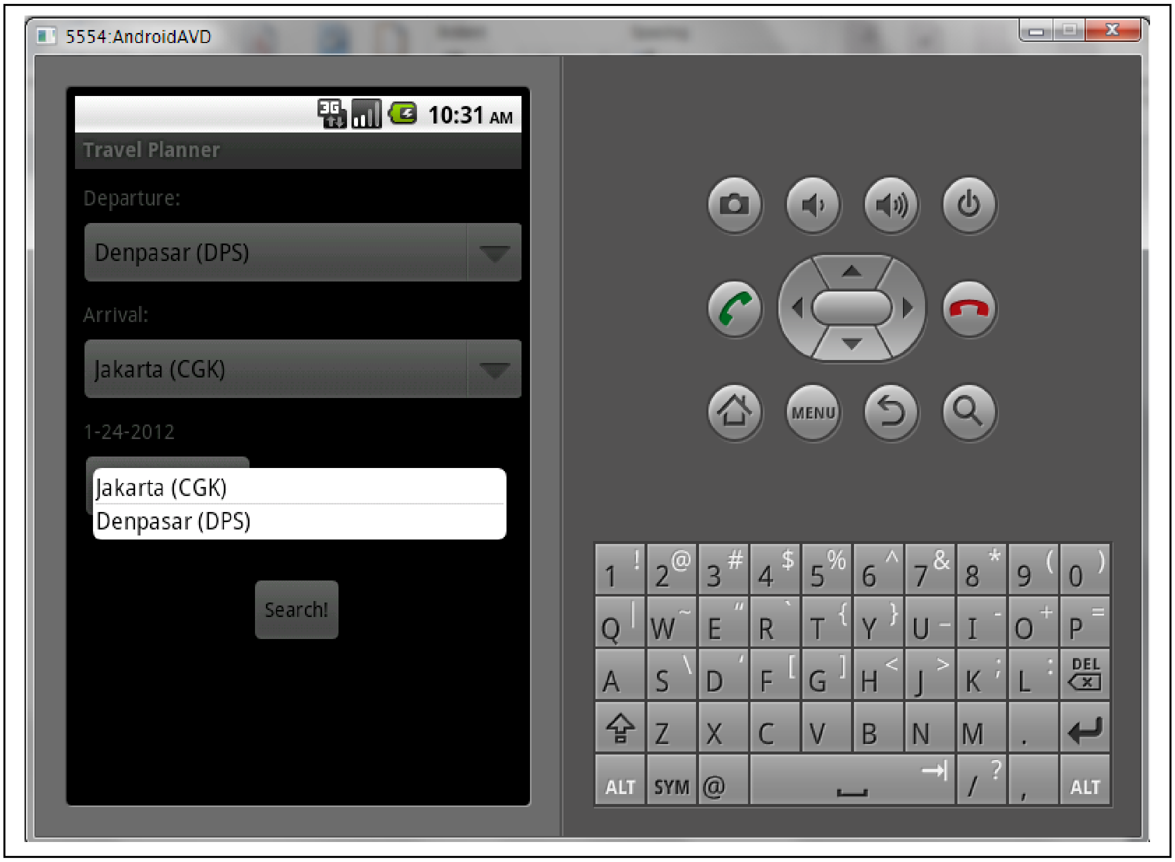
Denpasar (DPS) ●

Jakarta (CGK) ○

Figure 5.9 Departure Dropdown Box

4. Arrival dropdown box

Date	9 December 2011
Test Case	Arrival Spinner shows items
Pre-condition	AVD already running Application’s icon tapped Application interface design shown
Steps	1. Run the application from Eclipse IDE
	2. Choose the application’s icon from the menu
Expected Result:	Dropdown box showing arrival point appear when arrow clicked
Status:	**OK**



5554:AndroidAVD

Travel Planner

Departure:

Denpasar (DPS)

Arrival:

Jakarta (CGK)

1-24-2012

Jakarta (CGK)
Denpasar (DPS)

Figure 5.10 Arrival Dropdown Box

5. Date Dialog

Date	9 December 2011
Test Case	Check Date Dialog successfully implemented
Pre-condition	AVD already running Application’s icon tapped Application interface design shown
Steps	1. Run the application from Eclipse IDE
	2. Choose the application’s icon from the menu
	3. Click “Change Date” button
Expected Result:	Date Dialog appear, asking user to choose a date
Status:	**OK**

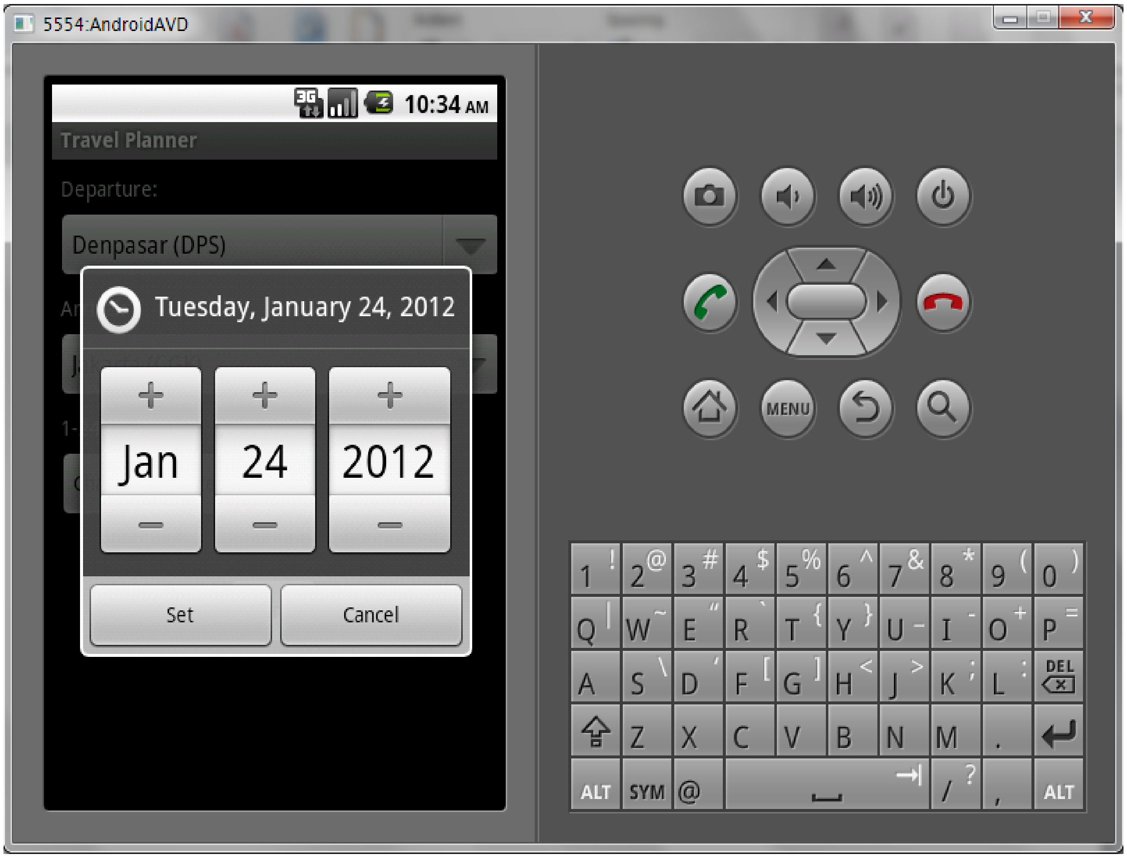
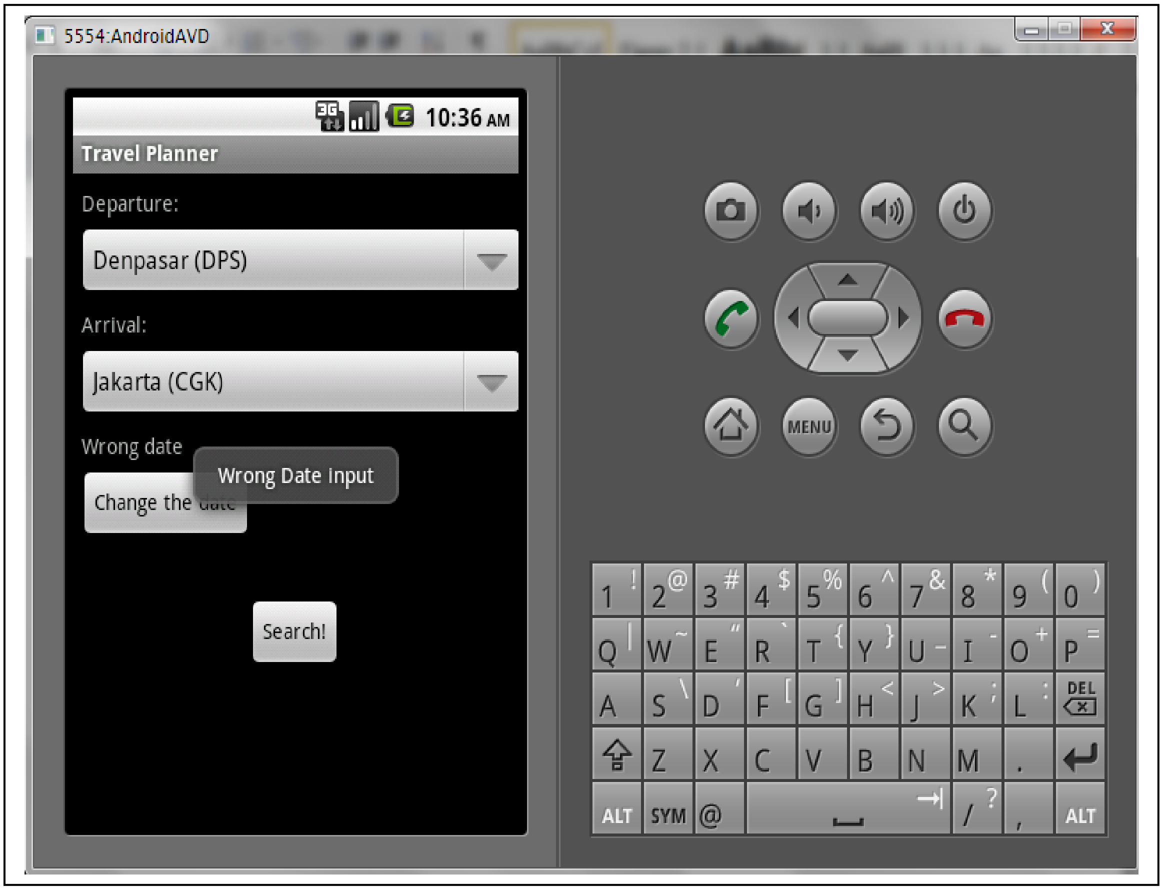


Figure 5.11 Date Dialog

6. Date Dialog error

Date	9 December 2011
Test Case	Check Date dialog error
Pre-condition	AVD already running Application’s icon tapped Application interface design shown Date Dialog appear
Steps	1. Run the application from Eclipse IDE
	2. Choose the application’s icon from the menu
	3. “Change Date” button clicked
	4. Choose yesterday’s date for the date
Expected Result:	Notification appear telling wrong date
Status:	**OK**



5554:AndroidAVD

Travel Planner

Departure: Denpasar (DPS)

Arrival: Jakarta (CGK)

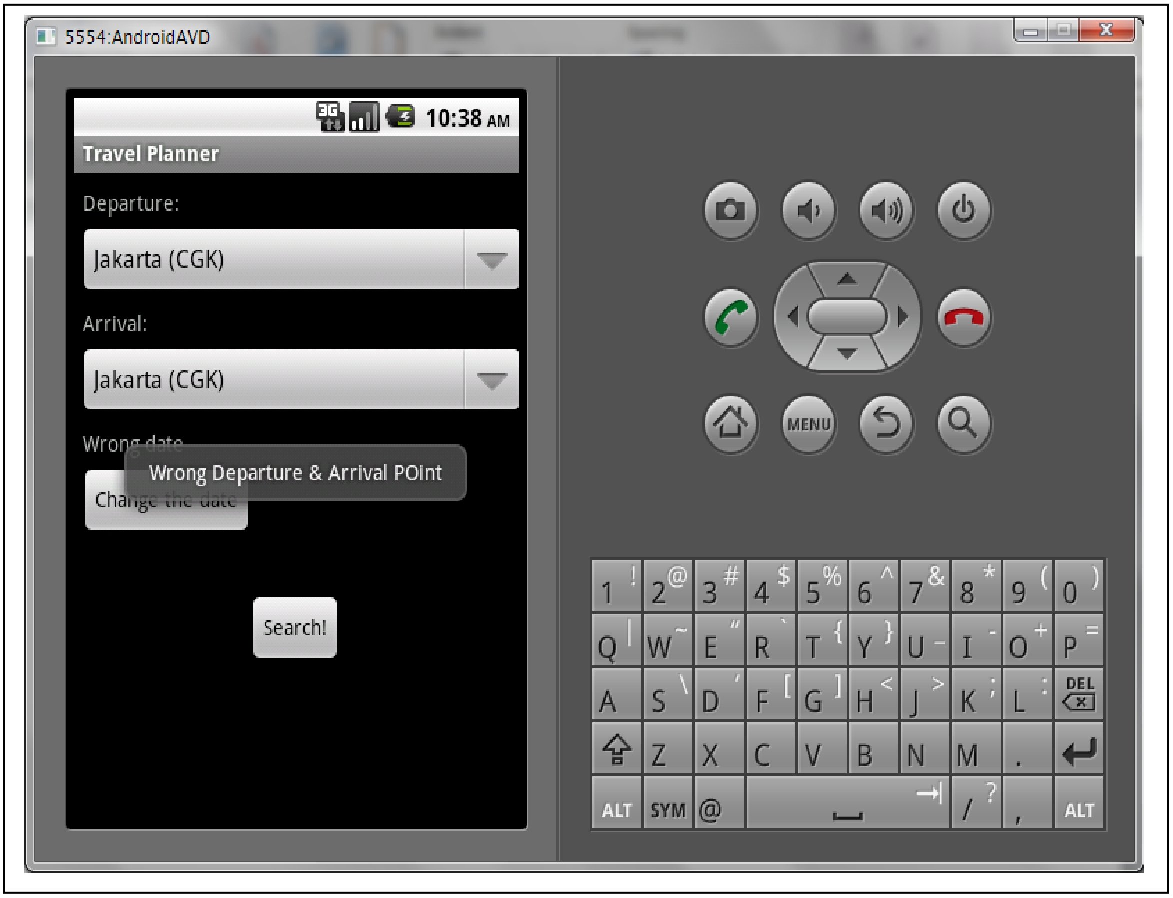
Wrong date

Wrong Date Input

Figure 5.12 Wrong Date Input Notification

7. Departure & Arrival Error Notification

Date	9 December 2011
Test Case	Check Departure and Arrival Point error
Pre-condition	AVD already running Application’s icon tapped Application interface design shown Departure and arrival point picked
Steps	1. Run the application from Eclipse IDE
	2. Choose the application’s icon from the menu
	3. Choose departure & arrival point
	4. “Search” button clicked
Expected Result:	Notification telling the Departure & Arrival Point contains same location
Status:	**OK**



Travel Planner

Departure: Jakarta (CGK)

Arrival: Jakarta (CGK)

Wrong date

Wrong Departure & Arrival POint

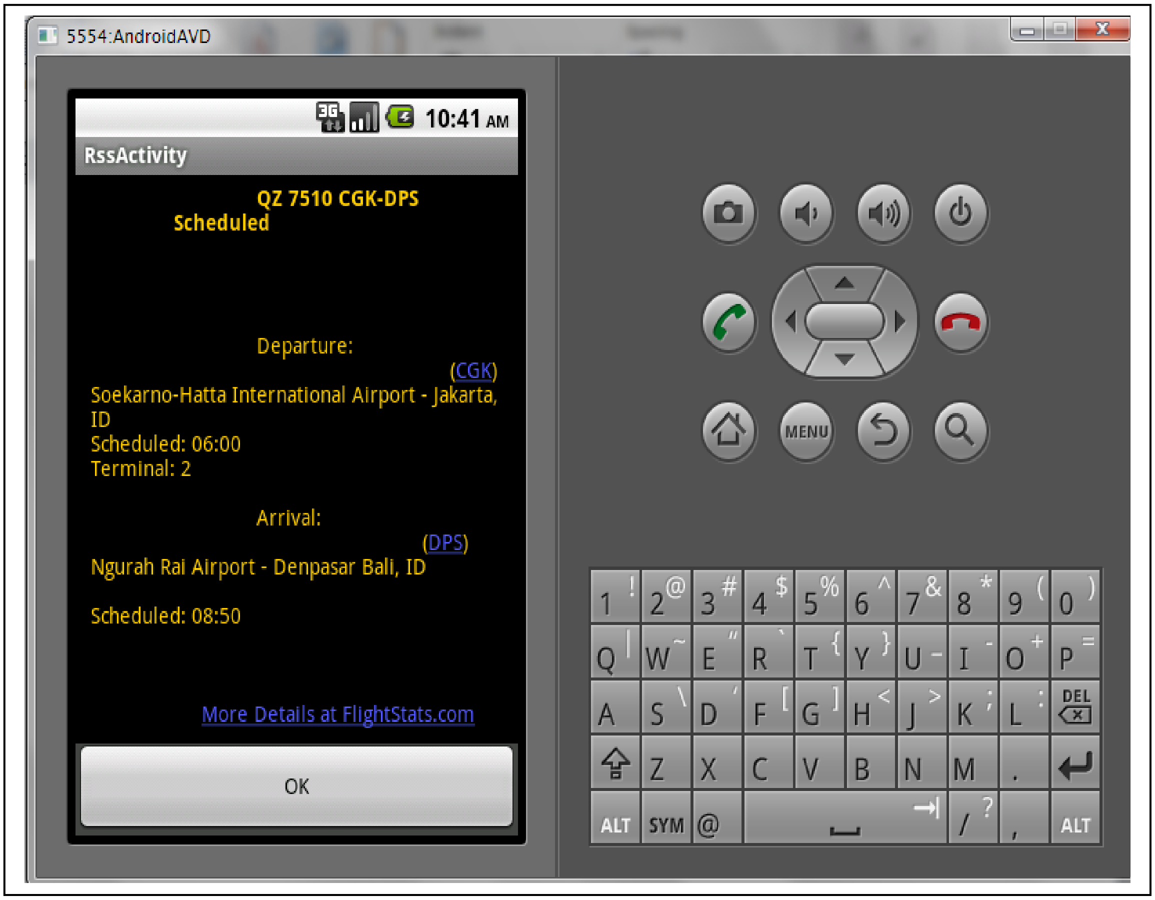
Figure 5.13 Wrong Departure & Arrival Point Notification

8. Data Passing

Date	9 December 2011
Test Case	Data passing between activity
Pre-condition	AVD already running Application's icon tapped Application interface design shown Required information inserted
Steps	1. Run the application from Eclipse IDE
	2. Choose the application's icon from the menu
	3. Set departure & arrival point and departure date
	4. "Search" button clicked
Expected Result:	Redirect to 2nd activity and show data passed
Status:	**OK**

9. Feed Request

Date	12 January 2012
Test Case	Request Feed
Pre-condition	AVD already running Application’s icon tapped Application interface design shown Required information inserted
Steps	1. Run the application from Eclipse IDE
	2. Choose the application’s icon from the menu
	3. Set departure & arrival point and departure date
	4. “Search” button clicked
Expected Result:	Redirect to 2nd activity and show Flight Status
Status:	**OK**



5554:AndroidAVD

10:41 AM

RssActivity

QZ 7510 CGK-DPS
Scheduled

Departure:
(CGK)
Soekarno-Hatta International Airport - Jakarta, ID
Scheduled: 06:00
Terminal: 2

Arrival:
(DPS)
Ngurah Rai Airport - Denpasar Bali, ID

Scheduled: 08:50

More Details at FlightStats.com

OK

Figure 5.14 Flight Status on 2nd screen

10. Redirect to Main Menu

Date	9 December 2011
Test Case	Return to 1st screen
Pre-condition	AVD already running Application's icon tapped Application interface design shown Required information inserted 2nd screen displayed
Steps	1. Run the application from Eclipse IDE
	2. Choose the application's icon from the menu
	3. Set departure & arrival point and departure date
	4. "Search" button clicked
	5. Flight status displayed in 2nd screen
	6. "Back" button clicked
Expected Result:	Redirect back to 1st screen
Status:	**OK**