Lab-2: Activities, Fragments, SQLite, Intents and ActionBar

SAN JOSÉ STATE UNIVERSITY

Author: Vivek Maran (012556895)
Submitted to: Prof. Hungwen Li
Date: Wednesday, February 27, 2019

TABLE OF CONTENTS

1. Lab overview

The objective of this lab is to implement an application which incorporates android functions such as activities, fragments, intents, database, toast notifications and dialogs. A part of our final course project (CrowData) was implemented to accomplish the requirements listed in this lab. CrowData is a platform to crowd-source the datasets required for AI research and development. Researchers and organizations can post the requirements of dataset with their cloud URL, and individuals can contribute images and videos (onto the mentioned cloud URL) for the posted requirements. Following list details, the overall features and the one's implemented in this lab.

Feature	Implementation status	Requirement satisfied
Login and register screen	Login screen implemented with static username and pasword	TextView, Toast Notifications, Buttons Activities, Intents, passing data between activities, and SQLite database
Home screen with posts of dataset requirements	Implemented with local DB	
Settings screen for the application	Placeholders implemented	Fragments and Action-Bar
Ability to contribute images and videos to the posted requirements	Not applicable for Lab-2	Will be done far final project
Ability to post requirements	Not applicable for Lab-2	Will be done for final project
Statistics, cloud integration and miscallenous	Not applicable for Lab-2	Wil be done for final project

Table 1: Implementation and requirements satisfied

Following table lists the SDK's used and the testing status.

Target SDK version	API 28: Android 9.0 (Pie)
Minimum SDK version	15
Tested version	Android 8.0.0

2. Global application settings (App name, theme)

2.1 Setting the application name

This section describes the method used for configuring the application settings like the application name and theme. The `'AndroidManifest.xml'` file uses the application name from the `'strings.xml'` file present in the `'$PROJECT_ROOT/res/values/'` folder. Following changes were made in the `'strings.xml'` file to change the application name

```
<resources>
    <string name="app_name">CrowData</string>
</resources>
```

2. 2 Setting the theme
This `‘AndroidManifest.xml’` uses the theme from the `‘AppTheme’` attribute of the `‘styles.xml’` file. This attribute is configured to use the `‘Theme.AppCompat’` for setting a black theme to the application.

```
<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat">
    <!-- Customize your theme here. -->
</style>
```

2. 3 Configuring the activity launch
The login activity is set with the intent filter action `"android.intent.action.MAIN".` This tells the system that the login activity is the entry point activity that should be started when the application is launched. In order for the activity to be visible in launcher, the category `"android.intent.category.LAUNCHER"` is used. The other activities are set to category `"android.intent.category.DEFAULT"` to receive implicit intents.

```
<activity
    android:name="com.cmpe277.lab2.SettingsActivity"
    android:label="@string/title_activity_settings">
    <intent-filter>
        <action android:name="android.intent.action.SETTINGS" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
<activity android:name="com.cmpe277.lab2.LoginActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity android:name="com.cmpe277.lab2.MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.POSTS" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
```

2. 4 Dependencies
```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    implementation 'com.android.support:support-v4:28.0.0'
    implementation 'com.android.support:support-vector-drawable:28.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
    implementation 'com.github.deano2390:FlowTextView:2.0.5'
    implementation 'com.android.support:cardview-v7:26.1.0'
    implementation 'com.amulyakhare:com.amulyakhare.textdrawable:1.0.1'
}
```

3. Application architecture

As mentioned in the introductory section, the application consists of a login activity which is shown on the application launch. Once the user logs-in, the main activity is shown which contains the list of posts retrieved from the local database (via SQLite). The main activity also contains the action bar which has the ‘settings’ feature implemented using ‘PreferenceActivity’ and fragments. The following figures show the activity interaction diagram and the class diagram of the application.

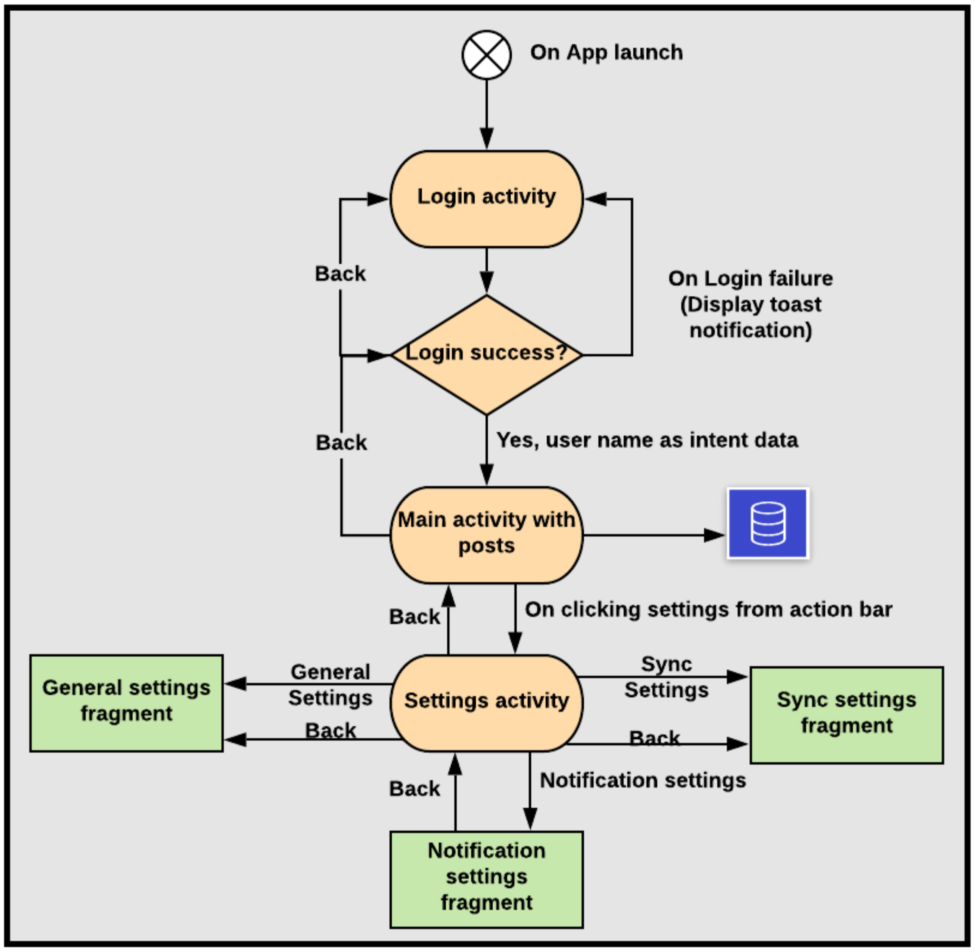


Figure 1: Activity diagram

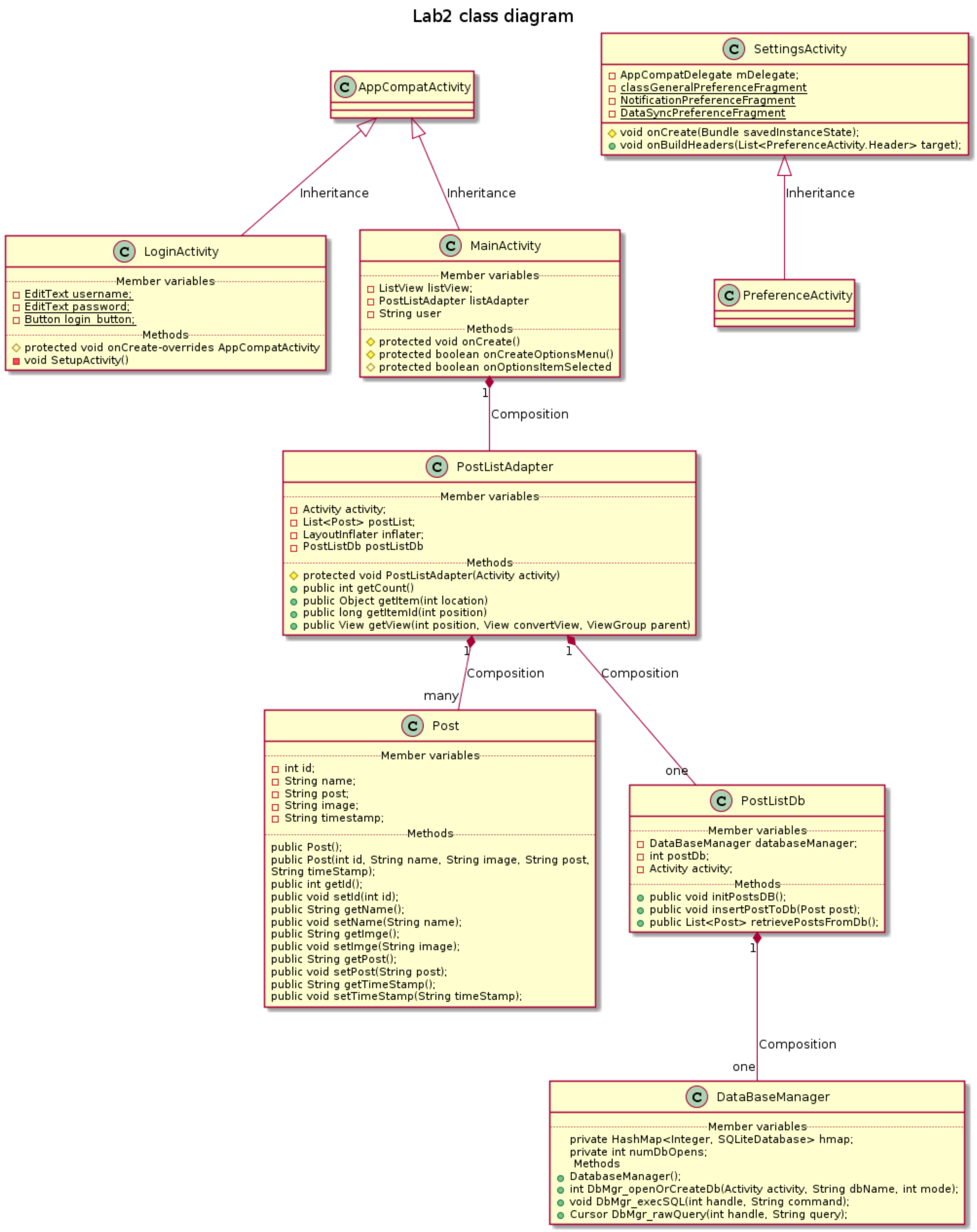

Lab2 class diagram
AppCompatActivity
SettingsActivity
AppCompatDelegate mDelegate;
classGeneralPreferenceFragment
NotificationPreferenceFragment
DataSyncPreferenceFragment
void onCreate(Bundle savedInstanceState);
void onBuildHeaders(List<PreferenceActivity.Header> target);
Inheritance
Inheritance
Inheritance
LoginActivity
Member variables
EditText username;
EditText password;
Button login_button;
Methods
protected void onCreate-overrides AppCompatActivity
void SetupActivity()
MainActivity
Member variables
ListView listView;
PostListAdapter listAdapter
String user
Methods
protected void onCreate()
protected boolean onCreateOptionsMenu()
protected boolean onOptionsItemSelected
PreferenceActivity
1
Composition
PostListAdapter
Member variables
Activity activity;
List<Post> postList;
LayoutInflater inflater;
PostListDb postListDb
Methods
protected void PostListAdapter(Activity activity)
public int getCount()
public Object getItem(int location)
public long getItemId(int position)
public View getView(int position, View convertView, ViewGroup parent)
1
Composition
many
1
Composition
one
Post
Member variables
int id;
String name;
String post;
String image;
String timestamp;
Methods
public Post();
public Post(int id, String name, String image, String post, String timeStamp);
public int getId();
public void setId(int id);
public String getName();
public void setName(String name);
public String getImge();
public void setImge(String image);
public String getPost();
public void setPost(String post);
public String getTimeStamp();
public void setTimeStamp(String timeStamp);
PostListDb
Member variables
DataBaseManager databaseManager;
int postDb;
Activity activity;
Methods
public void initPostsDB();
public void insertPostToDb(Post post);
public List<Post> retrievePostsFromDb();
1
Composition
one
DataBaseManager
Member variables
private HashMap<Integer, SQLiteDatabase> hmap;
private int numDbOpens;
Methods
DatabaseManager();
int DbMgr_openOrCreateDb(Activity activity, String dbName, int mode);
void DbMgr_execSQL(int handle, String command);
Cursor DbMgr_rawQuery(int handle, String query);

Figure 2: Class diagram

4. Activities

4. 1 Login activity

Login activity allows the user to login to the application. The login activity consists of an `‘ImageView’` to display the logo of the application, an `‘EditText’` for entering the user name, and an `‘EditText’`for the password. Additionally, `‘TextView’` for ‘Registration’ and icons for social media has been placed, but is yet to be implemented.

4. 1.1 Layout

Login activity uses an relative layout. The `‘EditText’`for the username is aligned center with respect to the parent using the `‘android:layout_centerInParent’` attribute. All the other views are placed relative to the `‘EditText’` view of the user name. The application logo is placed above this view and the password text box is placed below `‘EditText’` view of the password. Following is the layout

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="120dp"
    android:layout_height="150dp"
    android:layout_marginTop="40dp"
    app:srcCompat="@drawable/logo"
    android:layout_above="@id/editText"
    android:layout_centerHorizontal="true"  />

<EditText
    android:id="@+id/editText"
    android:layout_width="250dp"
    android:layout_height="40dp"
    android:background="#11000000"
    android:ems="10"
    android:hint="Username"
    android:inputType="textPersonName"
    android:textSize="16dp"
    android:layout_marginTop="20dp"
    android:layout_marginBottom="20dp"
    android:layout_centerInParent="true"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"/>

<EditText
    android:id="@+id/editText2"
    android:layout_width="250dp"
    android:layout_height="40dp"
    android:background="#11000000"
    android:ems="10"
    android:hint="Password"
    android:layout_below="@id/editText"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:layout_marginBottom="20dp"
    android:inputType="textPassword"/>
```

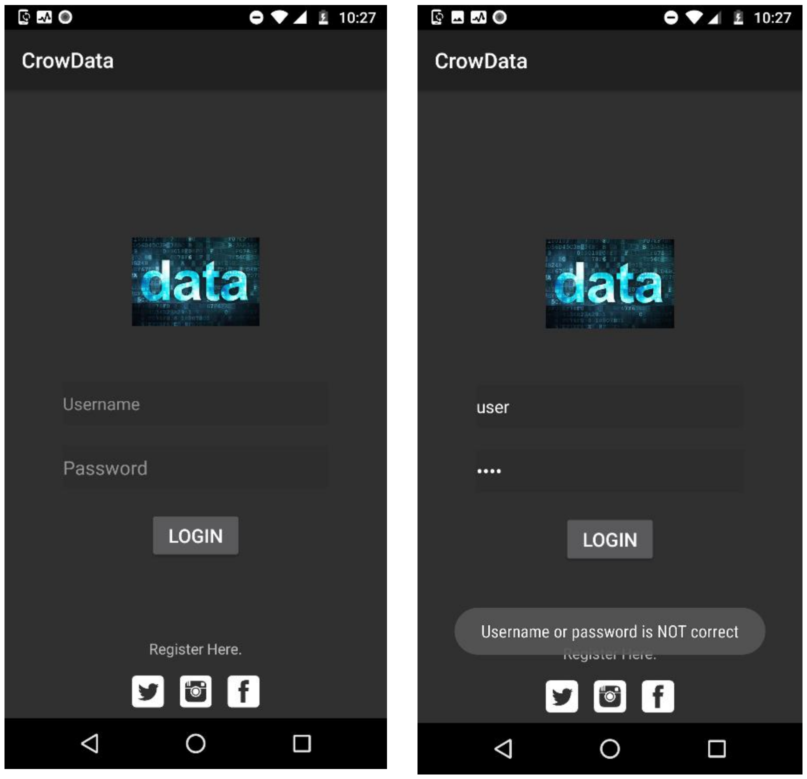
```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/editText2"
    android:text="Login"
    android:textColor="#fff"
    android:textSize="18sp"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true" />
```

4. 1.2 Controller

The `‘LoginActivity’`class implements the controller logic for the login activity. It registers an ‘`OnClickListener’` for the login button, where it implements the functionality of login. As of now, it validates the provided username and password against the hardcoded strings of “user” and “pass”. If the validation succeeds, it starts the ‘MainActivity’ which contains the list of posts containing the requirements of dataset. It also passes the user information to the main activity for display in successive screens. If the validation fails, it displays a toast notification as ‘Username or password is NOT correct’.

```
private void SetupActivity() {
    username = (EditText) findViewById(R.id.editText);
    password = (EditText) findViewById(R.id.editText2);
    login_button = (Button) findViewById(R.id.button);
    Log.i(TAG, "SetupActivity++");
    login_button.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Log.i(TAG, "Login onClick++");
                    if (username.getText().toString().equals("user") &&
                            password.getText().toString().equals("pass")) {
                        Toast.makeText(LoginActivity.this,
                                "Username and " +
                                        "password" +
                                        " is correct",
                                Toast.LENGTH_SHORT).show();
                        Intent intent = new Intent("android.intent.action" +
                                ".POSTS");
                        intent.putExtra("User",
                                username.getText().toString());
                        Log.d(TAG, "Starting POSTS intent");
                        startActivity(intent);
                    } else {
                        Log.d(TAG, "Password invalid " + "Expected: " +
                                "pass" + "Typed: "+password.getText().toString());
                        Toast.makeText(LoginActivity.this, "" +
                                        "Username or " +
                                        "password" +
                                        " is NOT correct",
                                Toast.LENGTH_SHORT).show();
                    }
                    Log.i(TAG, "Login onClick--");
                    }
                }
);
```

4. 1.3 Screenshots



Username or password is NOT correct

Figure 3: Login screen and login failure notification

4. 2 Main activity

The main activity retrieves the posts (made by the organization (or) research groups) listing dataset requirements and displays them in a list view. Each list item is a post containing the text describing the requirements and the profile picture of the one who posted. The activity uses adapter design to generate the view. The data source for the adapter is an `‘ArrayList’` populated from a SQLiteDB.

In addition to the view, the main activity also contains an actiobar with an overflow menu containing a settings button and a logo containing the initials of the logged in user. The user details is received as an intent message from the `‘LoginActivity’class`

4. 2.1 Layout

The main activity’s layout is a ‘`LinearLayout’` with a ‘`ListView’.` This view is inflated and populated using the adapter. Each item in the `‘ListView’` is a `‘FlowTextView’` which can wrap text around images. This is not natively present in the Android SDK and a library `‘uk.co.deanwild.flowtextview.FlowTextView’` is used for this functionality.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical">

    <ListView
        android:id="@+id/list"
        android:layout_width="fill_parent"
        android:layout_height="match_parent"
        android:divider="@null" />
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical">

    <uk.co.deanwild.flowtextview.FlowTextView
        android:id="@+id/ftv"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAlignment="center">

        <ImageView
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:padding="10dip"
            android:src="@drawable/stanfordlogo"/>
    </uk.co.deanwild.flowtextview.FlowTextView>
    <View android:layout_width="fill_parent" android:layout_height="1dp"
android:background="#ffffff"/>
</LinearLayout>
```

4. 2.2 Controller

Populating posts

The ‘`MainActivity`’ class implements the controller logic for the main activity. As a first step, the class initializes the helper class ‘`PostListAdapter`’ which extends from ‘`BaseAdapter`’. This class pulls the posts from the SQLite database and populates into a ‘`ArrayList`’ which is the added to the ‘`ListView`’

```
/** Setup the list view with the associated adapter */
listAdapter = new PostListAdapter(this);
listView = findViewById(R.id.list);
listView.setAdapter(listAdapter);

public PostListAdapter(Activity activity) {
    Log.i(TAG, "PostListAdapter()++");
    this.activity = activity;
    this. postListDb = new PostListDb(activity);
    postListDb.initPostsDB();
    this.postList =postListDb.retrievePostsFromDb();
    Log.i(TAG, "PostListAdapter()--");
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    Log.i(TAG, "getView()++ Position: "+position);
    if (inflater == null)
        inflater = (LayoutInflater) activity
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    if (convertView == null)
        convertView = inflater.inflate(R.layout.project_post,null);
    FlowTextView flowTextView = (FlowTextView) convertView.findViewById(R.id.ftv);
    flowTextView.setTextColor(Color.parseColor("#FFFFFF"));
    Log.d(TAG, "Post: "+ postList.get(position).getPost());
    flowTextView.setText(postList.get(position).getPost());
    Log.i(TAG, "getView()--");
    return convertView;
}
```

The `‘getView’` method which is an override of the one in the `‘BaseAdapter’` class is called to populate each item in the `‘ListView’.` The view containing the list item is inflated, populated with the post and then returned to the caller.

<u>Action bar</u>
The main activity uses a custom view in the action bar which enables the display of the profile logo containing the initials of the user. The profile logo is created using the ‘`TextDrawable’` view of the `'com.amulyakhare:com.amulyakhare.textdrawable:1.0.1'` library. This drawable is set into the `‘ImageView’`, and then placed into the action bar

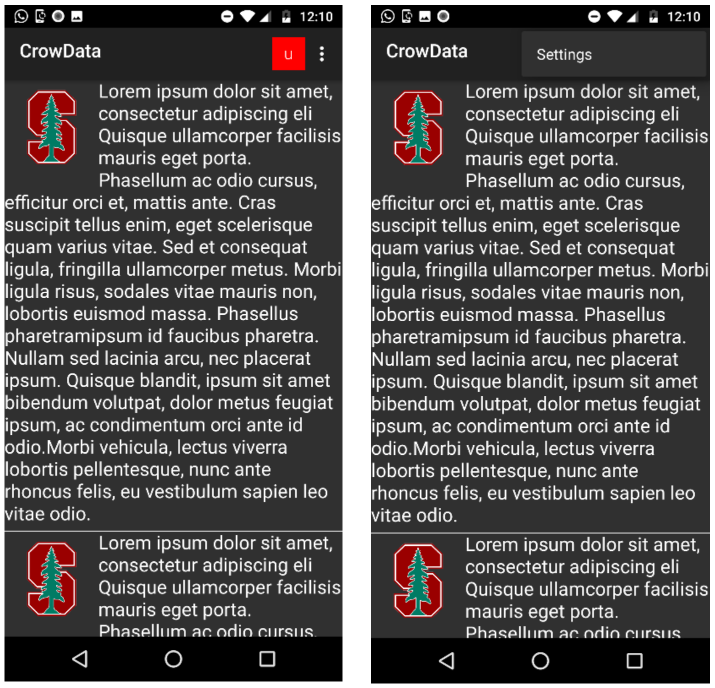
```
/**
 * Setup the action bar for this activity
 */
private void SetupActionBar()
{
    Log.i(TAG, "SetupActionBar++");
    ActionBar actionBar = getSupportActionBar();
    LayoutInflater inflater =
            (LayoutInflater) this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    View v = inflater.inflate(R.layout.custom_toolbar, null);
    ImageView login_icon = (ImageView) v.findViewById(R.id.image_view);
    TextDrawable drawable = TextDrawable.builder()
            .buildRect(""+user.charAt(0), Color.RED);
    login_icon.setImageDrawable(drawable);
    actionBar.setDisplayShowTitleEnabled(false);
    actionBar.setCustomView(v);
    actionBar.setDisplayShowCustomEnabled(true);
    Log.i(TAG, "SetupActionBar--");
}
```

<u>Settings menu</u>

Settings button is added as a overflow menu in the action bar using the `'onCreateOptionsMenu'` override. An `'OnClickListener'` is registered for this button. Upon click of this button a new intent is started.

```
// handle button activities
@Override
 public boolean onOptionsItemSelected(MenuItem item) {
     Log.i(TAG, "onCreateOptionsMenu++");
     int id = item.getItemId();
     if (id == R.id.stngs) {
         Intent intent = new Intent("android.intent.action" +
                 ".SETTINGS");
         startActivity(intent);
     }
     Log.i(TAG, "onCreateOptionsMenu--");
     return super.onOptionsItemSelected(item);
 }
```

4.2.3 Screenshots



Lorem ipsum dolor sit amet, consectetur adipiscing eli Quisque ullamcorper facilisis mauris eget porta. Phasellum ac odio cursus, efficitur orci et, mattis ante. Cras suscipit tellus enim, eget scelerisque quam varius vitae. Sed et consequat ligula, fringilla ullamcorper metus. Morbi ligula risus, sodales vitae mauris non, lobortis euismod massa. Phasellus pharetramipsum id faucibus pharetra. Nullam sed lacinia arcu, nec placerat ipsum. Quisque blandit, ipsum sit amet bibendum volutpat, dolor metus feugiat ipsum, ac condimentum orci ante id odio.Morbi vehicula, lectus viverra lobortis pellentesque, nunc ante rhoncus felis, eu vestibulum sapien leo vitae odio.

Lorem ipsum dolor sit amet, consectetur adipiscing eli Quisque ullamcorper facilisis mauris eget porta. Phasellum ac odio cursus

Figure 4: Main activity with action bar and settings

4. 3 Settings activity

The ‘SettingsActivity’ inherits from ‘PreferenceActivity’ to implement the settings functionality. The settings (or) preferences is categorized into multiple preference headers. The list of preference headers is displayed in the settings activity. Each preference header is associated with a fragment in which the preferences present within that header are displayed. The list of preference headers are placed in ‘res/values/xml/pref_headers.xml’ as listed below.

```
<preference-headers xmlns:android="http://schemas.android.com/apk/res/android">

    <!-- These settings headers are only used on tablets. -->
    <header

android:fragment="com.cmpe277.lab2.SettingsActivity$GeneralPreferenceFragment"
        android:icon="@drawable/ic_info_white_24dp"
        android:title="@string/pref_header_general" />

    <header

android:fragment="com.cmpe277.lab2.SettingsActivity$NotificationPreferenceFragment"
        android:icon="@drawable/ic_notifications_white_24dp"
        android:title="@string/pref_header_notifications" />

    <header

android:fragment="com.cmpe277.lab2.SettingsActivity$DataSyncPreferenceFragment"
        android:icon="@drawable/ic_sync_white_24dp"
        android:title="@string/pref_header_data_sync" />

</preference-headers>
```

It consists of three headers General, Notifications, Data & Sync. Each of these headers are associated with the static fragments ‘GeneralPreferenceFragment’, ‘NotificaionPreferenceFragment’ and ‘DataSyncPreferenceFragment’ respectively using the ‘android:fragment’ attribute. The preference headers are displayed on the ‘onBuildHeaders’ override using the ‘loadHeadersFromResource’ API.

```
@Override
@TargetApi(Build.VERSION_CODES.HONEYCOMB)
public void onBuildHeaders(List<PreferenceActivity.Header> target) {
    loadHeadersFromResource(R.xml.pref_headers, target);
}
```

By default, the action bar was not displayed in the ‘SettingsActivity’ as it inherits from ‘PreferenceActivity’ instead of ‘AppCompatActivity’. In order to make action bar work, ‘AppCompatDelegate’ was used.

Each of the fragments corresponding the preference header is a static inner class in the ‘SettingsActivityClass’. These fragments inherit from ‘PreferenceFragment’. They use ‘addPreferencesFromResource’ API on the ‘OnCreate’ method of the fragment to add preferences corresponding to that header.

```
public static class GeneralPreferenceFragment extends PreferenceFragment {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setHasOptionsMenu(true);
        addPreferencesFromResource(R.xml.pref_general);
        // Bind the summaries of EditText/List/Dialog/Ringtone preferences
        // to their values. When their values change, their summaries are
        // updated to reflect the new value, per the Android Design
        // guidelines.
        bindPreferenceSummaryToValue(findPreference("full_name"));
        bindPreferenceSummaryToValue(findPreference("email_address"));
    }
```

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">

    <!-- NOTE: EditTextPreference accepts EditText attributes. -->
    <!-- NOTE: EditTextPreference's summary should be set to its value by the
activity code. -->
    <EditTextPreference
        android:key="full_name"
        android:title="Name"
        android:summary="Enter Your Complete Name"
        android:dialogTitle="Your Name"
        android:dialogMessage="Enter Your Complete Name"
        android:defaultValue=""
        android:inputType="textCapWords"/>

    <!-- NOTE: Hide buttons to simplify the UI. Users can touch outside the dialog to
         dismiss it. -->
    <!-- NOTE: ListPreference's summary should be set to its value by the activity
code. -->
    <EditTextPreference
        android:key="email_address"
        android:title="Email Address"
        android:summary="Enter Your Email Address"
        android:dialogTitle="Enter Your Email Address"
        android:dialogMessage="Enter Your Email Address"
        android:defaultValue=""
        android:inputType="textEmailAddress"/>

</PreferenceScreen>
```

’The main activity retrieves the posts (made by the organization (or) research groups) listing dataset requirements and displays them in a list view. Each list item is a post containing the text describing the requirements and the profile picture of the one who posted. The activity uses adapter design to generate the view. The data source for the adapter is an `‘ArrayList’` populated from a SQLiteDB.

In addition to the view, the main activity also contains an actiobar with an overflow menu containing a settings button and a logo containing the initials of the logged in user. The user details is received as an intent message from the `‘LoginActivity’`class

4.3.1 Screenshots

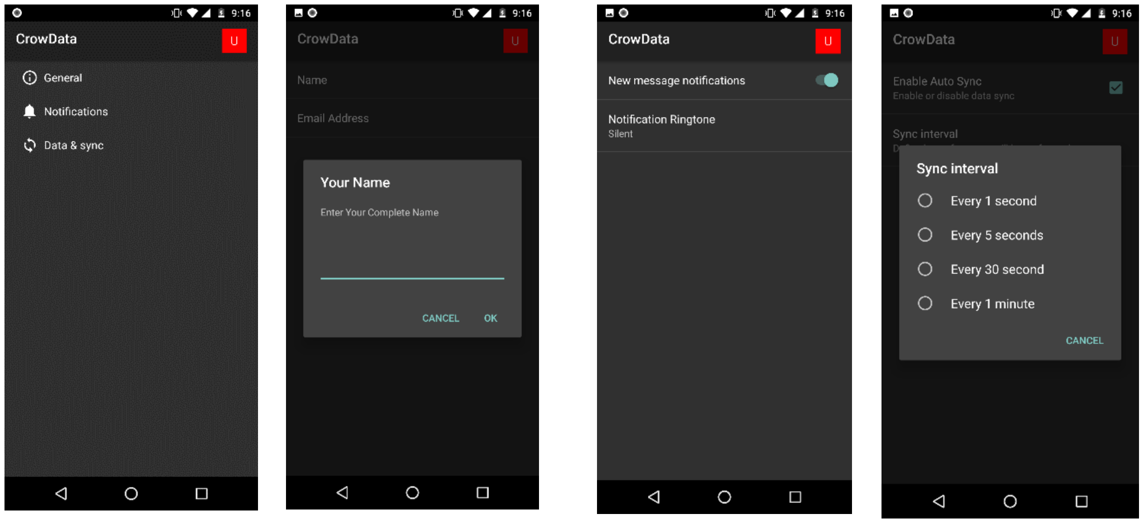


Figure 5: Settings screenshot

5. Meeting lab requirements

5. 1 Multiple activities

The app uses three activities login activity, main activity, and settings activity. Please refer figure-3, figure-4 and figure-5.

5. 2 Simple UI elements

- **TextView:** Used in the main activity for displaying the post along with the image. Please refer figure-4
- **Toast notifications:** Used to indicate login failures, please refer figure-3
- **Buttons:** Used for login and settings. Please refer figure-3 and figure-4
- **Dialogs:** Used in the Settings to get the preferences. Please refer figure-5

5. 3 Fragments

Each of the preference headers is implemented as static fragments. Please refer figure-5

5. 4 Action bar with custom view

An action bar with custom view is used across all the activities.

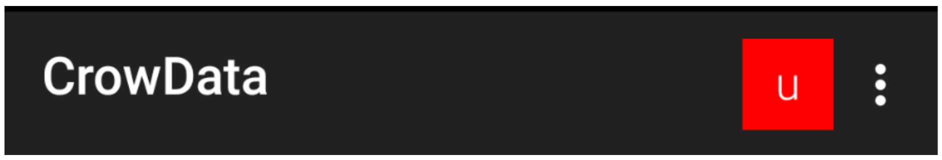


Figure 6: Action bar with custom view

```
/**
 * Setup the action bar for this activity
 */
private void SetupActionBar()
{
    Log.i(TAG, "SetupActionBar++");
    ActionBar actionBar = getSupportActionBar();
    LayoutInflater inflater =
            (LayoutInflater) this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    View v = inflater.inflate(R.layout.custom_toolbar, null);
    ImageView login_icon = (ImageView) v.findViewById(R.id.image_view);
    TextDrawable drawable = TextDrawable.builder()
            .buildRect(""+user.charAt(0), Color.RED);
    login_icon.setImageDrawable(drawable);
    actionBar.setDisplayShowTitleEnabled(false);
    actionBar.setCustomView(v);
    actionBar.setDisplayShowCustomEnabled(true);
    Log.i(TAG, "SetupActionBar--");
}
```

5. 4 Passing data between activities.

The user profile logo displayed in the action bar of the main activity (figure-6) is passed as a intent message from the login activity.

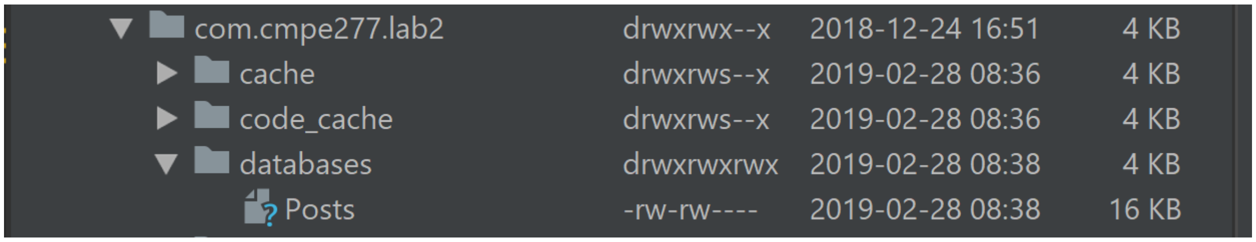
```
Intent intent = new Intent("android.intent.action" +
        ".POSTS");
intent.putExtra("User",
        username.getText().toString());
Log.d(TAG, "Starting POSTS intent");
```

5. 5 Using explicit/implicit intents

The main activity from login activity is started using intents. Similarly, the settings activity from the main activity is started using intents. Please refer the above code-snippet for intents.

5. 6 Using SQLite database

The main activity retrieves the list of posts from a local SQLite application database ('Posts'). This database was populated once using code, and is used successively for retrievals.



```
▼ com.cmpe277.lab2      drwxrwx--x    2018-12-24 16:51    4 KB
    ▶ cache             drwxrws--x    2019-02-28 08:36    4 KB
    ▶ code_cache        drwxrws--x    2019-02-28 08:36    4 KB
    ▼ databases         drwxrwxrwx    2019-02-28 08:38    4 KB
        Posts           -rw-rw----    2019-02-28 08:38   16 KB
```

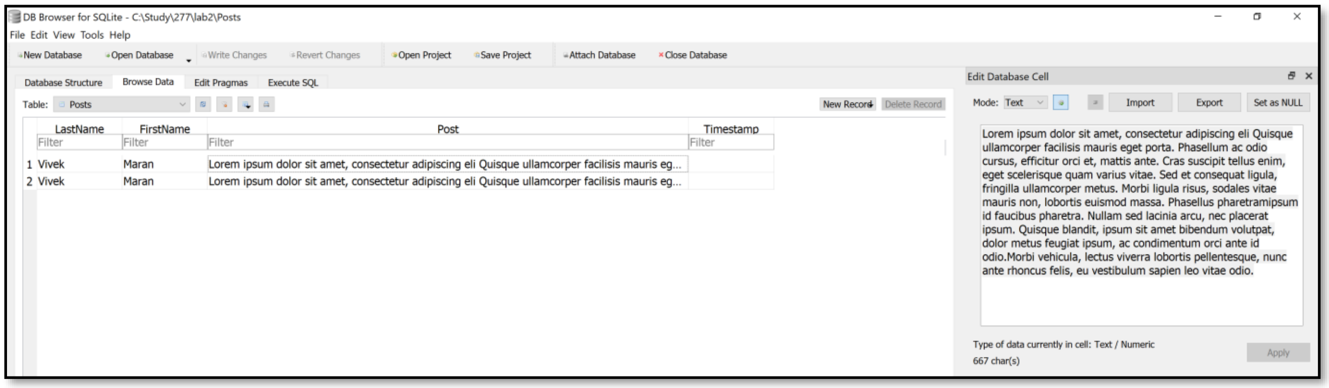


Table: Posts

	LastName	FirstName	Post	Timestamp
	Filter	Filter	Filter	Filter
1	Vivek	Maran	Lorem ipsum dolor sit amet, consectetur adipiscing eli Quisque ullamcorper facilisis mauris eg...	
2	Vivek	Maran	Lorem ipsum dolor sit amet, consectetur adipiscing eli Quisque ullamcorper facilisis mauris eg...	

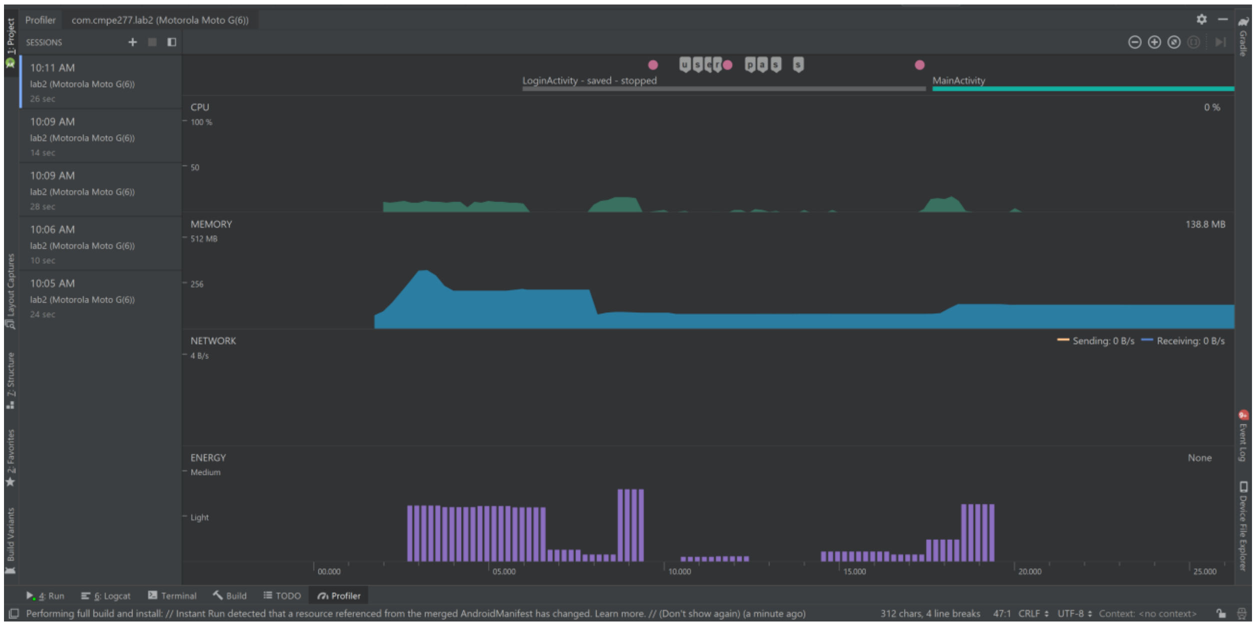
Mode: Text

Lorem ipsum dolor sit amet, consectetur adipiscing eli Quisque ullamcorper facilisis mauris eget porta. Phasellum ac odio cursus, efficitur orci et, mattis ante. Cras suscipit tellus enim, eget scelerisque quam varius vitae. Sed et consequat ligula, fringilla ullamcorper metus. Morbi ligula risus, sodales vitae mauris non, lobortis euismod massa. Phasellus pharetramipsum id faucibus pharetra. Nullam sed lacinia arcu, nec placerat ipsum. Quisque blandit, ipsum sit amet bibendum volutpat, dolor metus feugiat ipsum, ac condimentum orci ante id odio.Morbi vehicula, lectus viverra lobortis pellentesque, nunc ante rhoncus felis, eu vestibulum sapien leo vitae odio.

Type of data currently in cell: Text / Numeric
667 char(s)

Figure 7: SQLite DB of the application

6. Screenshots from android monitor



Performing full build and install: // Instant Run detected that a resource referenced from the merged AndroidManifest has changed. Learn more. // (Don't show again) (a minute ago)   312 chars, 4 line breaks   47:1   CRLF   UTF-8   Context: <no context>

Figure 8: Profiler

7. References

[1] https://developer.android.com/guide/topics/manifest/manifest-intro

[2] https://medium.com/androiddevelopers/picking-your-compilesdkversion-minsdkversion-targetsdkversion-a098a0341ebd

[3] https://developer.android.com

[4] https://examples.javacodegeeks.com/android/core/ui/settings/android-settings-example/

[5]
https://android.googlesource.com/platform/development/+/58bf5b99e6132332afb8b44b4c8cedf5756ad464/samples/Support7Demos/src/com/example/android/supportv7/app/AppCompatPreferenceActivity.java

[6] https://www.androidhive.info/2014/06/android-facebook-like-custom-listview-feed-using-volley/

[7] https://github.com/amulyakhare/TextDrawable

[8] https://github.com/deano2390/FlowTextView

[9] https://www.youtube.com/watch?v=GAdGmJxfcf8

[10] https://stuff.mit.edu/afs/sipb/project/android/docs/guide/components/intents-filters.html