COMP4521 Embedded Systems Software

Lab 3: Creating Activities For Menu Items and Parsing XML Files

Introduction

We have learnt how to create a simple menu in the last lab. Today, we will learn to add different activities for each of the menu item. After that, we will learn how to extract information from XML files in Android.

Objectives

- Define activities for menu items.
- Parse XML files.
- Use Table Layout
- Programmatically add rows to Table Layout
- Use of “Linkify” to automatically create clickable links from text

Define Activities For Menu Items.

1. Download CourseInfo4521.zip from the course website, and extract it to the workspace of Eclipse, e.g., D:\temp. The zip file contains the final result of Lab 2.
2. Locate the SDK of Android for Eclipse. In Eclipse folder, run eclipse.bat, set your workspace to D:\Temp, click **Window**-> **Preferences**-> **Android**, and choose the SDK location (must be D:\comp4521\android-sdk-windows) as where you have put the Android SDK.
3. Create an Android Virtual Device (AVD).
4. **Import the existing project**: In Eclipse, click **File**-> **Import** -> **General -> Existing Projects into Workspace**. Find the location of the root directory of CourseInfo4521, and press **Finish**. Make sure that after importing the project, Eclipse gives no warnings.
5. Define activities for each of the menu item. Open menu.java. Recall that in last lab, we used Toast to pop out a message box when clicking on a menu item.

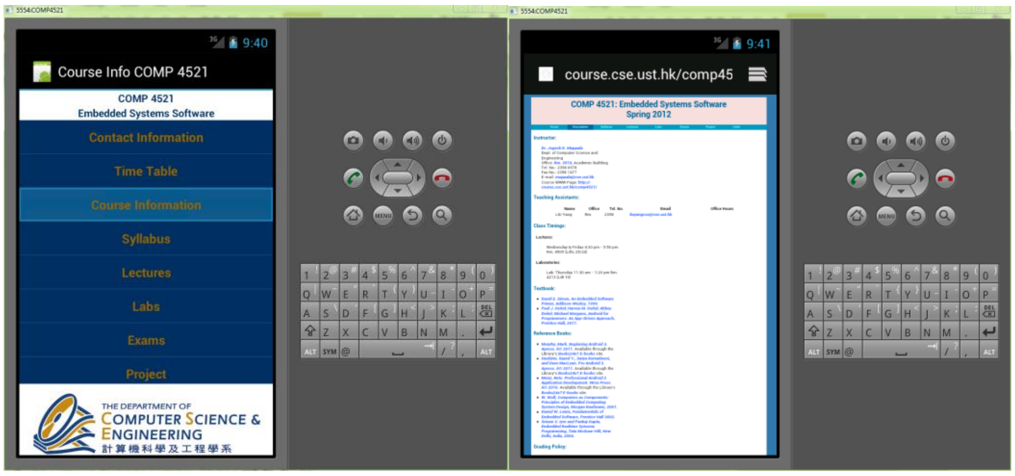
```
// When clicked, show a toast with the TextView text
Toast.makeText(getApplicationContext(), ((TextView) itemClicked).getText(),
        Toast.LENGTH_SHORT).show();
```

6. Now we replace this activity with different activities for each of the menu items:

```
        TextView textview = (TextView) itemClicked;
          String strText = textview.getText().toString();

          switch(position){
              case 0:
                  // When clicked, show a toast with the TextView text
                  Toast.makeText(getApplicationContext(), strText,
                      Toast.LENGTH_SHORT).show();
            break;
              case 1:
                  // When clicked, show a toast with the TextView text
                  Toast.makeText(getApplicationContext(), strText,
                      Toast.LENGTH_SHORT).show();
            break;
            case 2:
               startActivity(new Intent(Intent.ACTION_VIEW,
Uri.parse("http://course.cse.ust.hk/comp4521/Description.html")));
              break;
              case 3:
               startActivity(new Intent(Intent.ACTION_VIEW,
Uri.parse("http://course.cse.ust.hk/comp4521/Syllabus.html")));
              break;

              case 4:
               startActivity(new Intent(Intent.ACTION_VIEW,
Uri.parse("http://course.cse.ust.hk/comp4521/Lectures.html")));
              break;
              case 5:
               startActivity(new Intent(Intent.ACTION_VIEW,
Uri.parse("http://course.cse.ust.hk/comp4521/Labs.html")));
              break;
              case 6:
               startActivity(new Intent(Intent.ACTION_VIEW,
Uri.parse("http://course.cse.ust.hk/comp4521/Exams.html")));
              break;
              case 7:
               startActivity(new Intent(Intent.ACTION_VIEW,
Uri.parse("http://course.cse.ust.hk/comp4521/Project.html")));
              break;
              case 8:
               startActivity(new Intent(Intent.ACTION_VIEW,
Uri.parse("http://course.cse.ust.hk/comp4521/Links.html")));
              break;


            default:
                  // When clicked, show a toast with the TextView text
                  Toast.makeText(getApplicationContext(), strText,
                      Toast.LENGTH_SHORT).show();
            break;

          }
```

This switch statement selection defines activity for each of the menu item. Notice that the numbers 0~8 represents the position of the items of the string array defined in strings.xml. The items are automatically sequentially numbered when the string array is created in the application.

7. Run the application. Go to the menu screen. Except the first two items, if you click on any of the other items, it will open the corresponding webpage in the browser application. Press the return button of the emulator to go back to the menu screen and test other menu items.



EXTRACTING INFORMATION FROM XML FILES

1. Add necessary strings for the application. Open **/res/values/strings.xml**, add four strings:

```
<string name="instructor">Instructor</string>
<string name="tas">Teaching Assistant(s)</string>
<string name="lecture">Lectures</string>
<string name="lab">Lab Sessions</string>
```

2. Add two new layouts for the application. The first is contacts.xml, which displays contact information:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/ustblue"
    >
  <LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1">

    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/CourseCode"
    android:gravity="center_vertical|center_horizontal"
    android:textColor="@color/ustblue"
    android:background="@android:color/white"
```

```
        android:textSize="6pt"
        android:textStyle="bold"/>

        <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/CourseTitle"
        android:gravity="center_vertical|center_horizontal"
        android:background="@android:color/white"
        android:textColor="@color/ustblue"
        android:textSize="6pt"
        android:textStyle="bold"/>

        <ScrollView
        android:id="@+id/ScrollViewContacts"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:scrollbars="vertical">
      <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1">

        <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/instructor"
        android:background="@color/ustblue"
        android:textColor="@color/ustgold"
        android:textSize="6pt"
        android:textStyle="bold"/>

        <TableLayout
        android:id="@+id/TableLayout_Instructor"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:stretchColumns="*">
        </TableLayout>

        <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/tas"
        android:background="@color/ustblue"
        android:textColor="@color/ustgold"
        android:textSize="6pt"
        android:textStyle="bold"/>

        <TableLayout
        android:id="@+id/TableLayout_TA"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:stretchColumns="*">
        </TableLayout>
        </LinearLayout>
        </ScrollView>
    </LinearLayout>

       <ImageView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical|center_horizontal"
        android:src="@drawable/cse_logo"
        android:maxHeight="10px"
        android:background="@android:color/white"
        />
    </LinearLayout>
```

Notice the usage of scrollview and TableLayout above.

3. The second is timetable.xml, which displays the time table:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:background="@color/ustblue"
      >
   <LinearLayout
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1">

      <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/CourseCode"
      android:gravity="center_vertical|center_horizontal"
      android:textColor="@color/ustblue"
      android:background="@android:color/white"
      android:textSize="6pt"
      android:textStyle="bold"/>

      <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/CourseTitle"
      android:gravity="center_vertical|center_horizontal"
      android:background="@android:color/white"
      android:textColor="@color/ustblue"
      android:textSize="6pt"
      android:textStyle="bold"/>

      <ScrollView
      android:id="@+id/ScrollViewContacts"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:scrollbars="vertical">
   <LinearLayout
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1">

      <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/lecture"
      android:background="@color/ustblue"
      android:textColor="@color/ustgold"
      android:textSize="6pt"
      android:textStyle="bold"/>

      <TableLayout
      android:id="@+id/TableLayout_Lecture"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:stretchColumns="*">
      </TableLayout>

      <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
```

```
        android:text="@string/lab"
        android:background="@color/ustblue"
        android:textColor="@color/ustgold"
        android:textSize="6pt"
        android:textStyle="bold"/>

        <TableLayout
        android:id="@+id/TableLayout_Lab"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:stretchColumns="*">
        </TableLayout>
        </LinearLayout>
        </ScrollView>
</LinearLayout>

      <ImageView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical|center_horizontal"
        android:src="@drawable/cse_logo"
        android:maxHeight="10px"
        android:background="@android:color/white"
        />
</LinearLayout>
```

4. Create two XML files containing information that will be extracted later. In this special case, the XML files are included in the application itself. We can also make the application fetch XML files through Internet connection. In **/res**, create a subfolder named xml, and create two XML files in this **/res/xml**. The contents of contactinfo.xml are:

```
<?xml version="1.0" encoding="UTF-8"?>
<contact>
<instructor
   name="Jogesh K. Muppala"
   office="Rm. 3510"
   tel="2358 6978"
   email="muppala@cse.ust.hk"
   web="http://www.cse.ust.hk/~muppala/">
</instructor>
<assistant
   name="Liu Yang"
   office="Rm. 4205"
   tel="2358 6978"
   email="liuyangcse@cse.ust.hk">
</assistant>
<assistant
   name="Liu Yang"
   office="Rm. 4205"
   tel="2358 6978"
   email="liuyangcse@cse.ust.hk">
</assistant>
</contact>
```

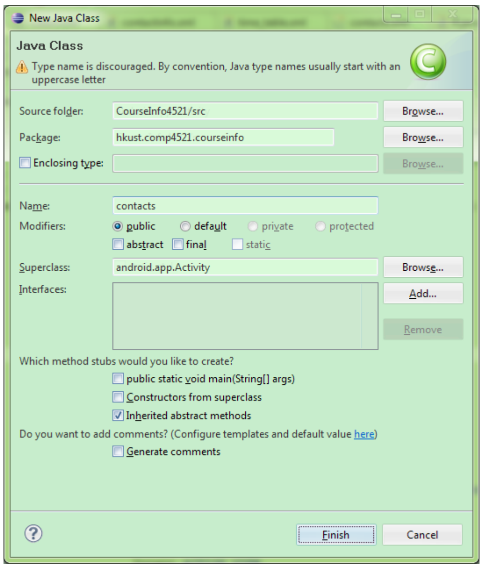
5. The contents of time_table.xml are:

```
   <?xml version="1.0" encoding="UTF-8"?>
<timetable>
<lecture
   days="Wednesday and Friday"
   room="Rm. 4505"
   time="4:30 pm - 5:50 pm">
</lecture>
<lab
   days="Thursday"
   room="Rm. 4213"
   time="11:30 am - 1:20 pm">
</lab>
   </timetable>
```

6. Define two new Activities. In menu.java, replace the activities of case 0 and case 1 with new functions:

```
  case 0:
        startActivity(new Intent(menu.this,contacts.class));
break;

  case 1:
        startActivity(new Intent(menu.this,time_table.class));
break;
```

7. Create the source code for contacts and time table Activities. In **/src/hkust.comp4521.courseinfo**, create a new public class named *contacts*, with android.app.Activity as its superclass.



New Java Class

Java Class

Type name is discouraged. By convention, Java type names usually start with an uppercase letter

Source folder: CourseInfo4521/src

Package: hkust.comp4521.courseinfo

[ ] Enclosing type:

Name: contacts

Modifiers: [✓ public] default private protected

[ ] abstract [ ] final [ ] static

Superclass: android.app.Activity

Interfaces:

Which method stubs would you like to create?

[ ] public static void main(String[] args)

[ ] Constructors from superclass

[✓] Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

[ ] Generate comments

Modify the function body of contacts with following code. Please read the comments to understand the meaning of each step.

```
  // Add a new string
public static final String DEBUG_TAG = "Contacts Log";

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.contacts);

    //Get pointers to the two table layouts in the contacts.xml file
    TableLayout instructorTable = (TableLayout)
findViewById(R.id.TableLayout_Instructor);
```

```
        TableLayout taTable = (TableLayout) findViewById(R.id.TableLayout_TA);

        // Open a XML resource parser to parse the contactinfo.xml file
        XmlResourceParser contactinfo = getResources().getXml(R.xml.contactinfo);

        // Now construct the information for the instructor and TA from the parsed XML file
        try {
            // process the contacts xml file to set up the information on the activity screen
            processcontacts(instructorTable, taTable, contactinfo);
        } catch (Exception e) {
            Log.e(DEBUG_TAG, "Failed to load Contacts", e);
        }
    }


    /**
     * Churn through an XML score information and populate a {@code TableLayout}
     *
     * @param instructorTable
     *           The {@code TableLayout} to populate
     * @param taTable
     *           The {@code TableLayout} to populate
     *
     * @param contact
     *           A standard {@code XmlResourceParser} containing the scores
     * @throws XmlPullParserException
     *            Thrown on XML errors
     * @throws IOException
     *            Thrown on IO errors reading the XML
     */
    private void processcontacts(final TableLayout instructorTable,
            final TableLayout taTable,
            XmlResourceParser contact) throws XmlPullParserException,
            IOException {
        int eventType = -1;
        boolean bFoundContacts = false;
        // Find records from XML
        while (eventType != XmlResourceParser.END_DOCUMENT) {
            if (eventType == XmlResourceParser.START_TAG) {
                // Get the name of the tag (eg contact, instructor or assistant)
                String strName = contact.getName();
                if (strName.equals("instructor")) {
                    bFoundContacts = true;
                    String name = contact.getAttributeValue(null, "name");
                    String office = contact.getAttributeValue(null, "office");
                    String tel = contact.getAttributeValue(null, "tel");
                    String email = contact.getAttributeValue(null, "email");
                    String web = contact.getAttributeValue(null, "web");
                    insertContactRow(instructorTable, "     " + name, -1);
                    insertContactRow(instructorTable, "     Office: " + office, -1);
                    insertContactRow(instructorTable, "     Tel: " + tel,
Linkify.PHONE_NUMBERS);
                    insertContactRow(instructorTable, "     Email: " + email,
Linkify.EMAIL_ADDRESSES);
                    insertContactRow(instructorTable, "     Web: " + web, Linkify.WEB_URLS);
                    insertContactRow(instructorTable, "     ", -1);

                }
                if (strName.equals("assistant")) {
                    bFoundContacts = true;
                    String name = contact.getAttributeValue(null, "name");
                    String office = contact.getAttributeValue(null, "office");
                    String tel = contact.getAttributeValue(null, "tel");
                    String email = contact.getAttributeValue(null, "email");
                    insertContactRow(taTable, "     " + name, -1);
                    insertContactRow(taTable, "     Office: " + office, -1);
                    insertContactRow(taTable, "     Tel: " + tel, Linkify.PHONE_NUMBERS);
                    insertContactRow(taTable, "     Email: " + email,
Linkify.EMAIL_ADDRESSES);
```

```
                insertContactRow(taTable, "     ", -1);
            }
        }
        eventType = contact.next();
    }
    // Handle no records available
    if (bFoundContacts == false) {
        final TableRow newRow = new TableRow(this);
        TextView noResults = new TextView(this);
        newRow.addView(noResults);
        instructorTable.addView(newRow);
    }
}

/**
 * {@code insertContactRow()} helper method -- Inserts a new contact information row 
{@code
 * TableRow} in the {@code TableLayout}
 *
 * @param contactTable
 *          The {@code TableLayout} to add the contact information to
 * @param strValue
 *          The value of text string to be inserted into the row
 * @param mask
 *          specifies what regex I need to look for in the string in order to Linkify 
it. mask <= 0 implies no need to Linkify.
 */
private void insertContactRow(final TableLayout contactTable, String strValue, int 
mask) {
   // create a new table row and populate it
    final TableRow newRow = new TableRow(this);
    TextView textView = new TextView(this);
    textView.setText(strValue);
    if (mask > 0)
      Linkify.addLinks(textView, mask);
    newRow.addView(textView);
    contactTable.addView(newRow);
}
```

8. Similarly, create source code for time_table.java. You can also make a copy of contacts.java, and replace the function body with following code:

```
public static final String DEBUG_TAG = "Time Table Log";

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.timetable);

    //Get pointers to the two table layouts in the contacts.xml file
    TableLayout lectureTable = (TableLayout) findViewById(R.id.TableLayout_Lecture);
    TableLayout labTable = (TableLayout) findViewById(R.id.TableLayout_Lab);

    // Open a XML resource parser to parse the contactinfo.xml file
    XmlResourceParser timetableinfo = getResources().getXml(R.xml.time_table);
    try {
        processtimetable(lectureTable, labTable, timetableinfo);
    } catch (Exception e) {
        Log.e(DEBUG_TAG, "Failed to load Time Table", e);
    }
}

/**
```

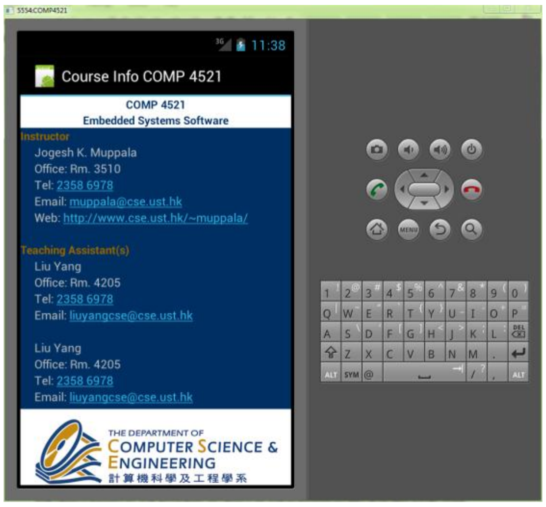
```java
 * Churn through an XML score information and populate a {@code TableLayout}
 *
 * @param lectureTable
 *          The {@code TableLayout} to populate
 * @param labTable
 *          The {@code TableLayout} to populate
 * @param timetable
 *          A standard {@code XmlResourceParser} containing the scores
 * @throws XmlPullParserException
 *           Thrown on XML errors
 * @throws IOException
 *           Thrown on IO errors reading the XML
 */
private void processtimetable(final TableLayout lectureTable,
       final TableLayout labTable,
       XmlResourceParser timetable) throws XmlPullParserException,
       IOException {
    int eventType = -1;
    boolean bFoundTimeTable = false;
    // Find records from XML
    while (eventType != XmlResourceParser.END_DOCUMENT) {
       if (eventType == XmlResourceParser.START_TAG) {
          // Get the name of the tag (eg timetable, lecture or lab)
          String strName = timetable.getName();
          if (strName.equals("lecture")) {
             bFoundTimeTable = true;
             String days = timetable.getAttributeValue(null, "days");
             String room = timetable.getAttributeValue(null, "room");
             String time = timetable.getAttributeValue(null, "time");
             insertTimeTableRow(lectureTable, "     " + days);
             insertTimeTableRow(lectureTable, "     Time: " + time);
             insertTimeTableRow(lectureTable, "     Room: " + room);
             insertTimeTableRow(lectureTable, "     ");

          }
          if (strName.equals("lab")) {
             bFoundTimeTable = true;
             String days = timetable.getAttributeValue(null, "days");
             String room = timetable.getAttributeValue(null, "room");
             String time = timetable.getAttributeValue(null, "time");
             insertTimeTableRow(labTable, "     " + days);
             insertTimeTableRow(labTable, "     Time: " + time);
             insertTimeTableRow(labTable, "     Room: " + room);
             insertTimeTableRow(labTable, "     ");
          }
       }
       eventType = timetable.next();
    }
    // Handle no records available
    if (bFoundTimeTable == false) {
       final TableRow newRow = new TableRow(this);
       TextView noResults = new TextView(this);
       newRow.addView(noResults);
       lectureTable.addView(newRow);
    }
}

/**
 * {@code insertTimeTableRow()} helper method -- Inserts a new time table row {@code
 * TableRow} in the {@code TableLayout}
 *
 * @param timeTable
 *          The {@code TableLayout} to add the time table information to
 * @param strValue
 *          The value of the  text string to be inserted into the row
 */
private void insertTimeTableRow(final TableLayout timeTable, String strValue) {
    final TableRow newRow = new TableRow(this);
    TextView textView = new TextView(this);
```

```
        textView.setText(strValue);
        newRow.addView(textView);
        timeTable.addView(newRow);
    }
```

9. Add activities in AndroidManifest.xml for the two new layouts.

```
    <activity android:name=".contacts"
              android:label="@string/app_name">
    </activity>
    <activity android:name=".time_table"
              android:label="@string/app_name">
     </activity>
```

10. Run the application. Now there are different activities for the first two items of the menu. If you click on *Contact Information*, it will display following information:



Course Info COMP 4521

COMP 4521
Embedded Systems Software

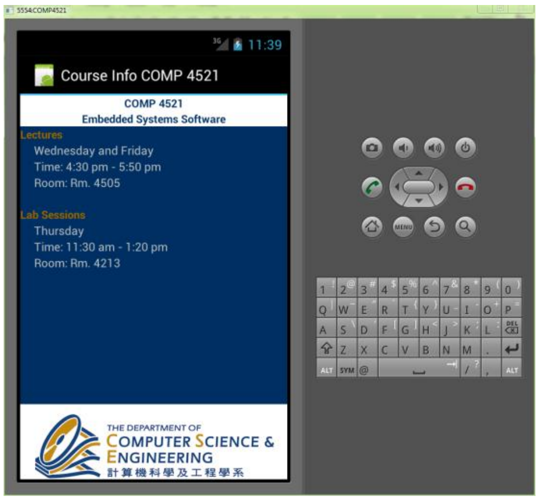
Instructor
Jogesh K. Muppala
Office: Rm. 3510
Tel: 2358 6978
Email: muppala@cse.ust.hk
Web: http://www.cse.ust.hk/~muppala/

Teaching Assistant(s)
Liu Yang
Office: Rm. 4205
Tel: 2358 6978
Email: liuyangcse@cse.ust.hk

Liu Yang
Office: Rm. 4205
Tel: 2358 6978
Email: liuyangcse@cse.ust.hk

Notice that phone numbers, email addresses and URLs are Linkified. If you click on them, they will open a dialer, email client (if configured) or a web browser respectively. Press the return button of the emulator to return to the previous screen and test other links.

11. If you click on *Time Table*, following information will be displayed.



Course Info COMP 4521

COMP 4521
Embedded Systems Software

Lectures
Wednesday and Friday
Time: 4:30 pm - 5:50 pm
Room: Rm. 4505

Lab Sessions
Thursday
Time: 11:30 am - 1:20 pm
Room: Rm. 4213