

Android UI Design Patterns

Dominic Duggan
Stevens Institute of Technology

Design Philosophy

- Clear vs. “simple”
- Content vs. chrome
- Consistent yet engaging (elegant variation)
- Enhanced by cloud

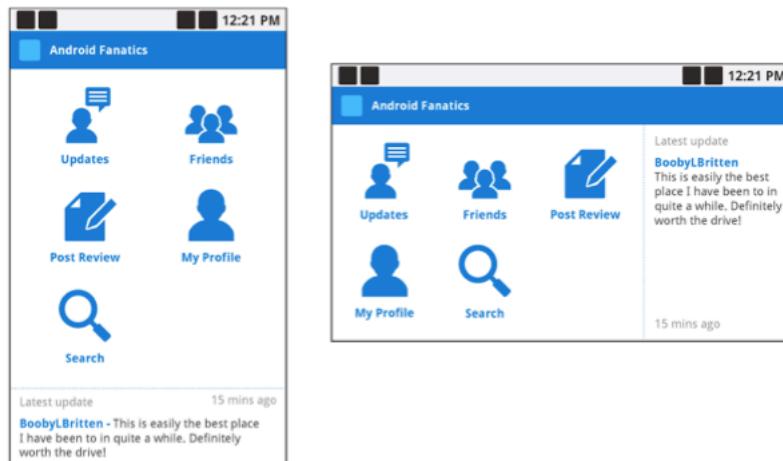


Design Patterns

- Dashboard
- Action Bar
- Search Bar
- Quick Actions
- Contextual Action Bar
- Companion Widget
- Multi-pane Layout
- App Navigation

DASHBOARD

Dashboard

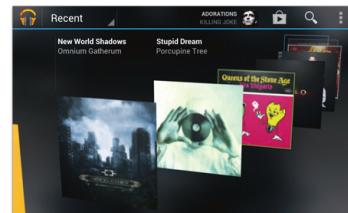


Dashboard

- “*What can I do with this app? What’s new?*”
- A quick intro to an app
- Full-screen
- Can be organized by:
 - Features
 - Categories
 - Accounts

Dashboard Recommendations

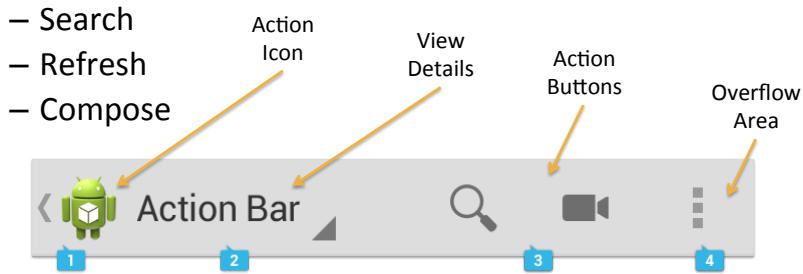
- DO highlight what's new
- DO focus on 3-6 most important choices
- DO be flavorful



ACTION BAR

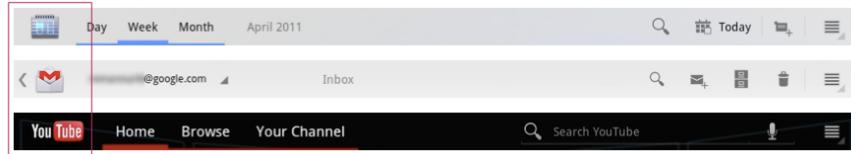
Action Bar

- “How can I do <common action> quickly?”
- Dedicated real estate
 - Navigation
 - Frequently used operations
- Actions common across app



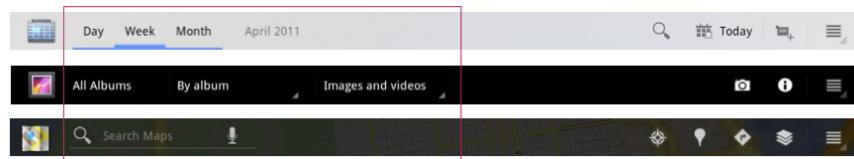
Action Bar

- Action Icon
 - Branding
 - Upward navigation



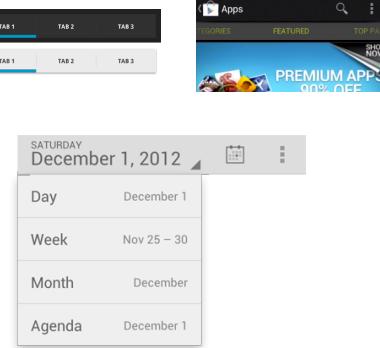
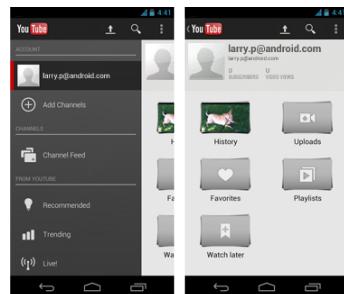
Action Bar

- View Details
 - Tabs (fixed or scrolling)
 - Drop-down Menus
 - Drawers



Action Bar

- View Details
 - Tabs (fixed or scrolling)
 - Drop-down Menus
 - Drawers (top bar only)

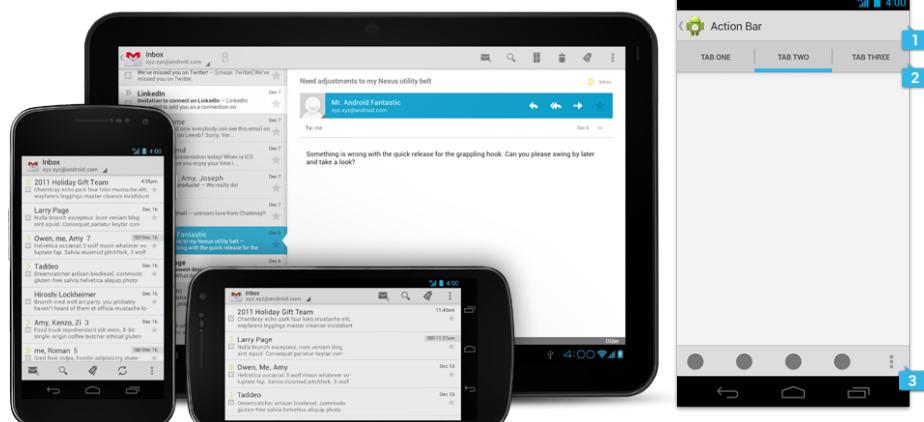


Action Bar

- Action Buttons
 - FIT: Frequent, Important or Typical
 - Icon-only, text-only, icon-and-text
 - Overflow menu

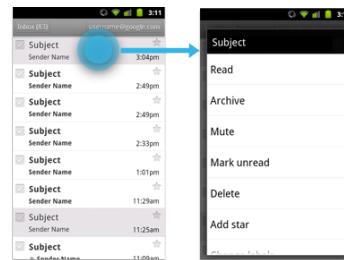


Action Bar on Smaller Screens

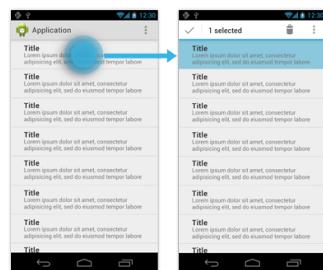


Contextual Actions

- Out: Contextual Menus

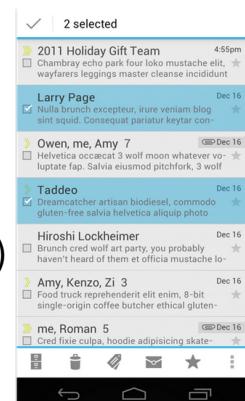


- In: Contextual Action Bar



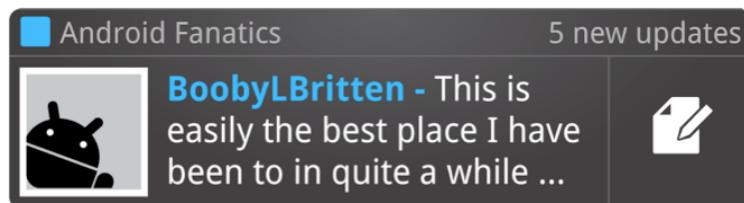
Contextual Action Bar

- When items are selected
 - Touch and hold
- Three sections
 - **Done** button (for releasing selection)
 - Selection details
 - Action Buttons
- Implemented using **ActionMode**



WIDGETS

Companion Widget



Companion Widget

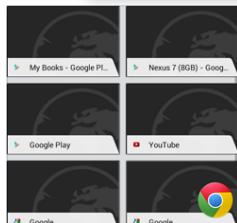
- “*Can I make this app a fun part of my Home screen?*”
- Supports app by displaying on Home screen
- Makes Home feel more custom, personalized

Types of Widgets

- Information



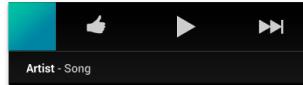
- Collection



- Control



- Hybrid



Companion Widget Recommendations

- DO provide value above a simple app icon (content)
- DO handoff to the full app for real tasks
- DO be space efficient
- DON'T just provide a larger app launcher

MULTI-PANE LAYOUTS

Multi-Pane Layouts

- Increased real estate
 - Give more context
 - Consolidate multiple related phone screens
- Selection on the left
- Content on the right

Example

leDE		Save			Cancel		New	Edit	⋮
Patients		Patient ID:	1234						
1234		Patient study key:	*****		Consent				
		Last change:	12/6/2013						
		Changed by:	kathy						
		<input checked="" type="checkbox"/> Still active?							
		<input type="checkbox"/> Patient is already registered at another site?							
		Patient study key (from other site):							

Example

leDEA

Consent Topics		Confidentiality
<input checked="" type="checkbox"/> Getting Started	You are eligible for this study because you are receiving care in one of the included sites. Records of this study will be kept confidential and you will not be identified in any written or verbal reports. Only clinic staff will have access to records. Your records will be identified by a code and no names, birthdates or other identifying information will be used to create that code. The research records will be kept in a secure manner and computer records will be password protected. There is a possibility that the Cameroon National Ethics Committee, sponsors and collaborating groups may inspect your records. This study is sponsored by the National Institute of Allergy and Infectious Diseases (NIAID), the National Cancer Institute (NCI) and the National Institute of Childhood Health and Human Development (NICHD), in the United States.	
<input checked="" type="checkbox"/> Introduction		
<input type="checkbox"/> Confidentiality		
<input type="checkbox"/> Purpose of Study		
<input type="checkbox"/> Risk of Participating		
<input type="checkbox"/> Benefits of the Study		
<input type="checkbox"/> Costs/ Compensation		
<input type="checkbox"/> Alternative to Participating		
<input type="checkbox"/> Withdrawal		
<input type="checkbox"/> Contact Information		
<input type="checkbox"/> Informed Consent		

[I understand.](#) No [Next](#) [Cancel](#)

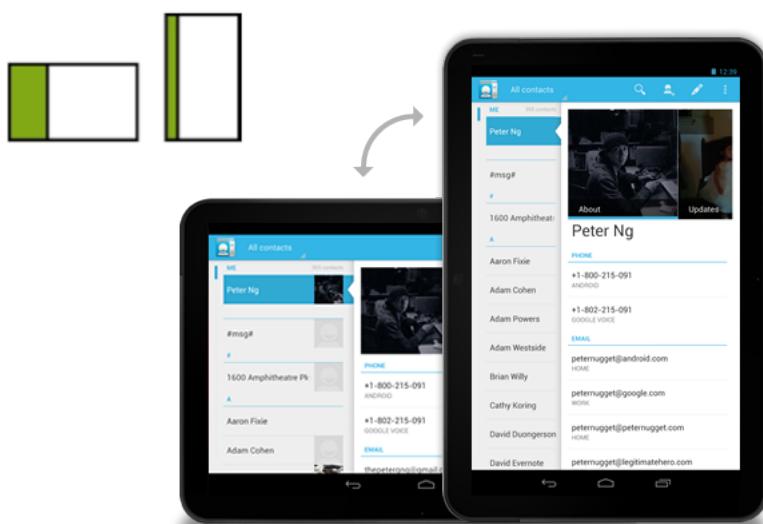
Stretch



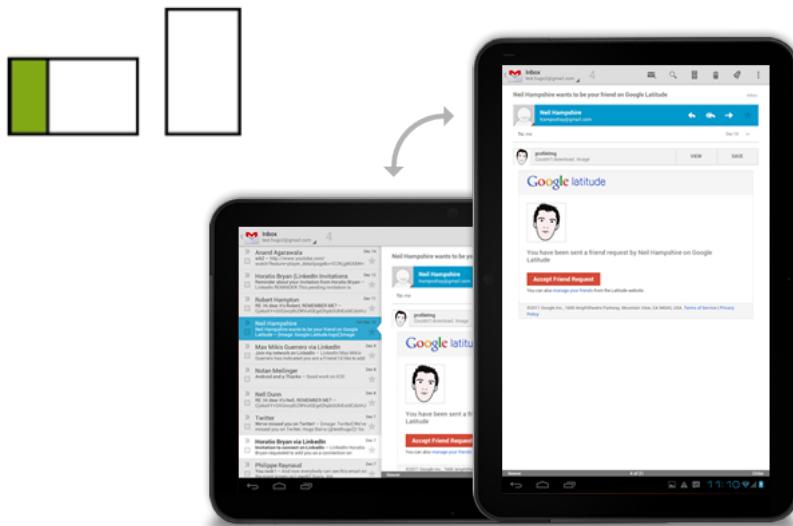
Stack



Expand/Collapse



Show/Hide



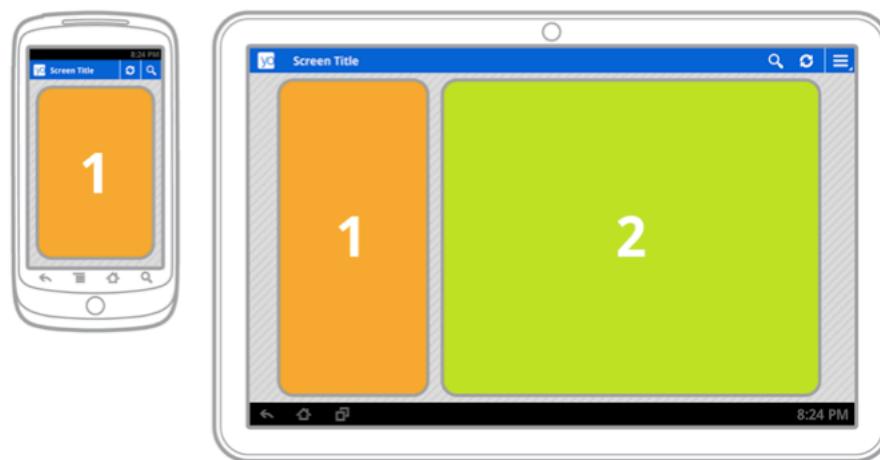
Strategies

- Orientation changes should preserve functional parity
- Strategies apply per-screen, not per app
- For **show/hide**, use **UP** navigation to show the master pane
 - e.g. Gmail

Fragments

- “Fragments” of an Activity
- Unit of reuse between Activities
- Separation of concerns
- Fragments don’t necessarily have views
 - Lifecycle construct

Fragments

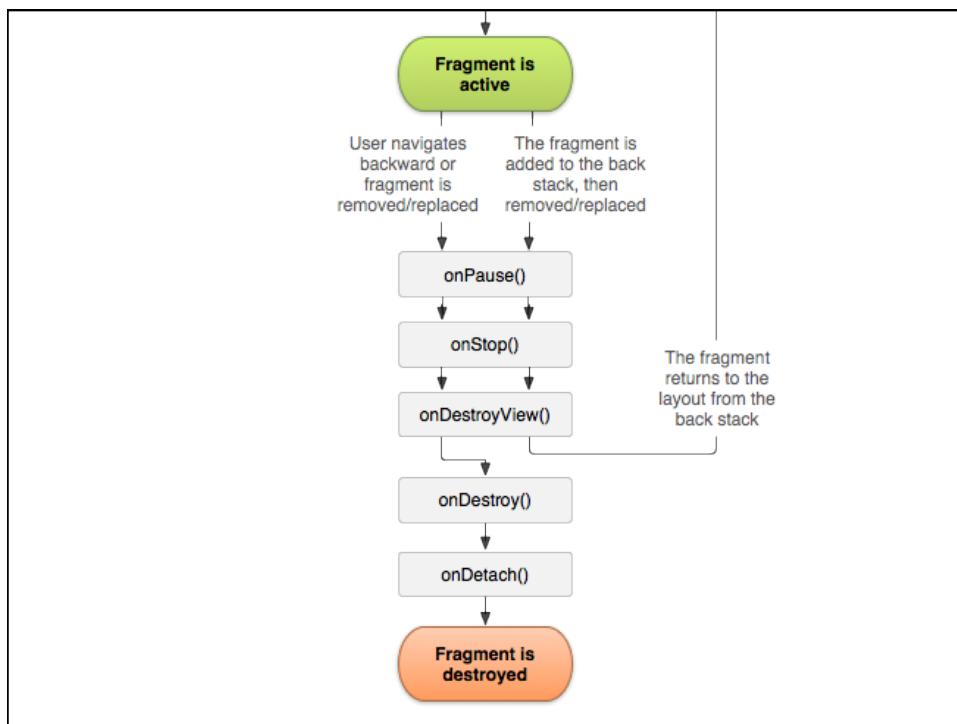
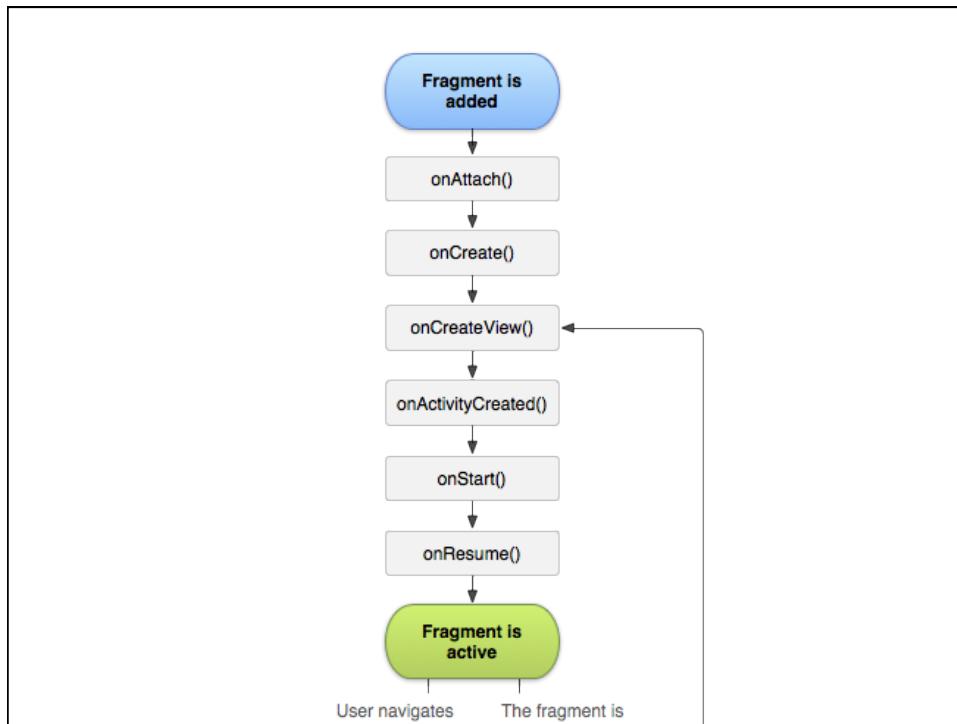


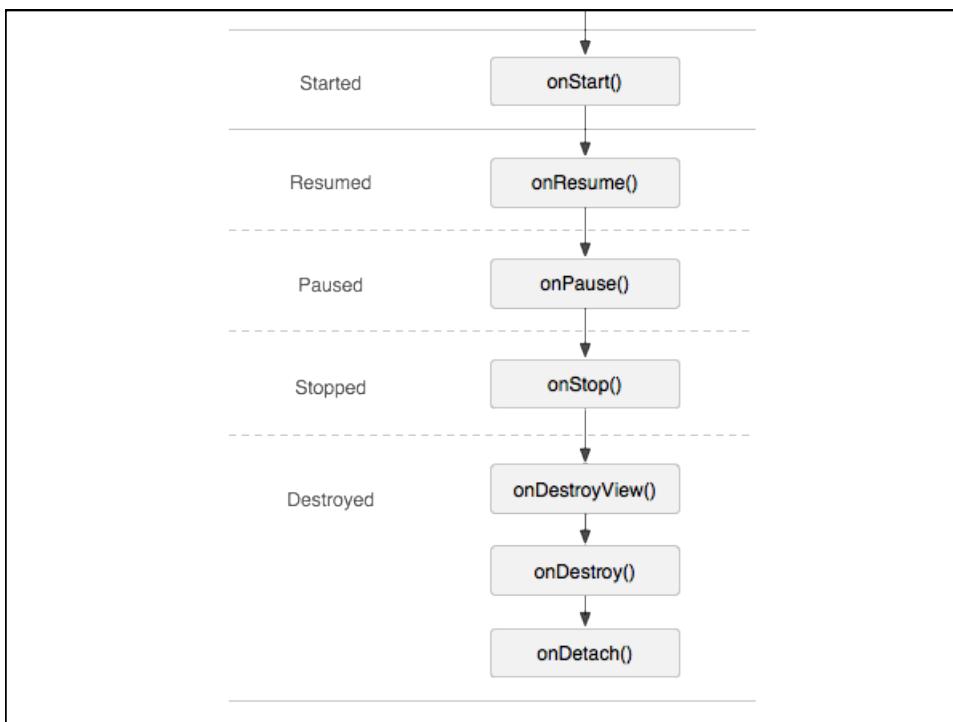
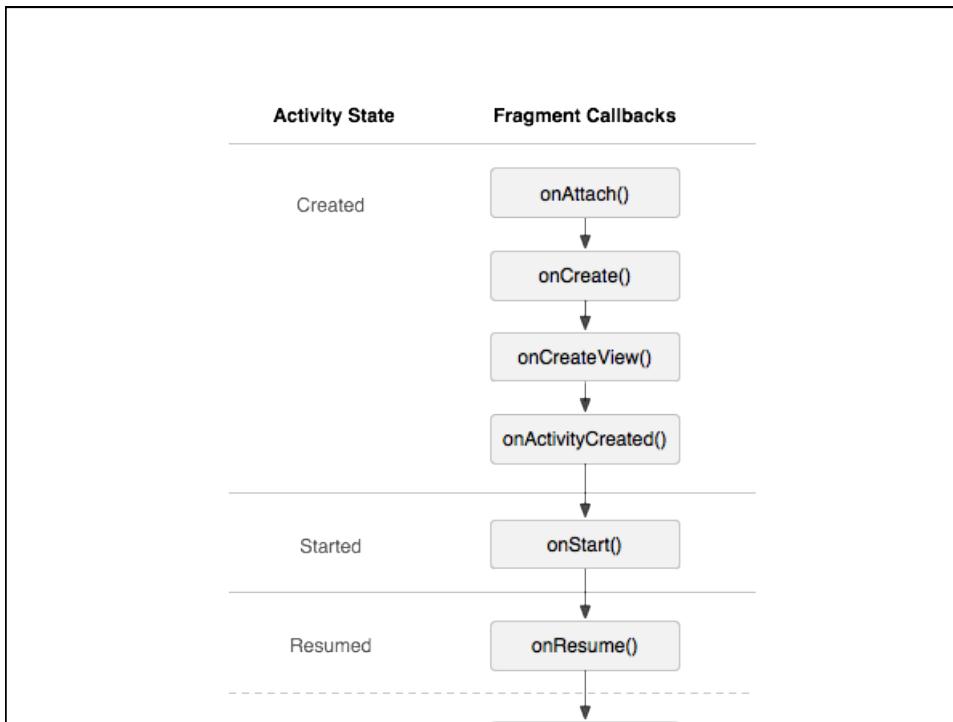
Static (Two-Pane) Layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="horizontal"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent">  
  
    <fragment android:name="com.example.android.fragments.HeadlinesFragment"  
        android:id="@+id/headlines_fragment"  
        android:layout_weight="1"  
        android:layout_width="0dp"  
        android:layout_height="match_parent" />  
  
    <fragment android:name="com.example.android.fragments.ArticleFragment"  
        android:id="@+id/article_fragment"  
        android:layout_weight="2"  
        android:layout_width="0dp"  
        android:layout_height="match_parent" />  
  
</LinearLayout>
```

Static (Two-Pane) Layout

```
public class ArticleFragment extends Fragment {  
    @Override  
    public View onCreateView(LayoutInflater inflater,  
                             ViewGroup container,  
                             Bundle savedInstanceState) {  
        // Inflate the layout for this fragment  
        return inflater.inflate(R.layout.article_view,  
                               container, false);  
    }  
}  
  
public class MainActivity extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.news_articles);  
    }  
}
```

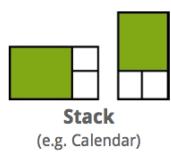




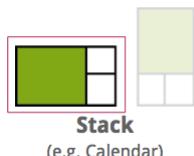
Resources

- Images, icons
 - drawable-ldpi/
 - drawable-mdpi/
 - drawable-hdpi/
- Layouts:
 - layout-normal/
 - layout-large/
 - layout-xlarge/
 - layout-xlarge-port/

Multi-pane Layouts via Resources



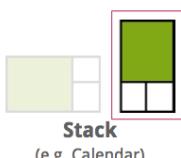
Multi-pane Layouts via Resources



layout-xlarge-land/my_layout.xml

```
<LinearLayout android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.MainPaneFragment"
        android:id="@+id/main_pane"
        android:layout_width="0dip" android:layout_weight="1"
        android:layout_height="match_parent" />
    <LinearLayout android:orientation="vertical"
        android:layout_width="wrap_content"
        android:layout_height="match_parent">
        <fragment android:name="com.example.MonthFragment"
            android:id="@+id/month_pane"
            android:layout_width="wrap_content"
            android:layout_height="0dip" android:layout_weight="1" />
        <fragment android:name="com.example.CalendarListFragment"
            android:id="@+id/list_pane"
            android:layout_width="wrap_content"
            android:layout_height="0dip" android:layout_weight="1" />
    </LinearLayout>
</LinearLayout>
```

Multi-pane Layouts via Resources



layout-xlarge-port/my_layout.xml

```
<LinearLayout android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.MainPaneFragment"
        android:id="@+id/main_pane"
        android:layout_width="match_parent"
        android:layout_height="0dip" android:layout_weight="1" />
    <LinearLayout android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <fragment android:name="com.example.MonthFragment"
            android:id="@+id/month_pane"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <fragment android:name="com.example.CalendarListFragment"
            android:id="@+id/list_pane"
            android:layout_width="0dip" android:layout_weight="1"
            android:layout_height="wrap_content" />
    </LinearLayout>
</LinearLayout>
```

Dynamic (Single-Pane) Layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <fragment
        android:name="com.example.android.fragments.HeadlinesFragment" />

    <fragment
        android:name="com.example.android.fragments.ArticleFragment" ... />

</LinearLayout>

<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Dynamic (Single-Pane) Layout

```
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.news_articles);

        if (findViewById(R.id.fragment_container) != null) {
            if (savedInstanceState != null) return;

            HeadlinesFragment firstFragment = new HeadlinesFragment();

            // Pass any Intent's extras to the fragment as arguments
            firstFragment.setArguments(getIntent().getExtras());

            getSupportFragmentManager()
                .beginTransaction()
                .add(R.id.fragment_container, firstFragment)
                .commit();
        }
    }
}
```

Dynamic Fragment Layout

- Fragment Transactions:
 - `beginTransaction`
 - `add`
 - `replace`
 - `commit`
- Back stack
 - `addToBackStack`

Dynamic (Single-Pane) Layout

```
ArticleFragment newFragment = new ArticleFragment();

Bundle args = new Bundle();
args.putInt(ArticleFragment.ARG_POSITION, position);
newFragment.setArguments(args);

FragmentTransaction transaction =
    getSupportFragmentManager().beginTransaction();

transaction.replace(R.id.fragment_container,
    newFragment);
transaction.addToBackStack(null);

transaction.commit();
```

EXAMPLE: PATIENT FRAGMENT

Single-Pane Layout

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android=
    "http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/fragment_index"
    android:name=".ui.IndexFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    tools:layout="@android:layout/list_content" />
```

Two-Pane Layout

```
<LinearLayout ...>

    <fragment
        android:id="@+id/fragment_index"
        android:name=".ui.IndexFragment"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        tools:layout="@android:layout/list_content" />

    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="3" />

</LinearLayout>
```

In res/values-large

```
<resources>

    <item name="activity_index" type="layout">
        @layout/activity_twopane
    </item>

</resources>

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_index);
}
```

Patient Activity

```
PatientsFragment editorFragment = new PatientsFragment();

private void startPatientEdit(Patient patient) {
    final Bundle args = new Bundle();
    args.putParcelable(PATIENT, patient);
    editorFragment.setArguments(args);

    getFragmentManager()
        .beginTransaction()
        .replace(R.id.fragment_container, editorFragment)
        .addToBackStack(ADDING_PATIENT_DIALOG)
        .commit();
}
```

Activity Callback

```
public interface IPatientModel {

    // Provide explicit callback or calls back to known fn
    public void requestClinics();

    public void update(Patient patient);

    public void exit();

    public void warning(String tag, int messageResid);

    public void confirm(String tag, int messageResid);

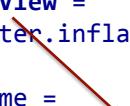
}
```

Patient Fragment

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    setHasOptionsMenu(true);  
}  
  
public void onAttach(Activity activity) {  
    super.onAttach(activity);  
  
    if (!(activity instanceof IPatientModel)) {  
        throw new IllegalStateException(...);  
    }  
    model = (IPatientModel) activity;  
}
```

Patient Fragment

```
public View onCreateView(LayoutInflater inflater,  
                        ViewGroup container,  
                        Bundle savedInstanceState) {  
    View rootView =  
        inflater.inflate(R.layout..., container, false);  
  
    patientName =  
        (EditText) rootView.findViewById(R.id.patient_name);  
    ...  
}
```



Patient Fragment

```
public void onActivityCreated (Bundle savedInstanceState) {  
    super.onActivityCreated(savedInstanceState);  
  
    final Activity context = getActivity();  
    ...  
    clinicsAdapter = new SimpleCursorAdapter(context,  
        R.layout.spinner_item,  
        null, from, to, 0);  
  
    clinicsAdapter.setDropDownViewResource(R.layout...);  
}  
}
```

Patient Fragment

```
public void onStart() {  
    super.onStart();  
    Bundle args = getArguments();  
    patient = args.getParcelable(PATIENT);  
    patientName.setName(patient.name);  
  
    // Initiate database retrievals or  
    // request data from the parent activity  
    model.requestClinics();  
}  
  
public void onReceiveClinics(List<Clinic> clinics) {  
    ...  
}
```

Patient Fragment

```

public void onCreateOptionsMenu(Menu menu,
                               MenuInflater inflater) {
    inflater.inflate(R.menu.patients_fragment, menu);
}

public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        ...
        case R.id.cancel:
            model.confirm(CONFIRM_CANCEL_DIALOG, ...);
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}

```

Patient Activity

```

private static final int CONFIRM_CANCEL_DIALOG = 1;

public void confirm(String tag, int messageResid) {
    Confirm.launch(this, tag, messageResid,
                  CONFIRM_CANCEL_DIALOG);
}

public void acknowledge(int dialogId, boolean confirm) {
    switch (dialogId) {
        case CONFIRM_CANCEL_DIALOG:
            if (confirm) {
                exit();
            }
            break;
    }
}

```

Patient Activity

```
public void exit() {
    getFragmentManager()
        .popBackStackImmediate(ADDING_PATIENT_DIALOG,
            FragmentManager.POP_BACK_STACK_INCLUSIVE);
}

public void update(Patient patient) {
    patientManager.update(patient);
}
```

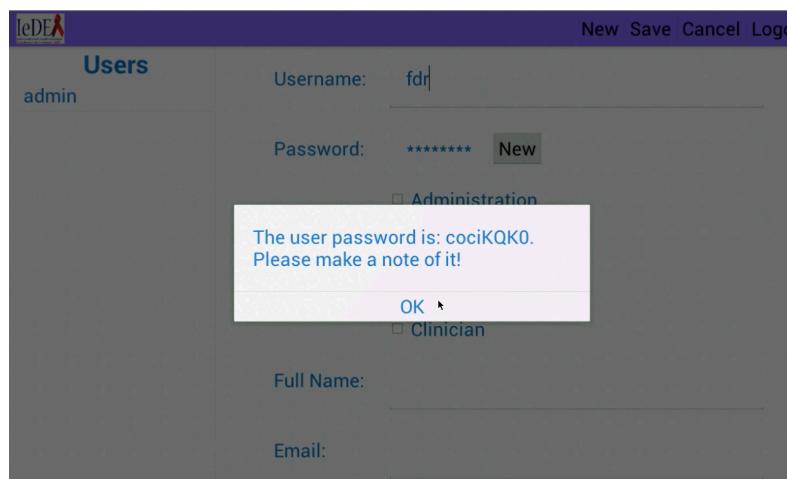
Patient Fragment

```
public void onDetach() {
    super.onDetach();
    model = dummyModel;
}

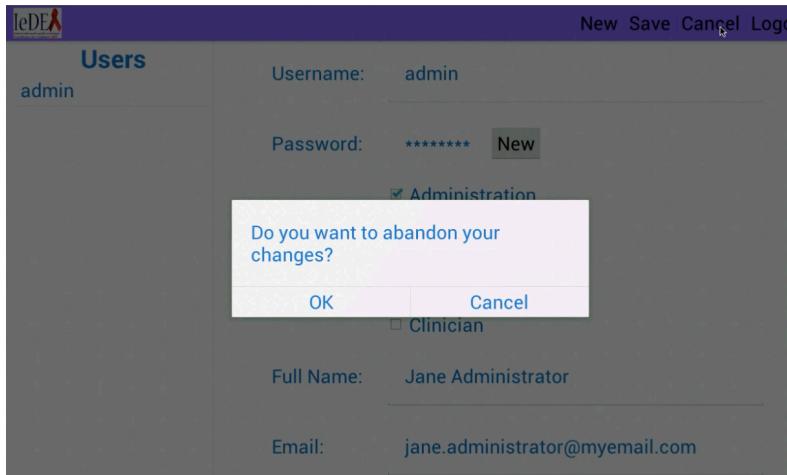
private IPatientModel dummyModel = new IPatientModel() {
    public void update(Patient patient) {
    }
    public void cancel() {
    }
    ...
};
```

DIALOGS

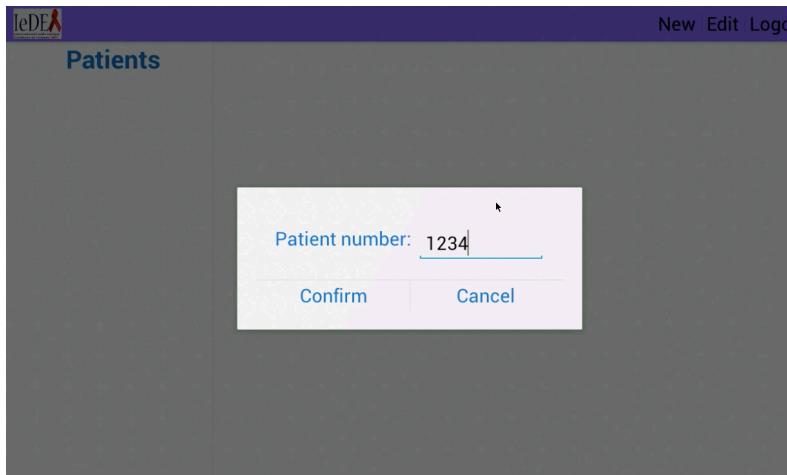
Alert Dialog



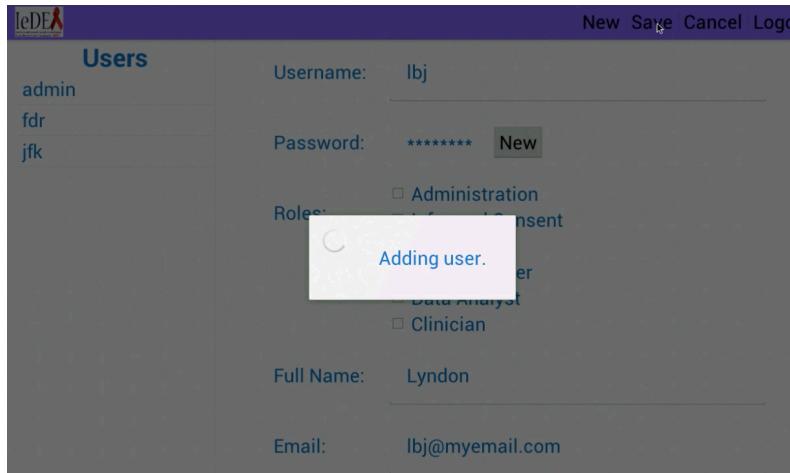
Confirmation Dialog



Data Entry Dialog



ProgressDialog



Example: Confirmation Dialog

```
public class Confirm extends DialogFragment {

    public static final String MESSAGE_KEY = "confirm_message";
    public static final String DIALOG_KEY = "dialog_id";

    public static void launch(Activity context, String tag,
                           int messageRid, int dialogId) {
        Confirm dialog = new Confirm();
        Resources res = context.getResources();
        Bundle args = new Bundle();
        args.putString(MESSAGE_KEY, res.getString(messageRid));
        args.putInt(DIALOG_KEY, dialogId);
        dialog.setArguments(args);
        dialog.show(context.getFragmentManager(), tag);
    }

    private IConfirmListener listener;

    private int dialogId;
```

Example: Confirmation Dialog

```
public interface IConfirmListener {
    public void acknowledge(int dialog, boolean confirm);
}

public void onAttach(Activity activity) {
    super.onAttach(activity);
    if (!(activity instanceof IConfirmListener)) {
        throw new IllegalStateException
            ("Activity must implement IConfirmListener.");
    }
    listener = (IConfirmListener) activity;
}

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    dialogId = getArguments().getInt(DIALOG_KEY);
}
```

Example: Confirmation Dialog

```
public View onCreateView(LayoutInflater inflater,
                        ViewGroup container,
                        Bundle savedInstanceState) {
    View rootView = inflater.inflate(R.layout.dialog_confirm,
                                     container, false);

    TextView message =
        (TextView) rootView
            .findViewById(R.id.dialog_confirm_message);
    message.setText(getArguments().getString(MESSAGE_KEY));

    confirm.setOnClickListener(confirmListener);
    cancel.setOnClickListener(cancelListener);

    return rootView;
}
```

Example: Confirmation Dialog

```
// Not much to do here
public Dialog onCreateDialog(Bundle savedInstanceState) {
    Dialog dialog = super.onCreateDialog(savedInstanceState);
    dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
    return dialog;
}
```

Alert Dialog

- Pro: Easy
- Con: Difficult to customize

```
public class ConfirmDialogFragment extends DialogFragment {
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder =
            new AlertDialog.Builder(getActivity());
        builder.setMessage(R.string.confirm_message)
            .setPositiveButton(R.string.yes,
                new DialogInterface.OnClickListener() { ... })
            .setNegativeButton(R.string.no,
                new DialogInterface.OnClickListener() { ... });
        return builder.create();
    }
}
```

APP NAVIGATION

App Navigation

- **Temporal** memory:
 - Remembering what **just** happened
 - Great for snapping back one context
 - Harder to sequence order of events
 - More potential for error, surprise
- **Structural** memory
 - Relationships between screens in an app
 - “Home” in web apps
 - Clearer expectations for behavior

App Navigation

- **SYSTEM BACK** navigates history between related screens
- **APPLICATION UP** navigates hierarchy within a single app

App Navigation

Contacts Task



Contacts

App Navigation

Contacts Task



App Navigation

Contacts Task



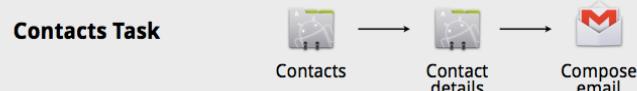
App Navigation



App Navigation



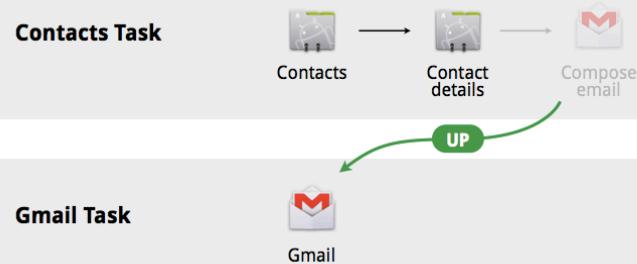
App Navigation



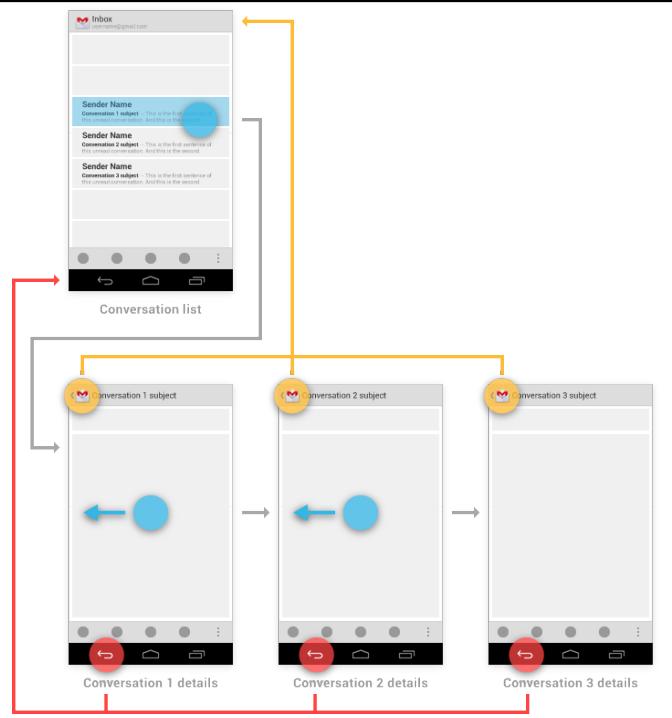
App Navigation



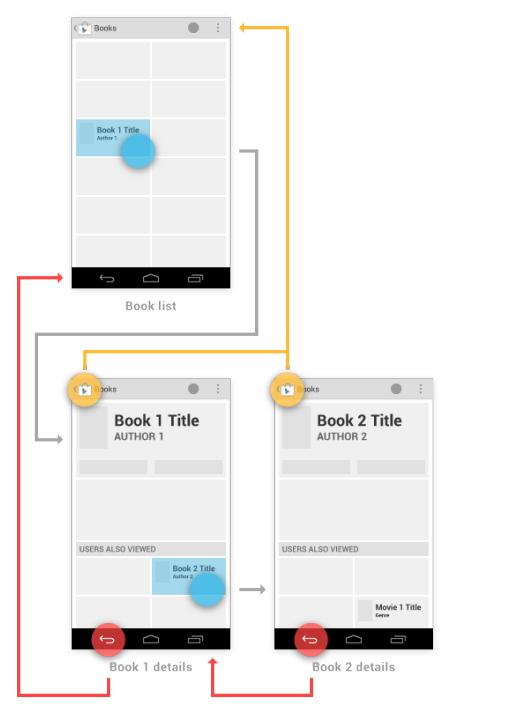
App Navigation

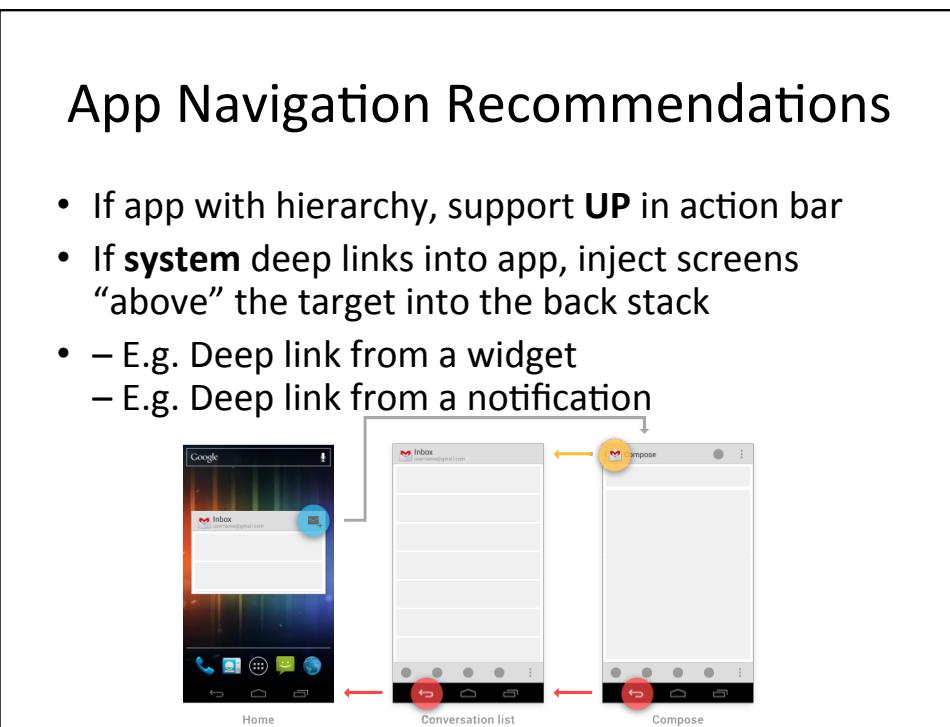
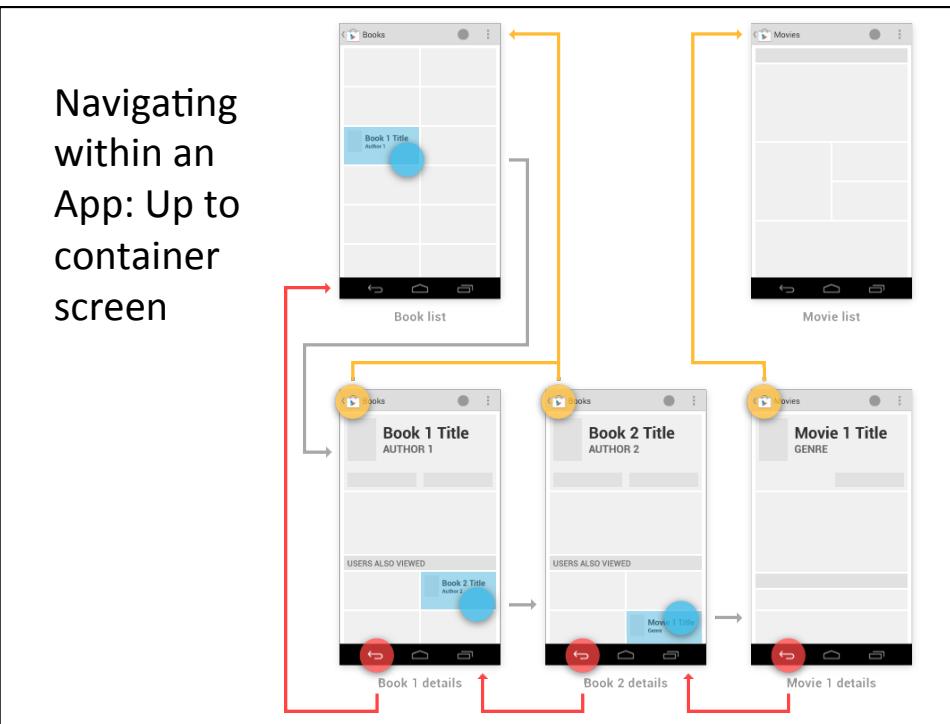


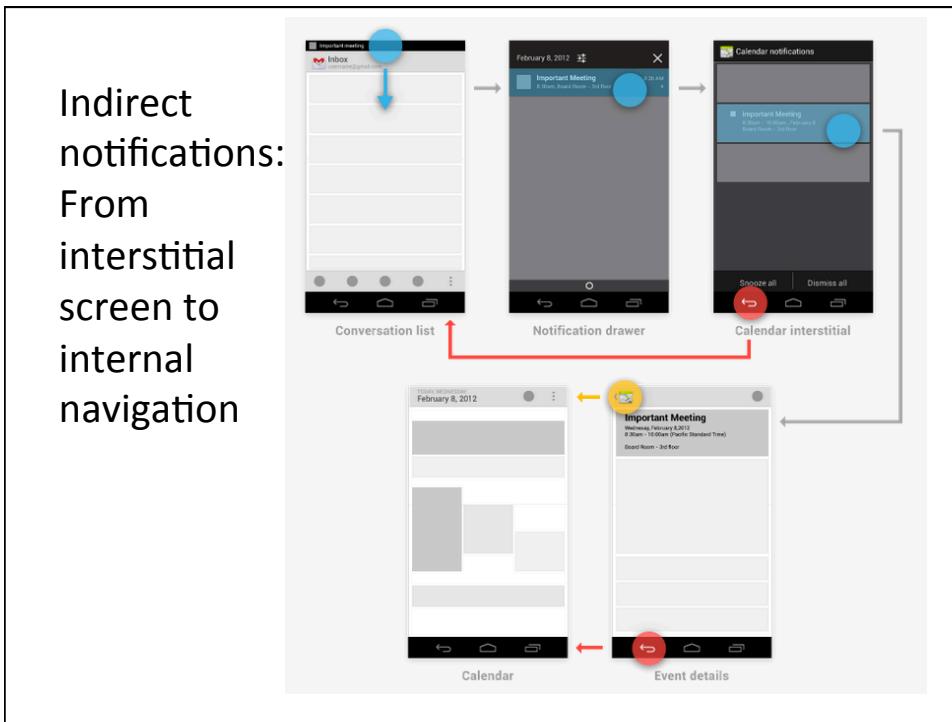
Navigating within an App: Sibling navigation with no history



Navigating within an App: Sibling navigation with history







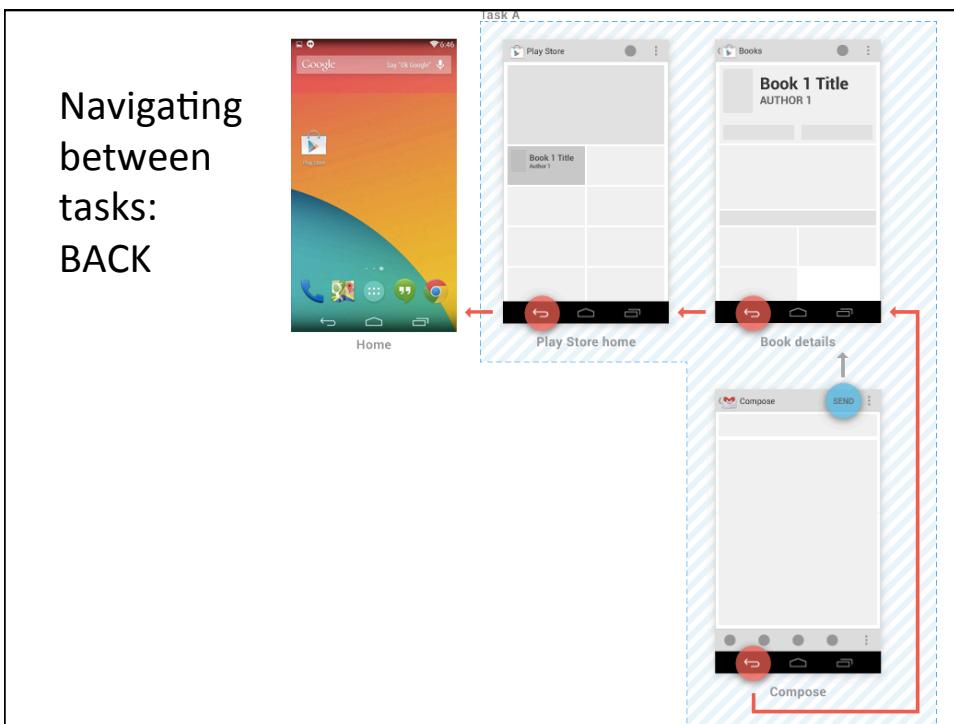
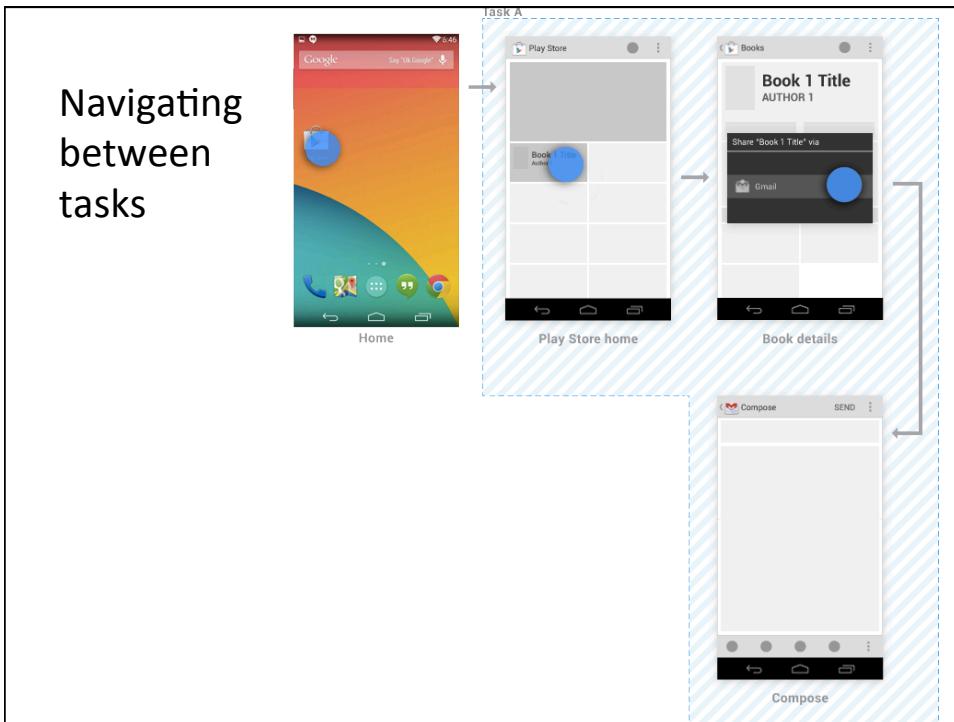
Synthetic Task Stack

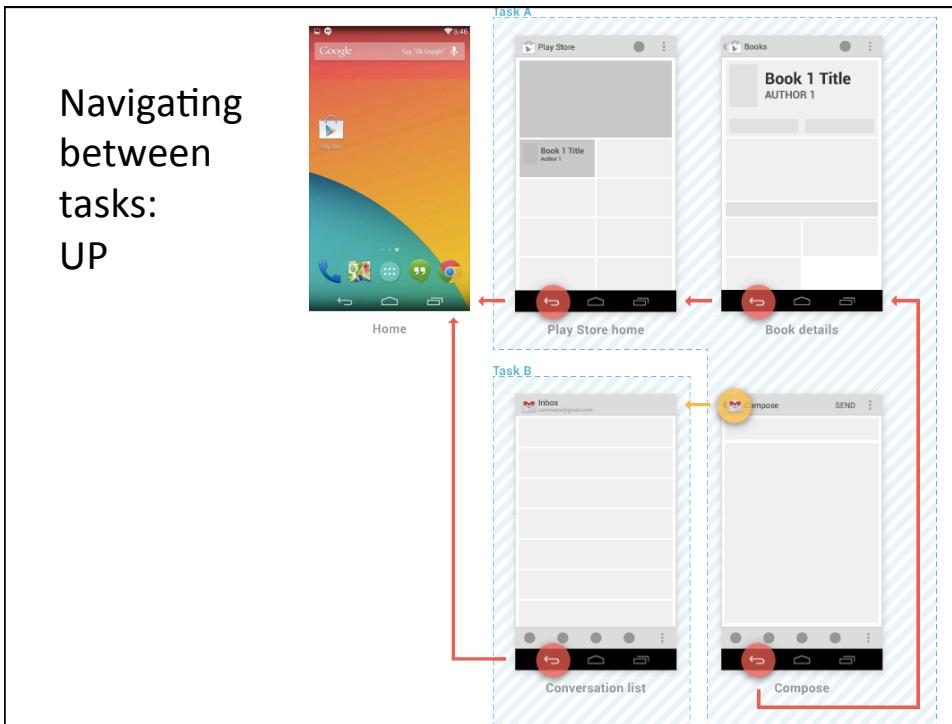
```

<activity
    android:name=".app.ContentActivity"
    android:label="Content"
    android:parentActivityName=".app.HomeActivity">
    <!-- ... -->
</activity>

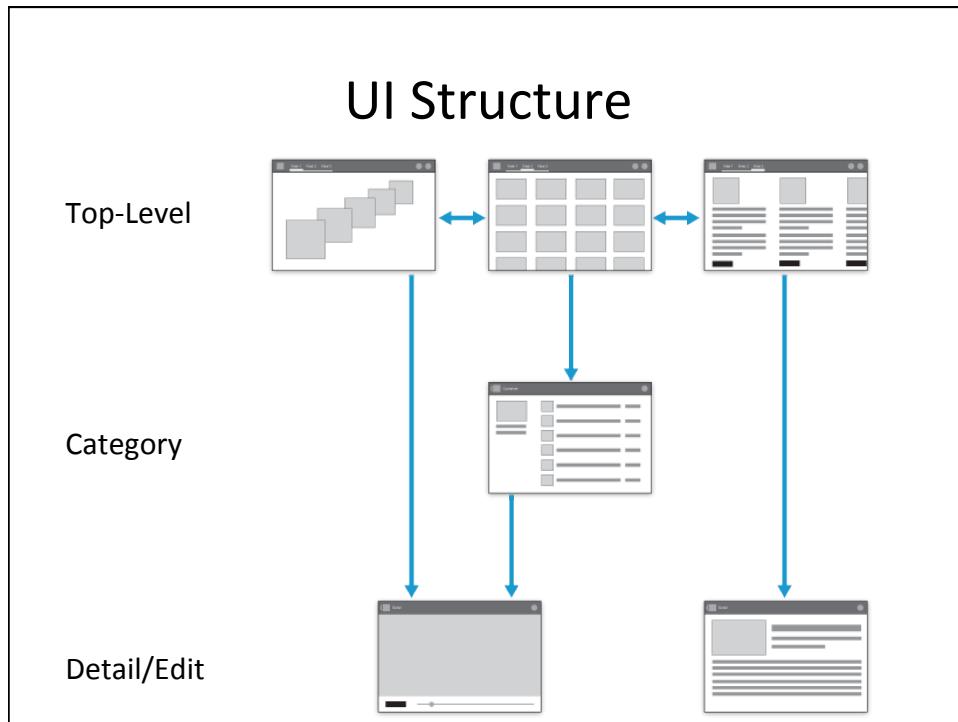
TaskStackBuilder.create(this)
    .addParentStack(ContentActivity.class)
    .addNextIntent(new Intent(this,
        ContentActivity.class)
        .putExtra(ContentActivity.EXTRA_TEXT,
        "From Notification"))
    .startActivities();

```





UI ARCHITECTURE



Top-Level

Bring content forward

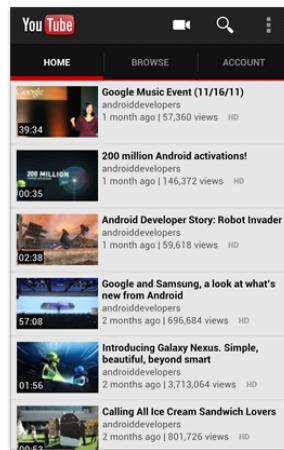
Create app identity

Action bar for navigation and actions

Categories

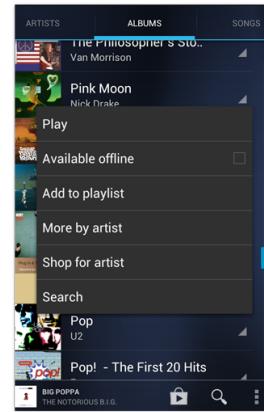
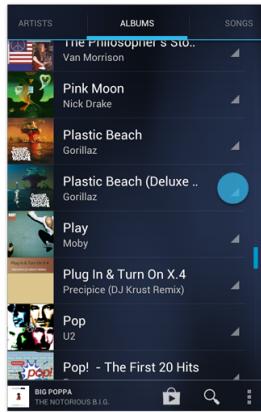


- Combining selection and data display
- **Scrolling tabs** (e.g. browsing related categories)



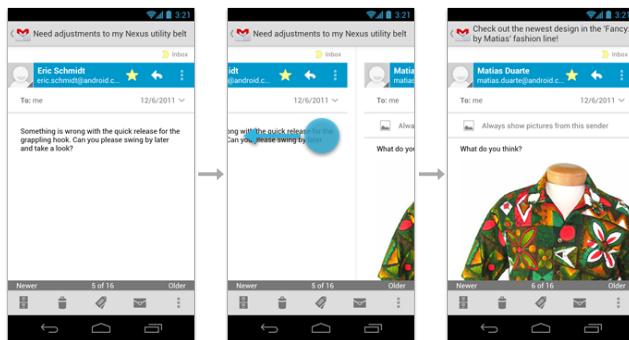
- Combining selection and data display
- **Fixed tabs** (unrelated functions)

Categories



Allow shortcut through hierarchy

Detail



Support efficient navigation between
Detail views (e.g. swiping)

UI Architecture

