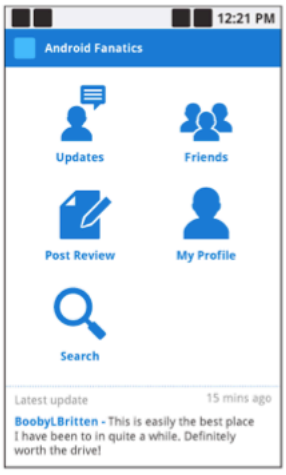
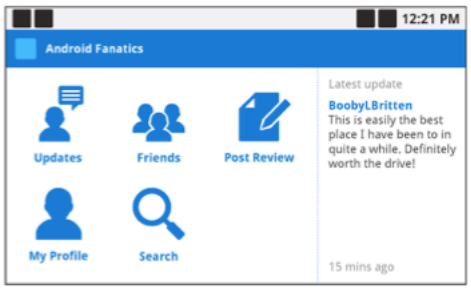
Android UI Design Patterns

Dominic Duggan
Stevens Institute of Technology

Dashboard



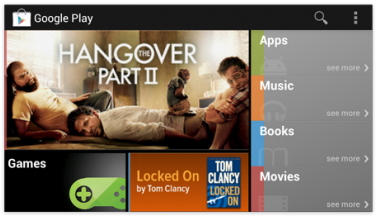


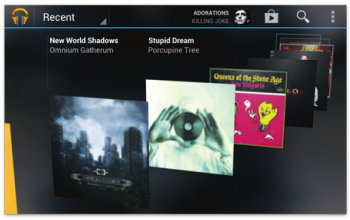
Dashboard

- *"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

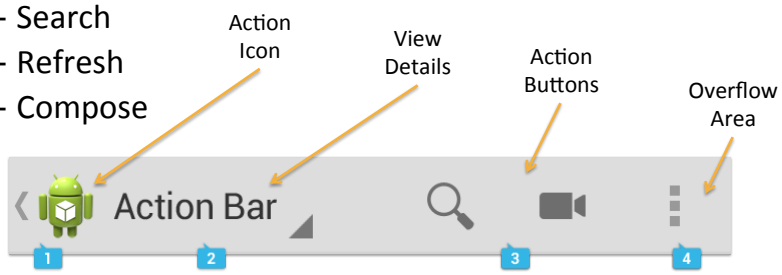
- DO highlight what's new
- DO focus on 3-6 most important choices
- DO be flavorful





Action Bar

- *"How can I do <common action> quickly?"*
- Dedicated real estate
  - Navigation
  - Frequently used operations
- Actions common across app
  - Search
  - Refresh
  - Compose

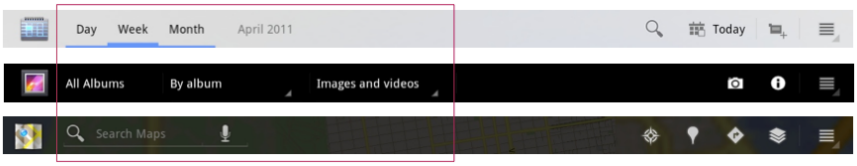


Action Bar

- Action Icon
  - Branding
  - Upward navigation

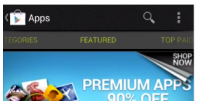
Action Bar

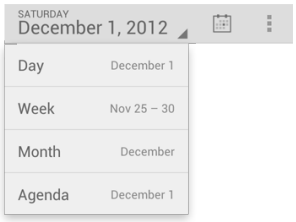
- View Details
  - Tabs (fixed or scrolling)
  - Drop-down Menus
  - Drawers

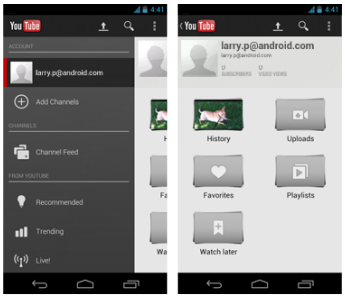


Action Bar

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



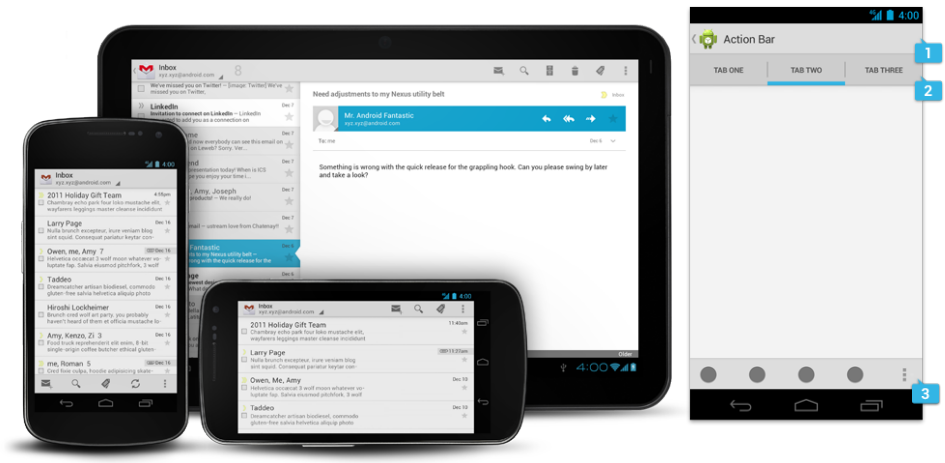




Action Bar

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

Action Bar on Smaller Screens

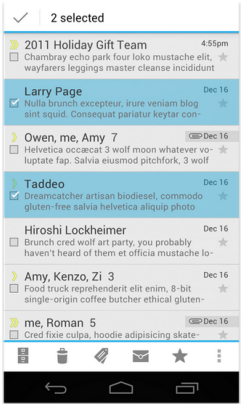


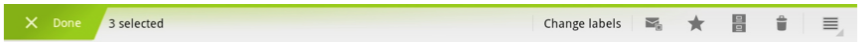
Action Bar Recommendations

- DO use to bring key actions onscreen
- DO help to convey a sense of place
- DO use consistently within your app

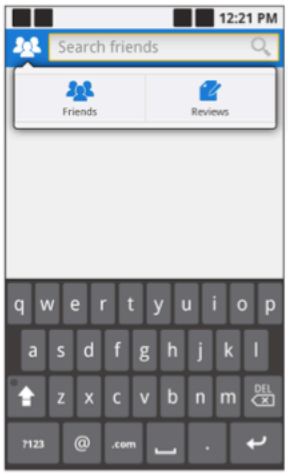
Contextual Action Bar

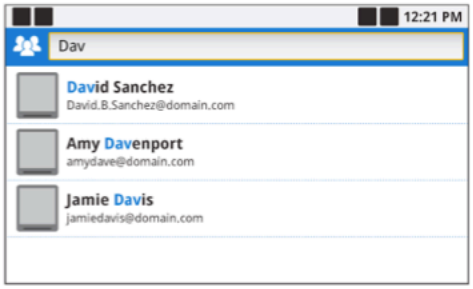
- When items are selected
  - Touch and hold
- Three sections
  - **Done** button (for releasing selection)
  - Selection details
  - Action Buttons
- Implemented using **ActionMode**





Search Bar





Search Bar Recommendations

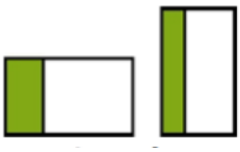
- DO use for simple searches
- DO present rich suggestions
- DO use the same behavior

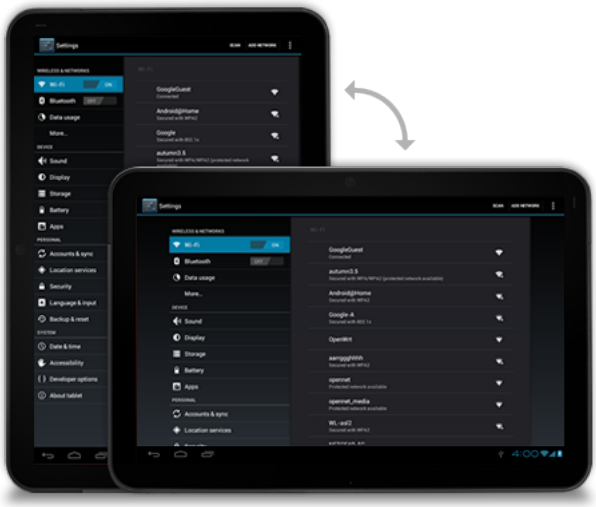
MULTI-PANE LAYOUTS

Multi-Pane Layouts

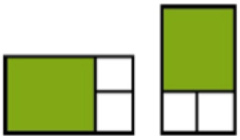
- Increased real estate
  - Give more context
  - Consolidate multiple related phone screens
- Selection on the left
- Content on the right

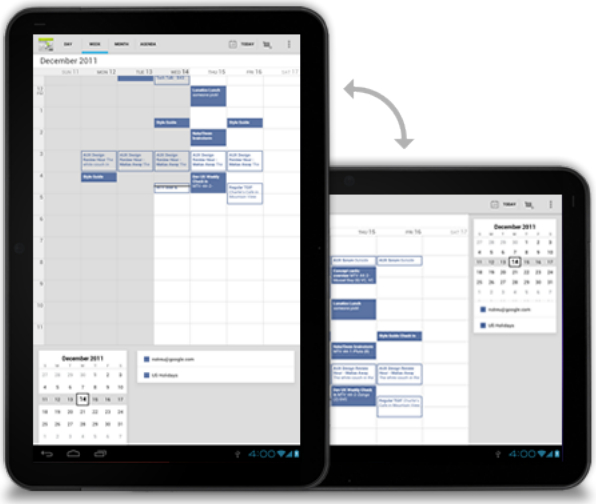
Stretch





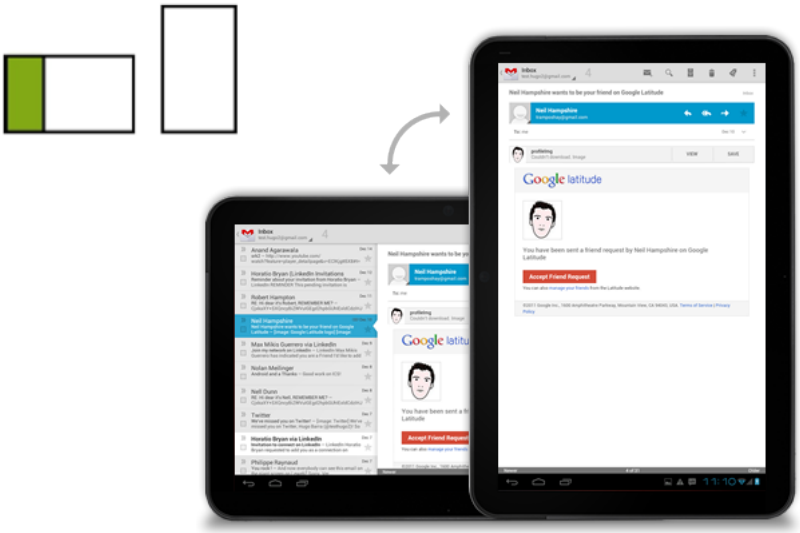
Stack





Expand/Collapse

Show/Hide



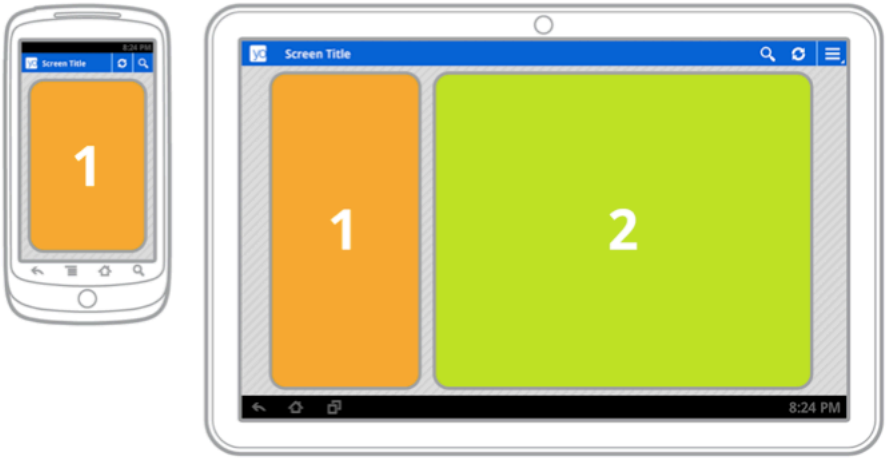
Strategies

- 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

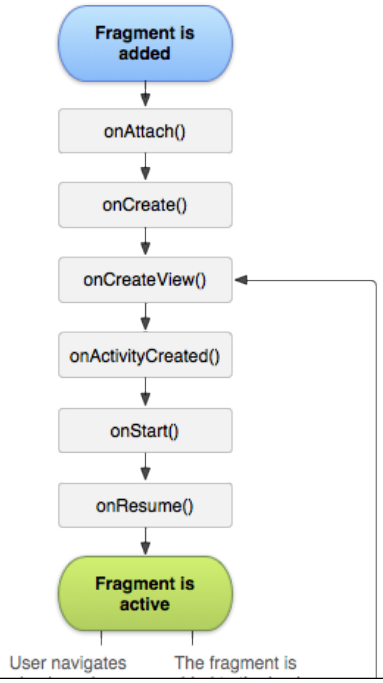


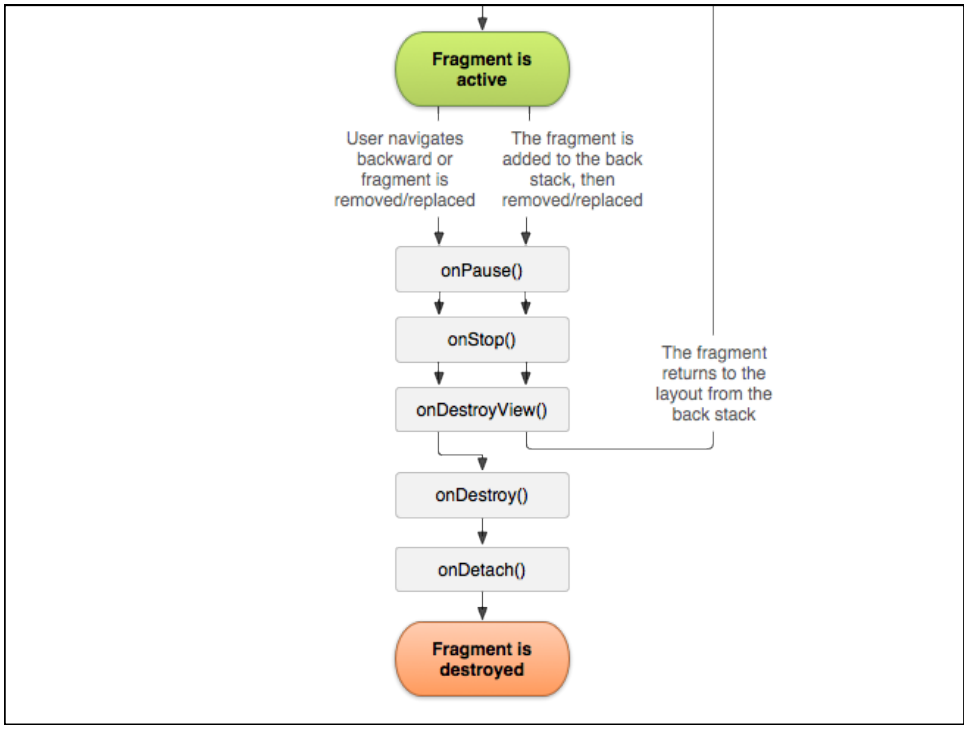
Layout with Fragments

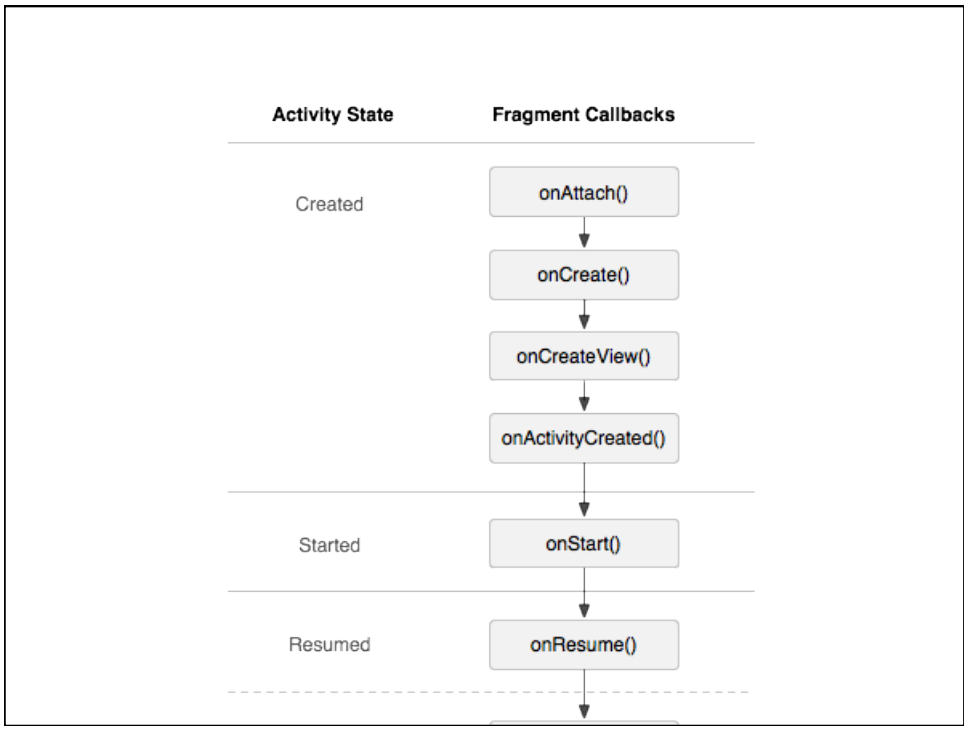
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <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>
```

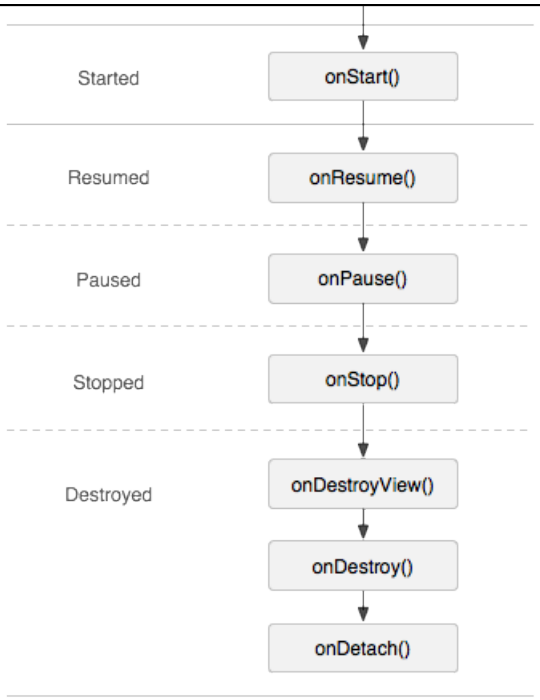
Layout with Fragments

```
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);
    }
}
```




Fragment is active
User navigates backward or fragment is removed/replaced
The fragment is added to the back stack, then removed/replaced
onPause()
onStop()
onDestroyView()
The fragment returns to the layout from the back stack
onDestroy()
onDetach()
Fragment is destroyed

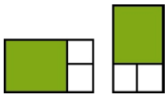

Activity State
Fragment Callbacks
Created
onAttach()
onCreate()
onCreateView()
onActivityCreated()
Started
onStart()
Resumed
onResume()



Resources

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

Multi-pane Layouts via Resources



Stack
(e.g. Calendar)

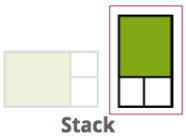
Multi-pane Layouts via Resources

Stack
(e.g. Calendar)

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



Stack
(e.g. Calendar)

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="0dip" android:layout_weight="1"
        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>
```

Layout with Fragments

```
<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" />
```

Layout with Fragments

```
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());

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

Layout with Fragments

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

Layout with Fragments

```
ArticleFragment newFragment = new ArticleFragment();

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

FragmentTransaction transaction =
              getFragmentManager().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: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="org.medea.main.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

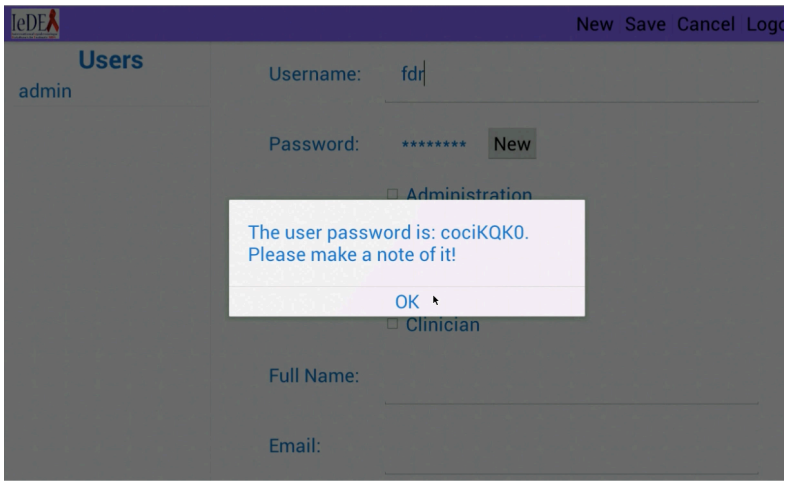
```
public void exit() {
   getFragmentManager()
      .popBackStackImmediate(ADDING_PATIENT_DIALOG,
                  FragmentManager.POP_BACK_STACK_INCLUSIVE);
}

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

Patient Fragment

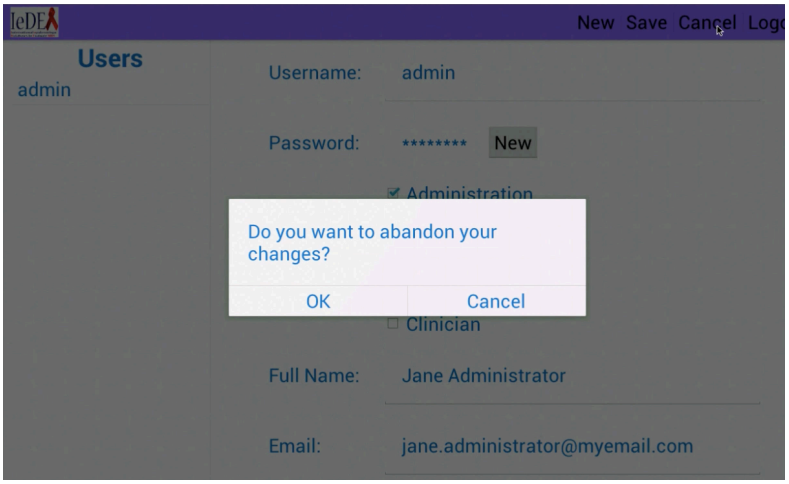
```
public void onDetach() {
   super.onDetach();
   model = dummyModel;
}

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

DIALOGS

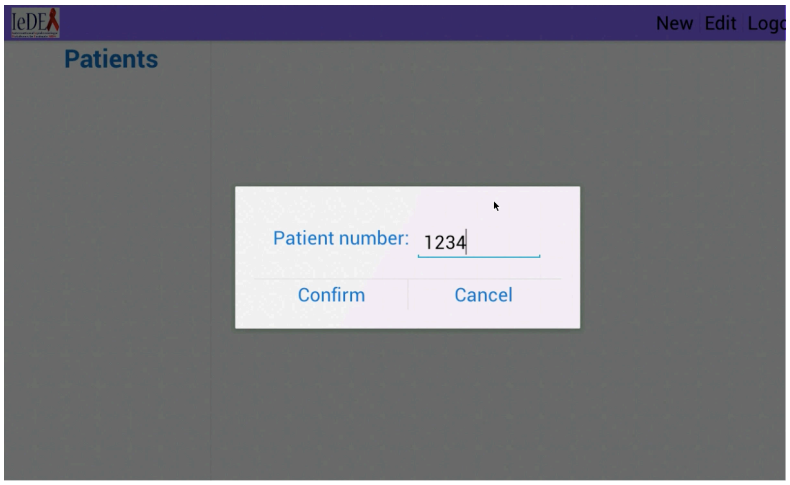
Alert Dialog



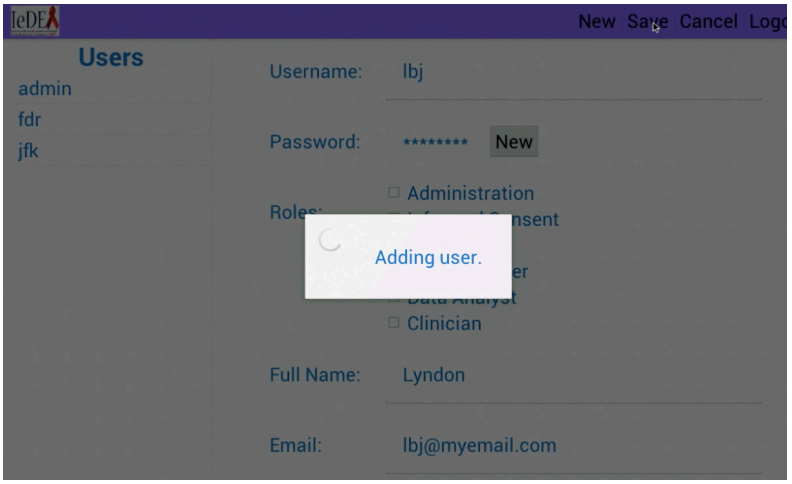
Confirmation Dialog



Data Entry Dialog



Progress Dialog



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

- T**emporal** memory:
  - Remembering what **just** happened
  - Great for snapping back one context
  - Harder to sequence order of events
    - More potential for error, surprise
- S**tructural** 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

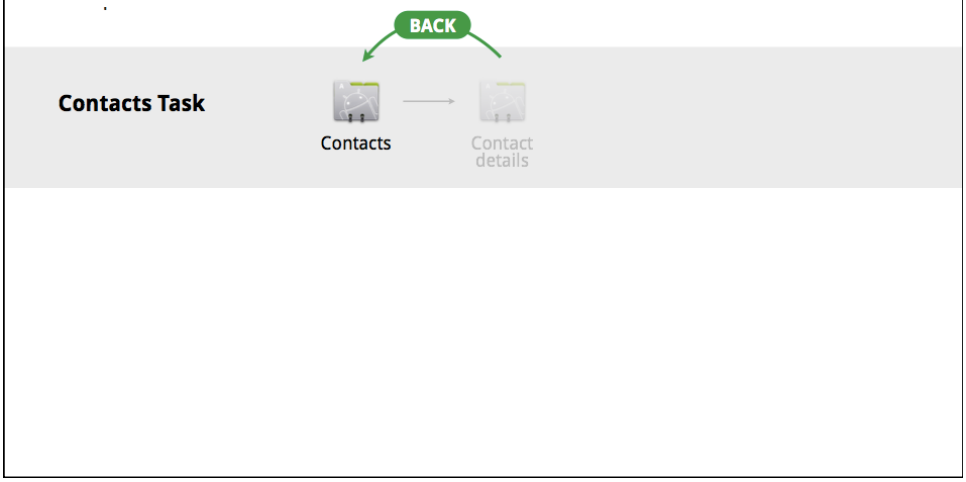
App Navigation

Contacts Task

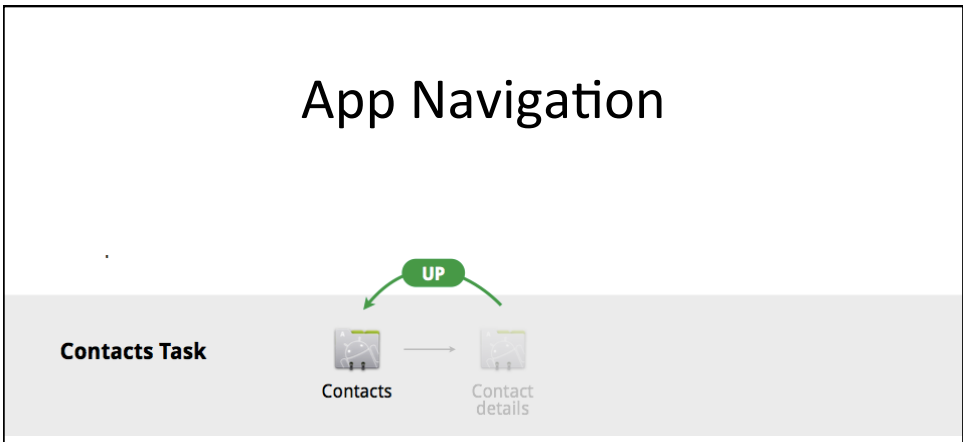
Contacts

Contact details

App Navigation



App Navigation



App Navigation

Contacts Task

Contacts

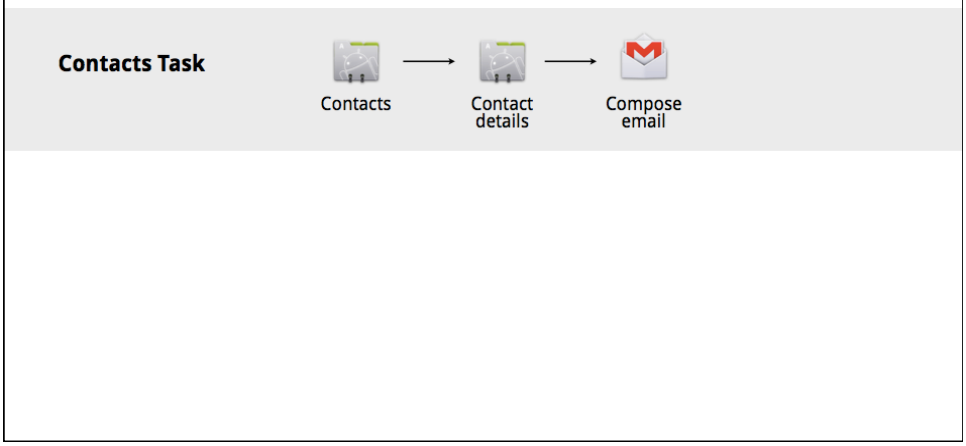
App Navigation

Contacts Task

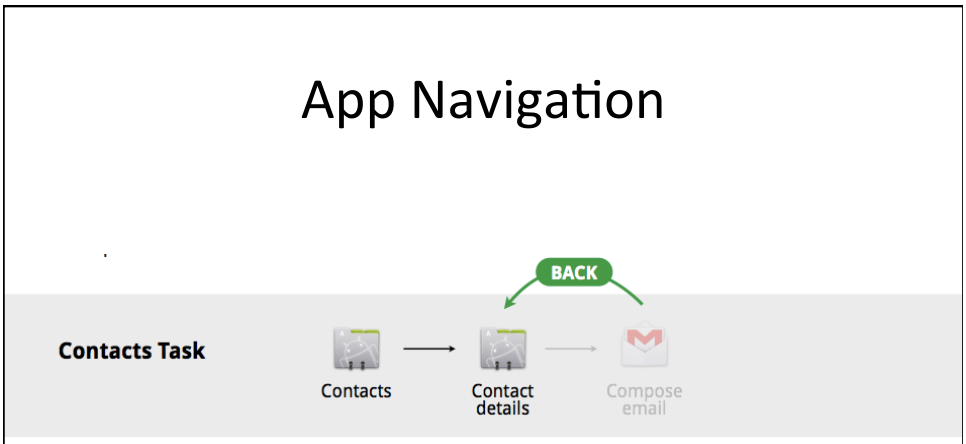
Contacts

Contact details

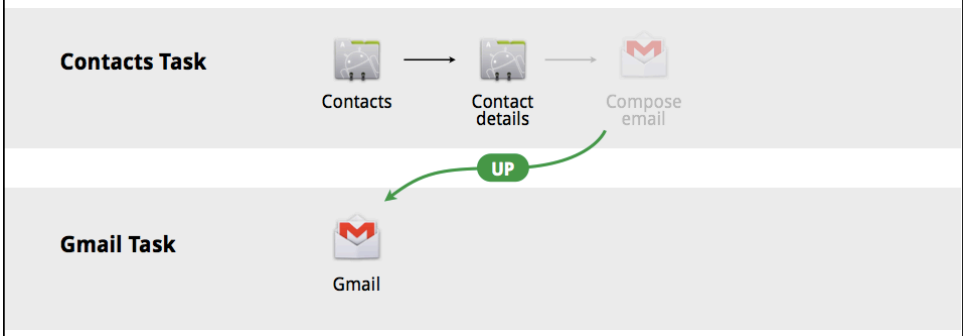
App Navigation



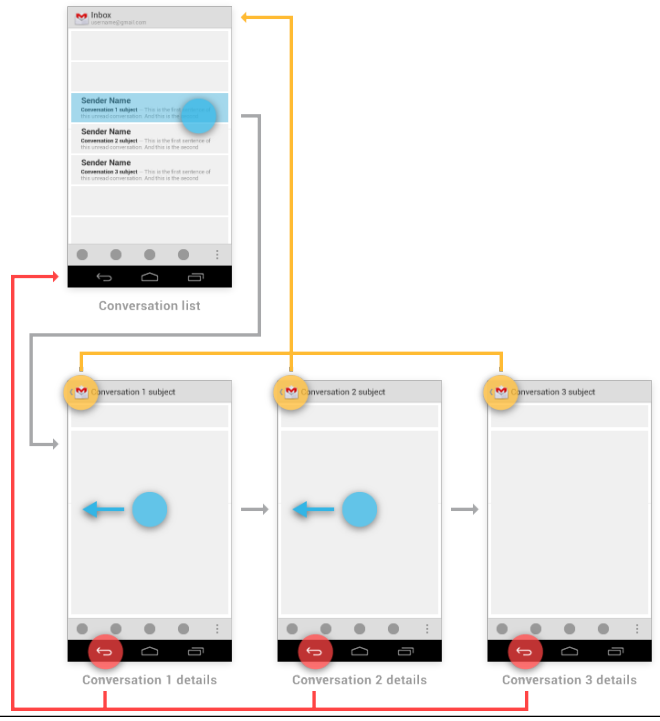
App Navigation



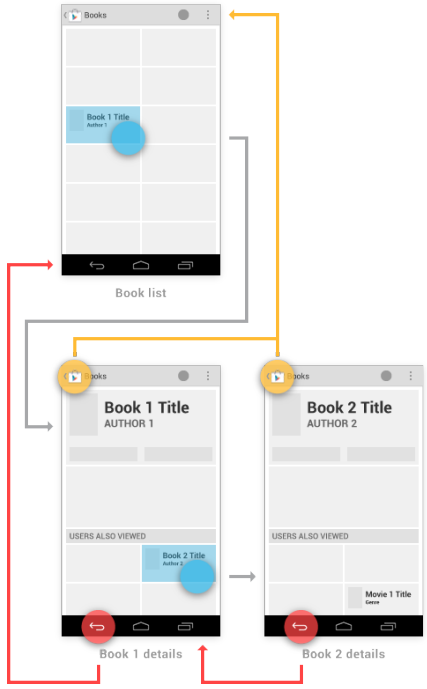
App Navigation



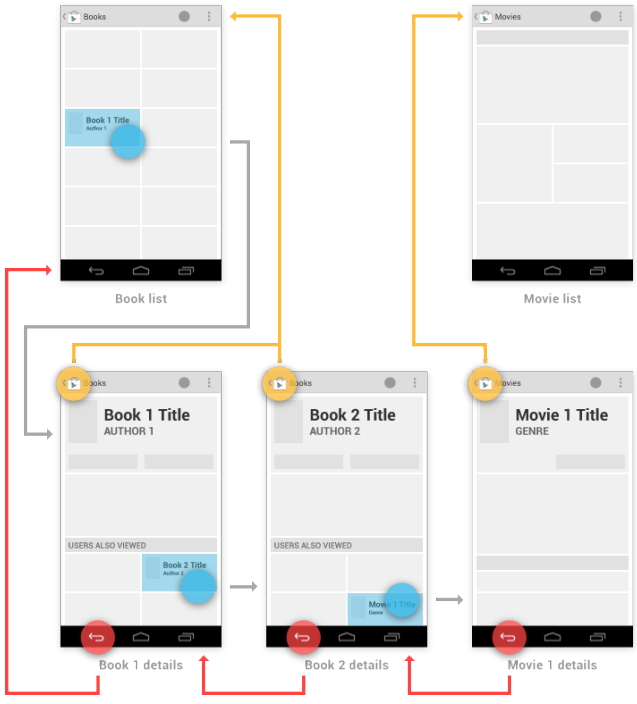
Navigating within an App: Sibling navigation with no history



Navigating within an App: Sibling navigation with history

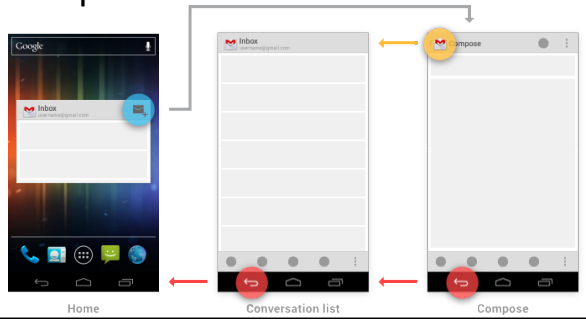


Navigating within an App: Up to container screen

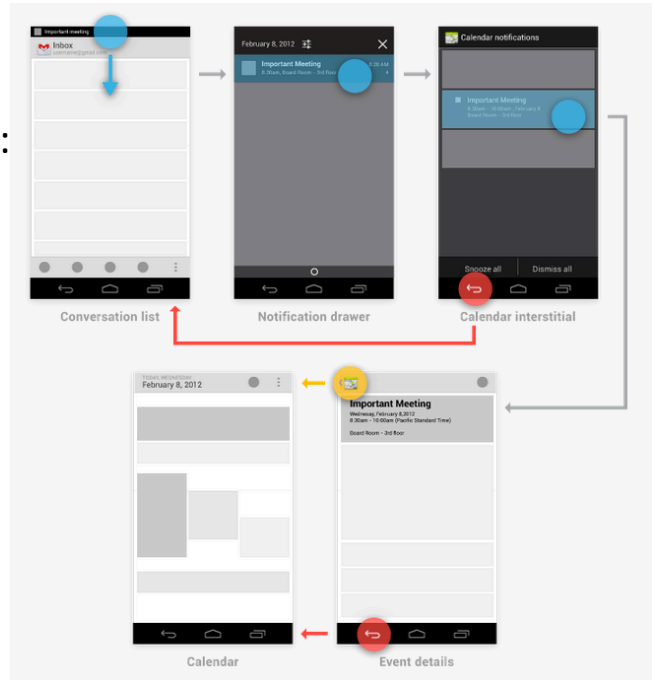


App Navigation Recommendations

- If app with hierarchy, support **UP** in action bar
- If **system** deep links into app, inject screens "above" the target into the back stack
- – E.g. Deep link from a widget
  – E.g. Deep link from a notification



Indirect notifications: From interstitial screen to internal navigation



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();
```

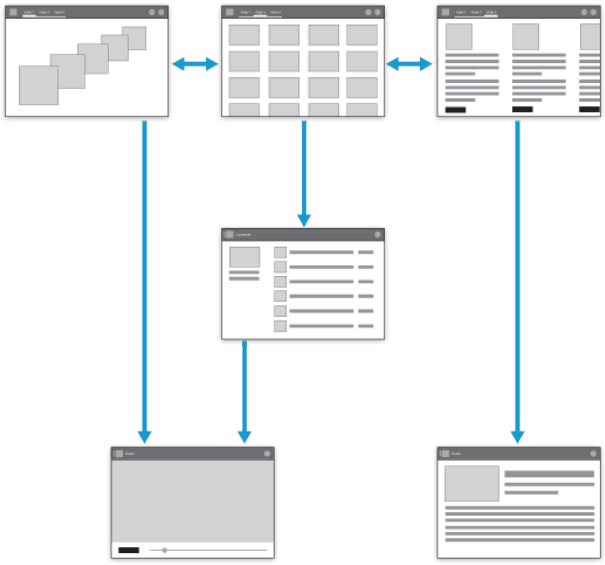
UI ARCHITECTURE

UI Structure

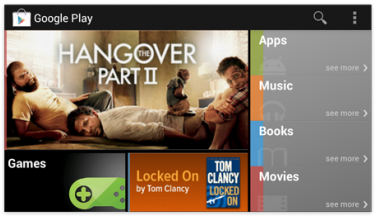
Top-Level

Category

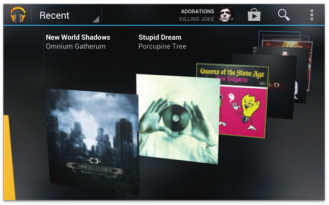
Detail/Edit



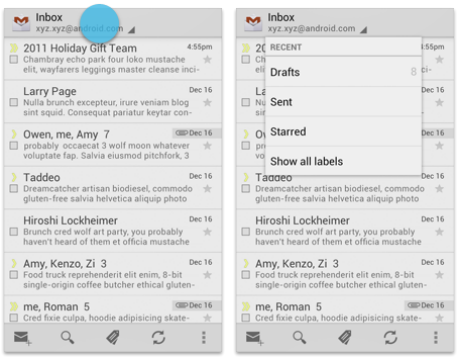
Top-Level



Bring content forward

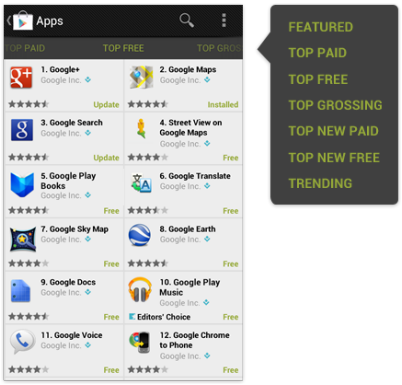


Create app identity



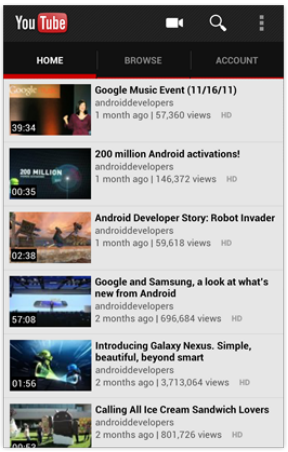
Action bar for navigation and actions

Categories



Combining selection and data display

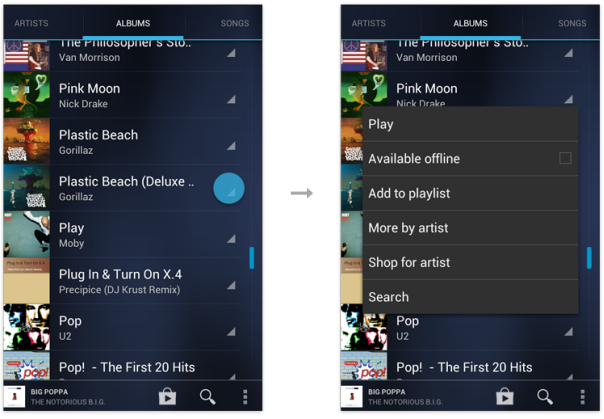
- Scrolling tabs (e.g. browsing related categories)



Combining selection and data display

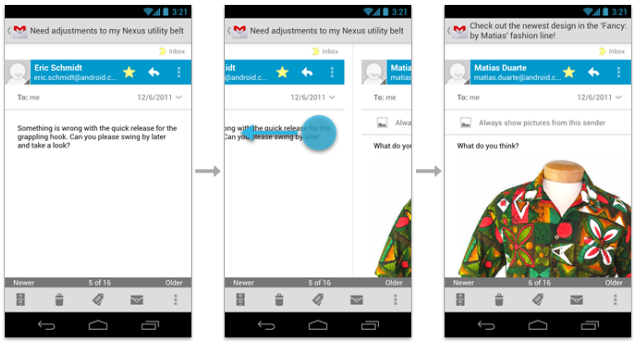
- Fixed tabs (unrelated functions)

Categories



Allow shortcut through hierarchy

Detail



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

UI Architecture