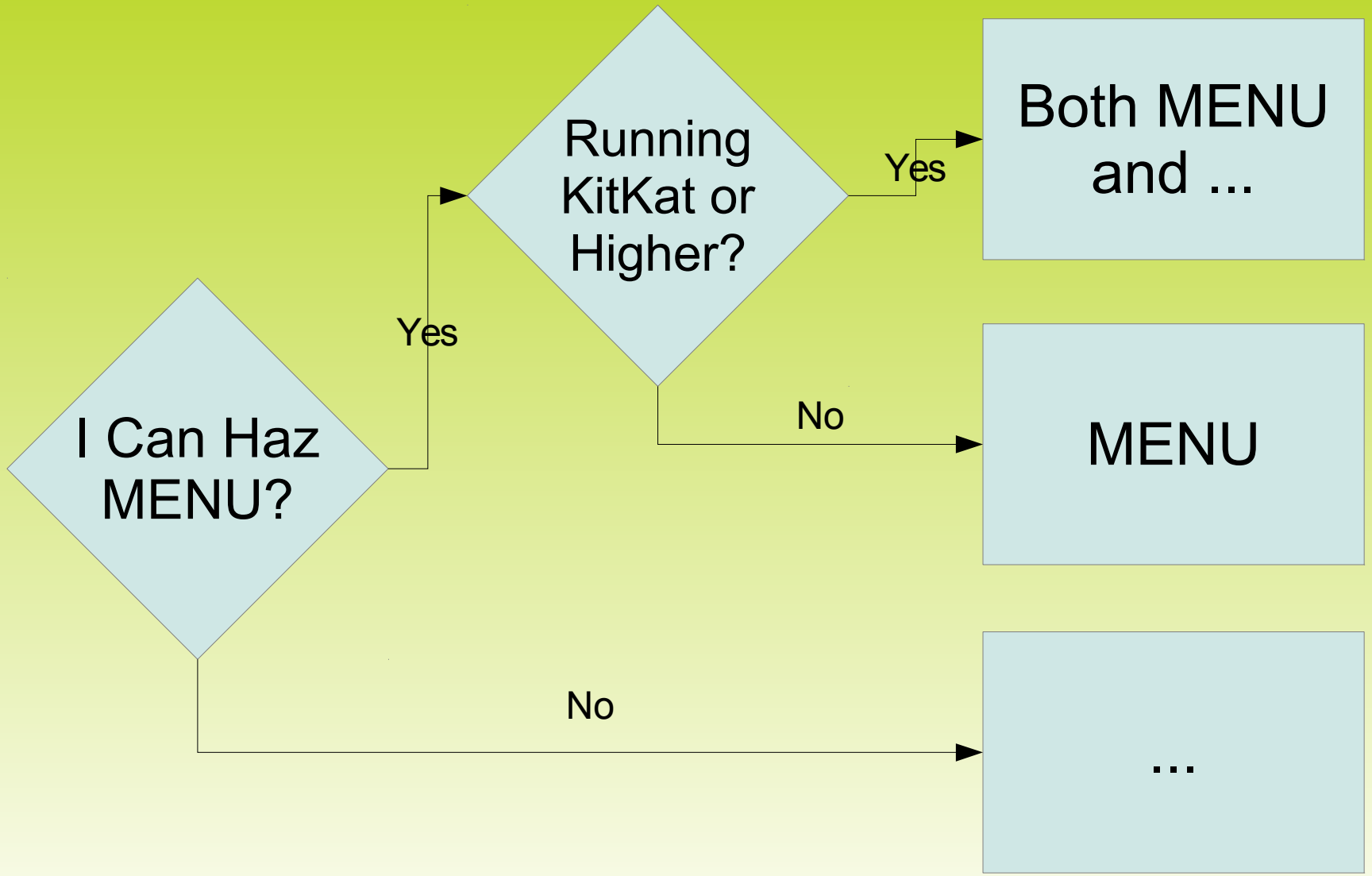
Advanced Action Bar

What You Should Already Know

- Native Implementation + Backports
- Key Manifest Bits
- Menu Resources
- `onCreateOptionsMenu()`/ `onOptionsItemSelected()`
- Overflow
- Adding Basic Toolbar Buttons

Overflow Trigger



Styles and Themes

- Theme.Holo / Theme.Holo.Light
  - Standard themes, standard color scheme
- Can Style the Action Bar
  - Action Bar Style Generator
    - http://jgilfelt.github.io/android-actionbarstylegenerator

Android Action Bar Style Generator

<< Android Asset Studio

The **Android Action Bar Style Generator** allows you to easily create a simple, attractive and seamless custom action bar style for your Android application. It will generate all necessary nine patch assets plus associated XML drawables and styles which you can copy straight into your project.

Style name
Used as name suffix when generating resources. No spaces or punctuation.

example

Style compatibility
Sherlock styles require the ActionBarSherlock library.

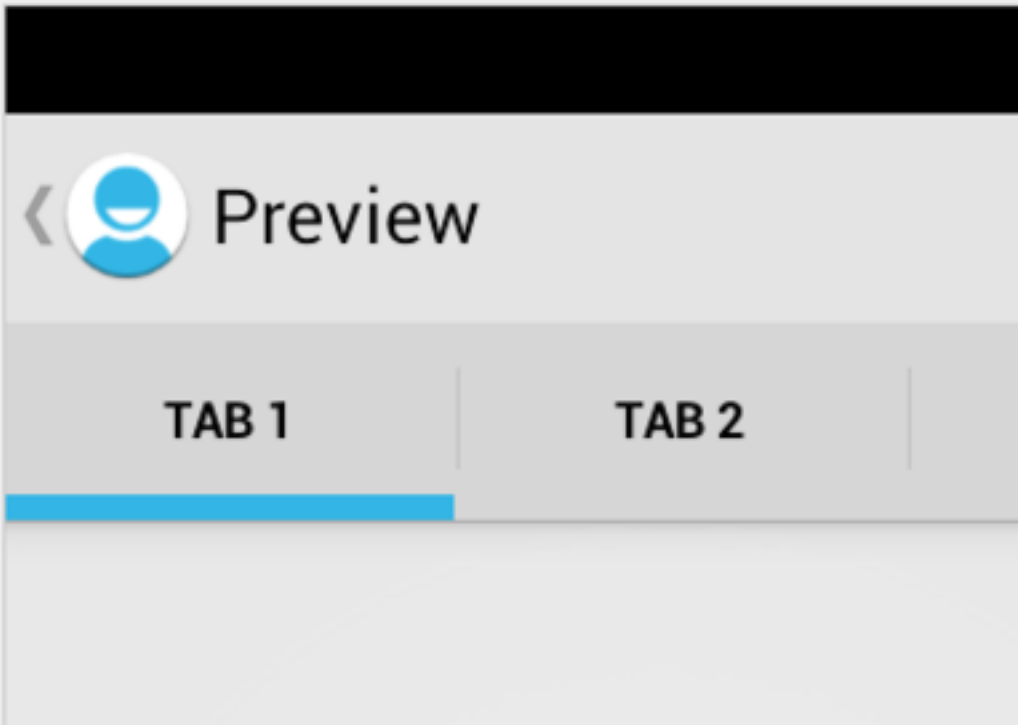
Holo

Base theme
The application base theme. Changing this setting will reset the form to Android defaults.

Light

Action bar style

Solid | Transparent



Action bar texture
Solid style action bar texture effect on API 14+.

On | Off

Tab hairline style

On | Off

Neutral pressed states
Default KitKat style.

On | Off

Action bar color
Solid style action bar background.

Stacked color
Solid style stacked tab background.

Tab indicator color

Popup color
Overflow menu, submenu and spinner panel background.

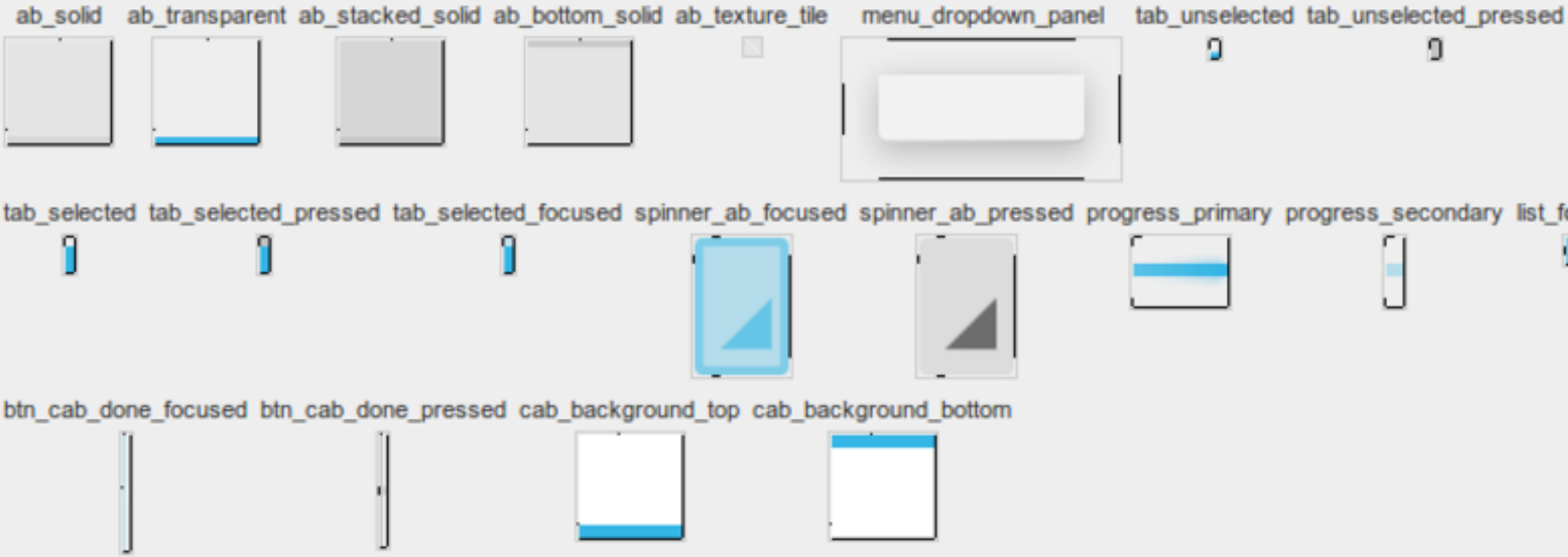
Accent color
Focus & pressed states, horizontal progress and

Action mode highlight color

Contextual action bar highlight.

Output resources

DOWNLOAD .ZIP

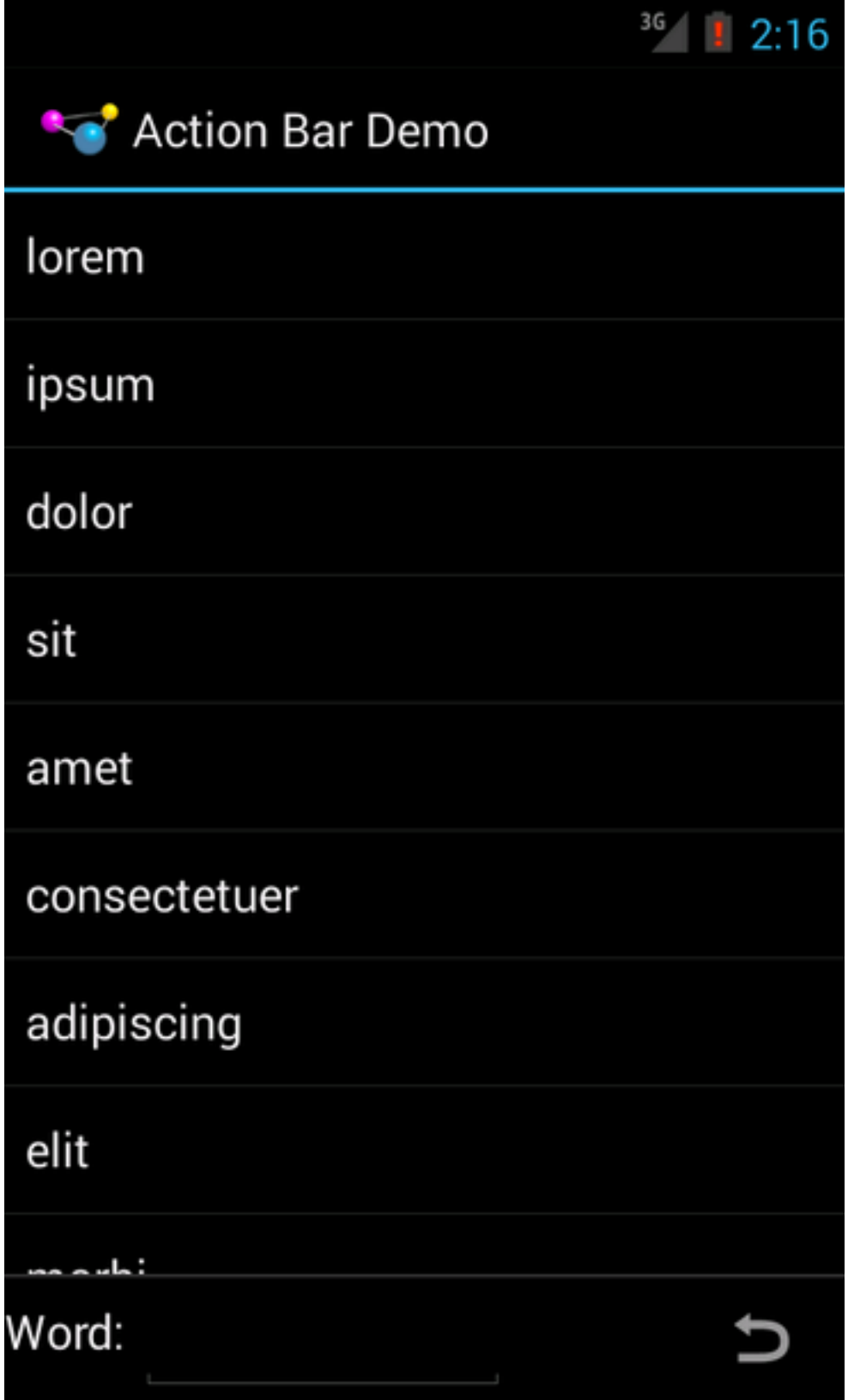


Custom Action Bar Widgets

- Option #1: Substitute Own Inflated Layout for Standard Button
  - Add `android:actionLayout` to `<item>` in menu resource
  - Call `getActionView()` on `MenuItem` to configure at runtime

```
<item
  android:id="@+id/add"
  android:actionLayout="@layout/add"
  android:icon="@android:drawable/ic_menu_add"
  android:showAsAction="ifRoom"
  android:title="@string/add"/>
```

```
private void configureActionItem(Menu menu) {
  EditText add=
      (EditText)menu.findItem(R.id.add).getActionView()
                    .findViewById(R.id.title);

  add.setOnEditorActionListener(this);
}
```


3G
2:16
Action Bar Demo
lorem
ipsum
dolor
sit
amet
consectetuer
adipiscing
elit
Word:

Custom Action Bar Widgets

- Option #2: `android:actionViewClass`
  - Skip the layout, directly reference a View class
  - Often implements CollapsibleActionView interface
    - Allows automatic expansion to fill available space or collapse to allow other action bar items to be seen
  - Built-In: SearchView

Custom Action Bar Widgets

- Option #3: ActionProvider
  - Extend ActionProvider, implement `onCreateActionView()`
  - Wire in via `android:actionProviderClass` in menu resource
  - Supports overflow with simplified UI
  - Built-in
    - ShareActionProvider
    - MediaRouteActionProvider

ShareActionProvider

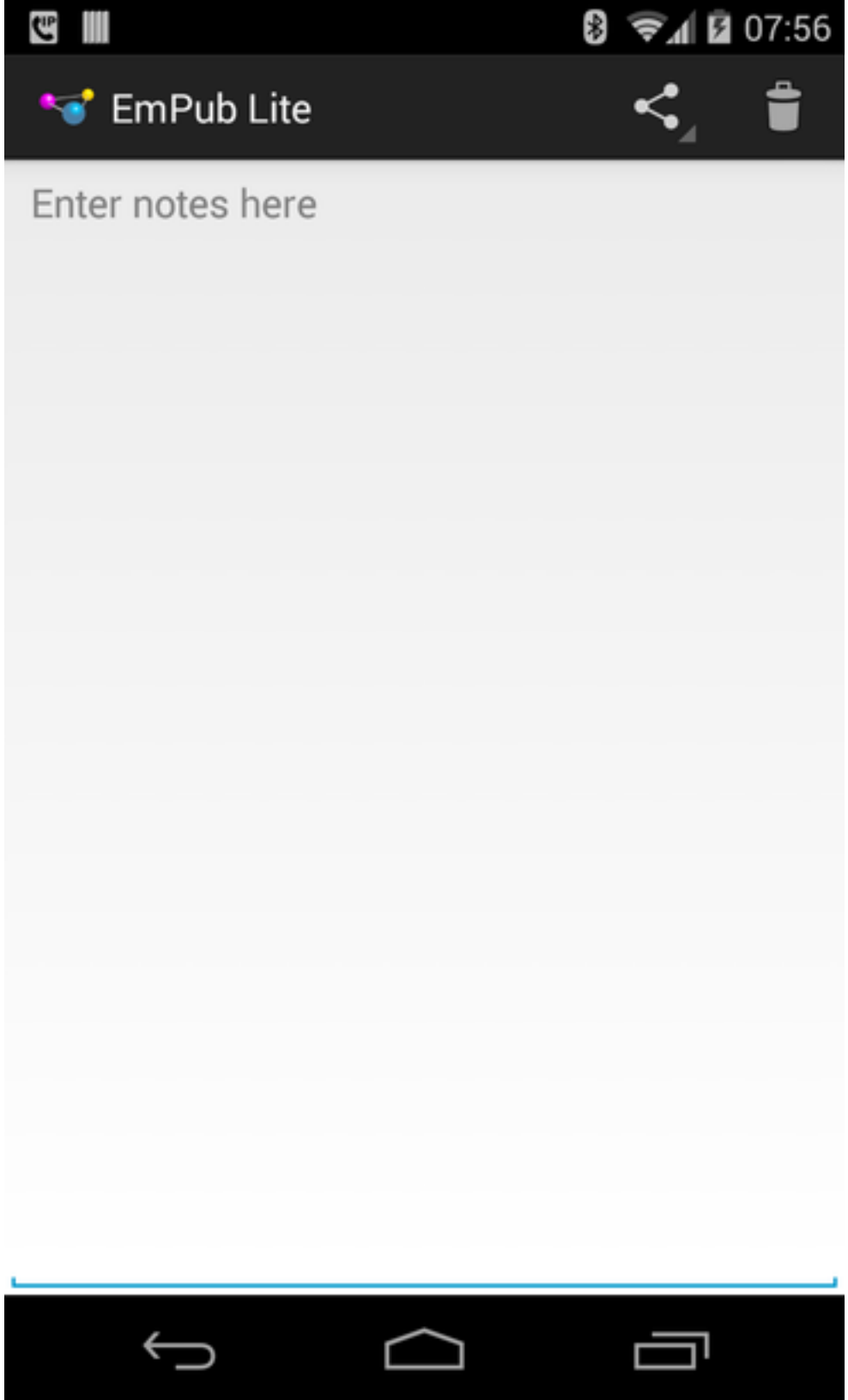
- Stock Way to Share Content
- Step #1: Add to `<menu>`
- Step #2: Call `setShareIntent()`
  - Once or many times, as appropriate
  - Be sure to set MIME type!
- Optional
  - Control share history
  - Register OnShareTargetSelectedListener, to update UI

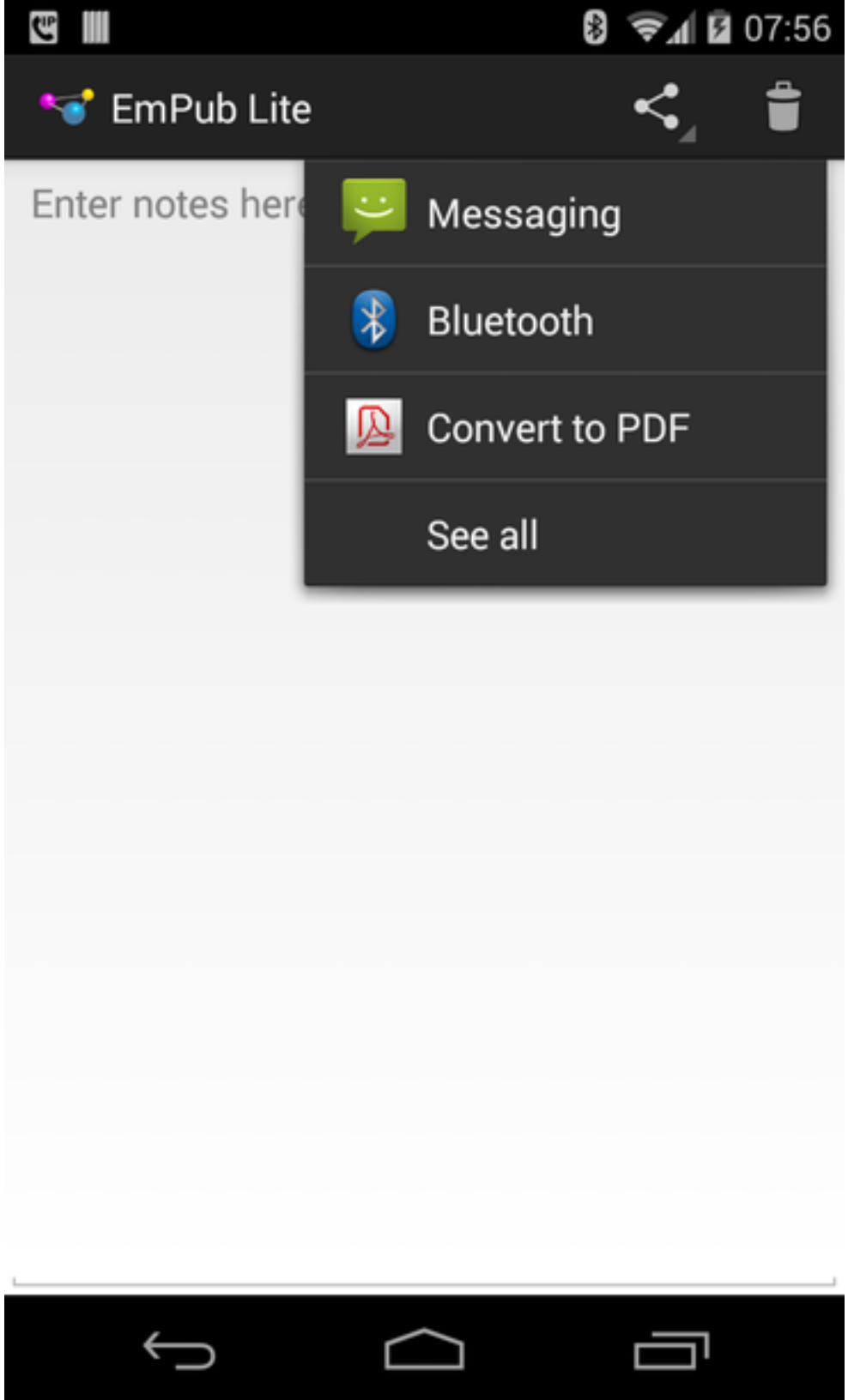
```
<item
  android:id="@+id/share"
  android:actionProviderClass="android.widget.ShareActionProvider"
  android:showAsAction="ifRoom"/>
```

```
private ShareActionProvider share=null;
private Intent shareIntent=new Intent(Intent.ACTION_SEND);
private EditText editor=null;

@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);
  setContentView(R.layout.activity_main);

  shareIntent.setType("text/plain");
  editor=(EditText)findViewById(R.id.editor);
  editor.addTextChangedListener(this);
}
```

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  getMenuInflater().inflate(R.menu.actions, menu);

  share=
      (ShareActionProvider)menu.findItem(R.id.share)
                               .getActionProvider();
  share.setOnShareTargetSelectedListener(this);

  return(super.onCreateOptionsMenu(menu));
}

@Override
public boolean onShareTargetSelected(ShareActionProvider source,
                                     Intent intent) {
  Toast.makeText(this, intent.getComponent().toString(),
                 Toast.LENGTH_LONG).show();

  return(false);
}
```

```
@Override
public void afterTextChanged(Editable s) {
  shareIntent.putExtra(Intent.EXTRA_TEXT, s.toString());
  share.setShareIntent(shareIntent);
}

@Override
public void beforeTextChanged(CharSequence s, int start, int count,
                              int after) {
  // ignored
}

@Override
public void onTextChanged(CharSequence s, int start, int before,
                          int count) {
  // ignored
}
```


07:56
EmPub Lite
Enter notes here


07:56
EmPub Lite
Enter notes here
Messaging
Bluetooth
Convert to PDF
See all

MediaRouteActionProvider

- User-Selectable Media Routes
  - External displays
  - Chromecast and kin
- Three Implementations
  - The native one (which Google no longer likes)
  - The AppCompat one
  - The cross-port of the AppCompat one to the native action bar (written by some balding guy)

SearchView

- The Classic Magnifying Glass
- Approaches
  - Iconified by default, expanding on click
  - Expanded by default
    - Good for tablets, particularly in landscape

Basic SearchView Usage

- Step #1: Add to <menu>
- Step #2: Configure in onCreateOptionsMenu()
  - Register listeners
    - OnQueryTextListener
    - OnCloseListener
  - Other settings
- Step #3: Respond to Events
  - E.g., manage a ListView filter

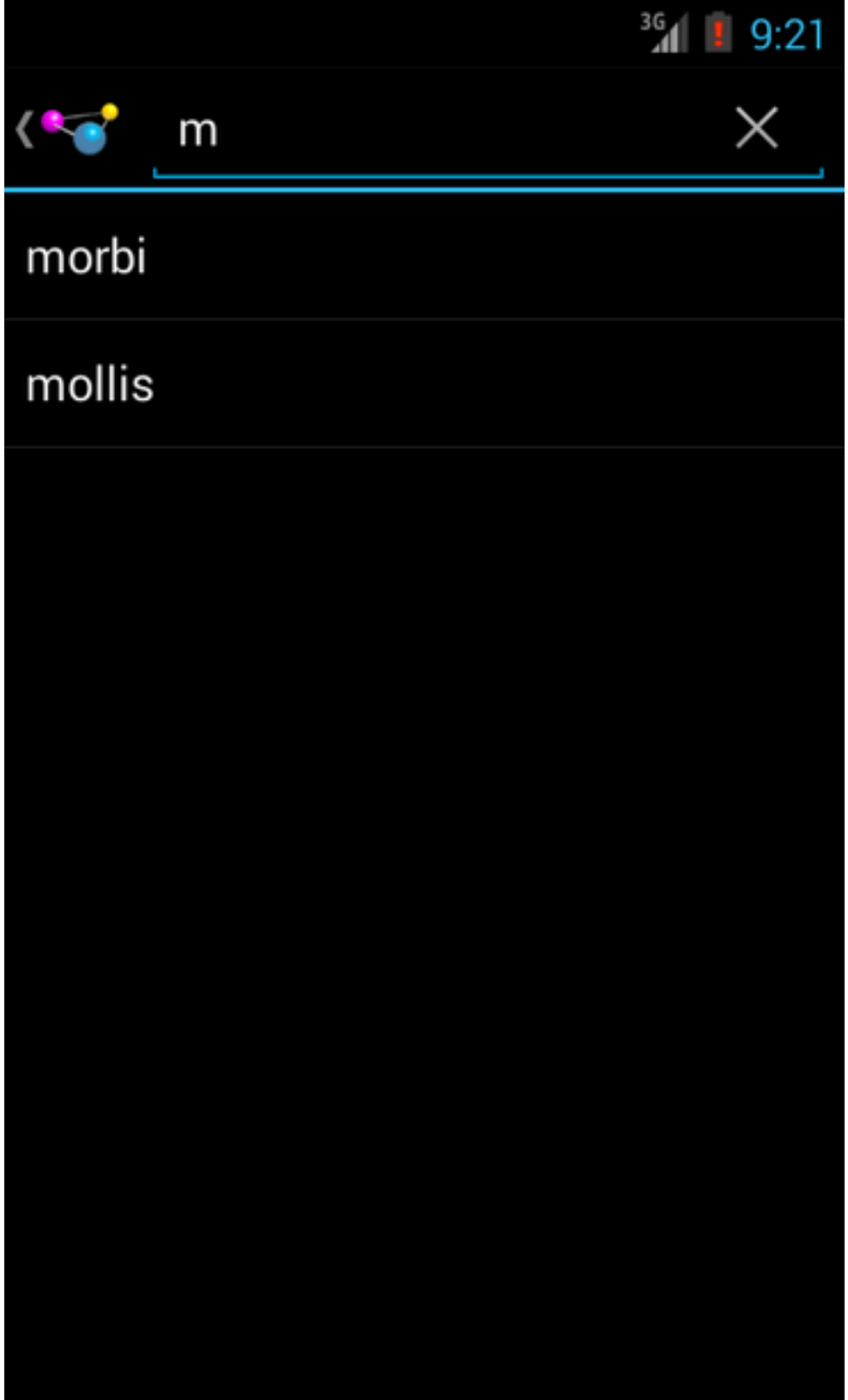
```
<item
  android:id="@+id/search"
  android:actionViewClass="android.widget.SearchView"
  android:icon="@android:drawable/ic_menu_search"
  android:showAsAction="ifRoom|collapseActionView"
  android:title="@string/filter">
</item>
```

```
public class ActionBarFragment extends ListFragment implements
    TextView.OnEditorActionListener, SearchView.OnQueryTextListener,
    SearchView.OnCloseListener {
```

```
@Override
public boolean onQueryTextChange(String newText) {
  if (TextUtils.isEmpty(newText)) {
    adapter.getFilter().filter("");
  }
  else {
    adapter.getFilter().filter(newText.toString());
  }

  return(true);
}

@Override
public boolean onQueryTextSubmit(String query) {
  return(false);
}

@Override
public boolean onClose() {
  adapter.getFilter().filter("");

  return(true);
}
```

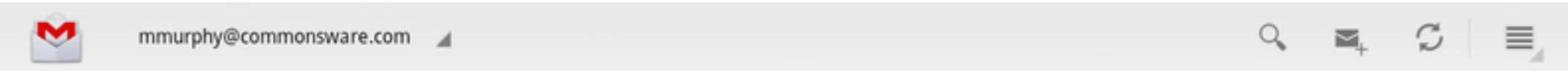
```
private void configureSearchView(Menu menu) {
  MenuItem search=menu.findItem(R.id.search);

  sv=(SearchView)search.getActionView();
  sv.setOnQueryTextListener(this);
  sv.setOnCloseListener(this);
  sv.setSubmitButtonEnabled(false);
  sv.setIconifiedByDefault(true);

  if (initialQuery != null) {
    sv.setIconified(false);
    search.expandActionView();
    sv.setQuery(initialQuery, true);
  }
}
```


SearchView Demo


9:21
m
morbi
mollis

Action Modes

- Alternate Action Bar for Contextual Actions
  - Operations on selections
    - Multiple selections in a list
    - Selected text in a TextView, EditText, WebView, etc.
  - Replacement for context menu


mmurphy@commonsware.com


Done
3 selected
Change labels

Action Modes

- ActionMode.Callback
  - Configure ActionMode in `onCreateActionMode()`
  - `onActionItemClicked()` if user clicks a toolbar button
  - `finish()` the ActionMode when done
  - Clean up in `onDestroyActionMode()`

```
@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);

  initAdapter();
  getListView().setLongClickable(true);
  getListView().setChoiceMode(ListView.CHOICE_MODE_SINGLE);
  getListView().setOnItemLongClickListener(new ActionModeHelper(
                                                    this,
                                                    getListView()));

}
```

```
public class ActionModeHelper implements ActionMode.Callback,
    AdapterView.OnItemLongClickListener {
  ActionModeDemo host;
  ActionMode activeMode;
  ListView modeView;

  ActionModeHelper(final ActionModeDemo host, ListView modeView) {
    this.host=host;
    this.modeView=modeView;
  }

  @Override
  public boolean onItemLongClick(AdapterView<?> view, View row,
                                 int position, long id) {
    modeView.clearChoices();
    modeView.setItemChecked(position, true);

    if (activeMode == null) {
      activeMode=host.startActionMode(this);
    }

    return(true);
  }
```

```
@Override
public boolean onCreateActionMode(ActionMode mode, Menu menu) {
  MenuInflater inflater=host.getMenuInflater();

  inflater.inflate(R.menu.context, menu);
  mode.setTitle(R.string.context_title);

  return(true);
}

@Override
public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
  return(false);
}
```

```
@Override
public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
  boolean result=
      host.performAction(item.getItemId(),
                         modeView.getCheckedItemPosition());

  if (item.getItemId() == R.id.remove) {
    activeMode.finish();
  }

  return(result);
}

@Override
public void onDestroyActionMode(ActionMode mode) {
  activeMode=null;
  modeView.clearChoices();
  modeView.requestLayout();
}
```

Action Modes

- Automatic Multiple-Choice Action Mode
  - `CHOICE_MODE_MULTIPLE_MODAL` and an appropriate row layout
  - Checking item toggles on action mode with your supplied MultiChoiceModeListener callback
    - Serves as ActionBar.Callback, plus `onItemCheckedStateChanged()` for check/uncheck events

Action Modes

- Long-Press-Initiated Automatic Action Mode
  - Start off in single-choice mode
  - On long-click of item, toggle into `CHOICE_MODE_MULTIPLE_MODAL`
  - When action mode destroyed, switch back to single-choice mode
  - Need to remember choice mode across configuration changes!

```
private ActionMode activeMode=null;

@Override
public void onCreate(Bundle state) {
  super.onCreate(state);

  if (state == null) {
    initAdapter(null);
  }
  else {
    initAdapter(state.getStringArrayList(STATE_MODEL));
  }

  getListView().setOnItemLongClickListener(this);
  getListView().setMultiChoiceModeListener(this);

  int choiceMode=
      (state == null ? ListView.CHOICE_MODE_NONE
          : state.getInt(STATE_CHOICE_MODE));

  getListView().setChoiceMode(choiceMode);
}
```

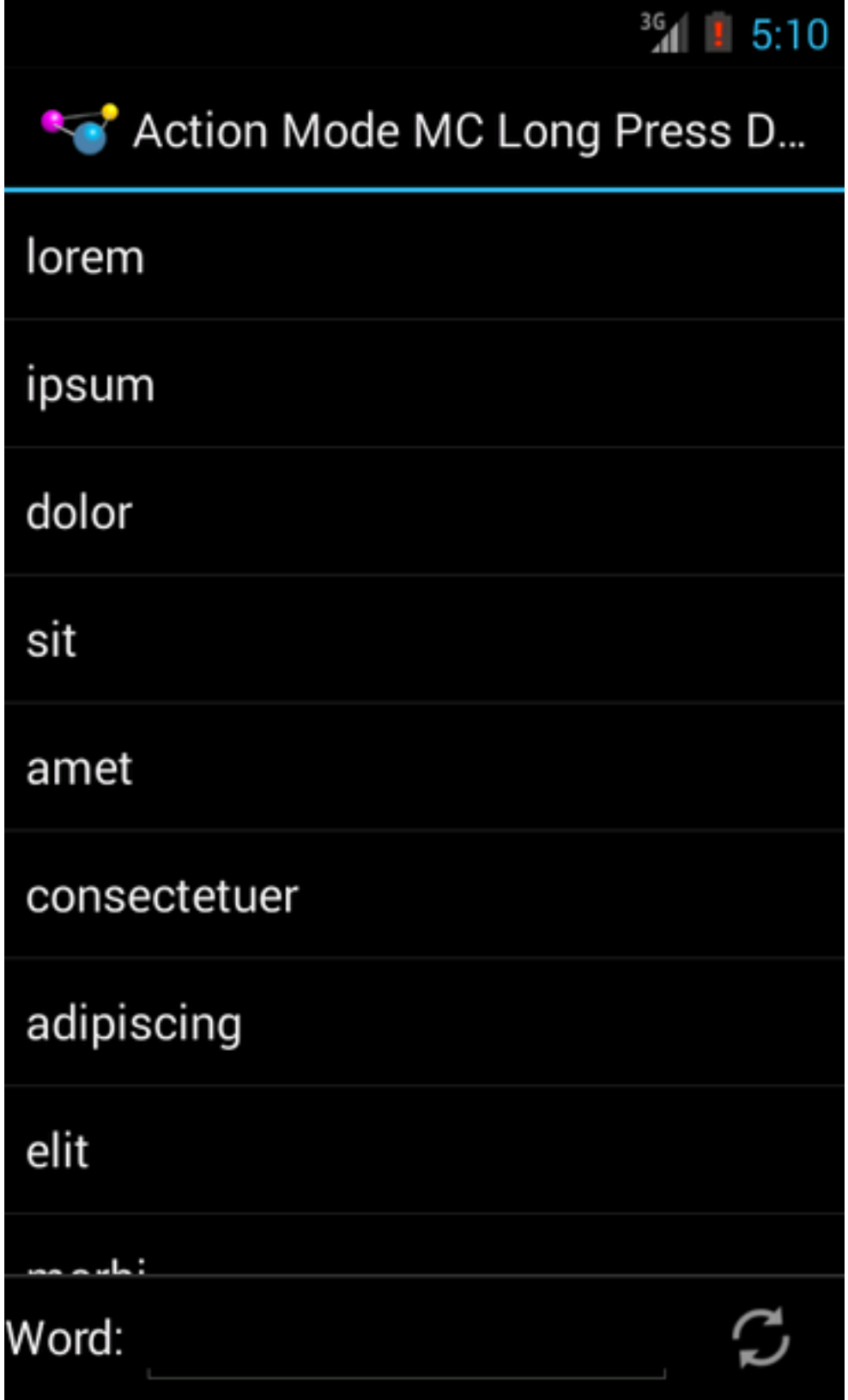
```
@Override
public void onListItemClick(ListView l, View v, int position, long
  if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
    l.setItemChecked(position, true);
  }
}
```

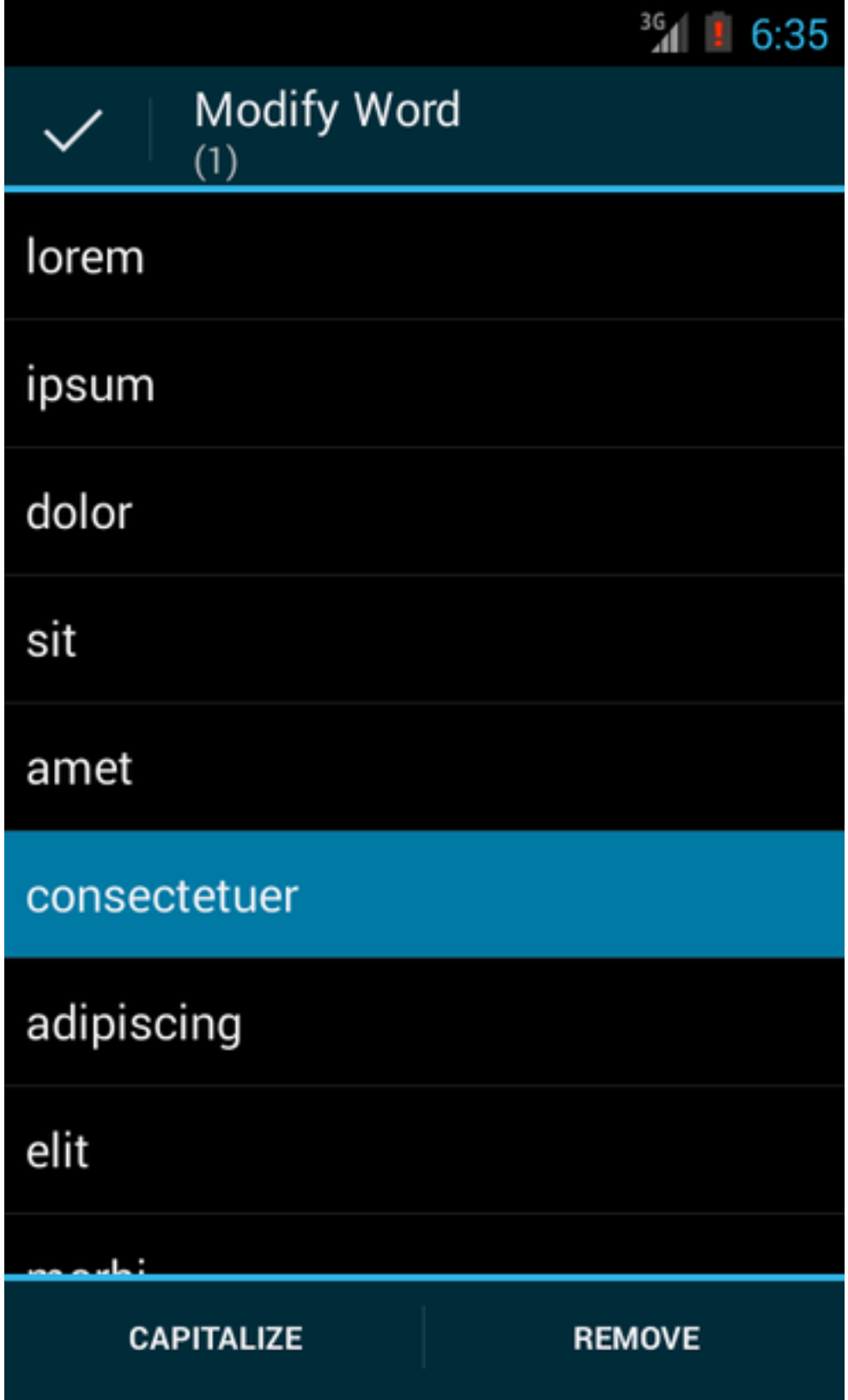
```
@Override
public boolean onItemLongClick(AdapterView<?> parent, View view,
                               int position, long id) {
  getListView().setChoiceMode(ListView.CHOICE_MODE_MULTIPLE_MODAL);
  getListView().setItemChecked(position, true);

  return(true);
}
```

```
@Override
public boolean onCreateActionMode(ActionMode mode, Menu menu) {
  MenuInflater inflater=getMenuInflater();

  inflater.inflate(R.menu.context, menu);
  mode.setTitle(R.string.context_title);
  activeMode=mode;
  updateSubtitle(activeMode);

  return(true);
}

@Override
public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
  return(false);
}
```

```
@Override
public boolean onActionItemClicked(ActionMode mode, MenuItem item)
  boolean result=performActions(item);

  updateSubtitle(activeMode);

  return(result);
}

@Override
public void onDestroyActionMode(ActionMode mode) {
  if (activeMode != null) {
    activeMode=null;
    getListView().setChoiceMode(ListView.CHOICE_MODE_NONE);
    getListView().setAdapter(getListView().getAdapter());
  }
}
```

```
@Override
public void onItemCheckedStateChanged(ActionMode mode, int position
                                      long id, boolean checked) {
  if (activeMode != null) {
    updateSubtitle(mode);
  }
}

private void updateSubtitle(ActionMode mode) {
  mode.setSubtitle("(" + getListView().getCheckedItemCount() + ")")
}
```


5:10
Action Mode MC Long Press D...
lorem
ipsum
dolor
sit
amet
consectetuer
adipiscing
elit
Word:


6:35
Modify Word
(1)
lorem
ipsum
dolor
sit
amet
consectetuer
adipiscing
elit
CAPITALIZE
REMOVE


6:35
Modify Word
(3)
lorem
ipsum
dolor
sit
amet
consectetuer
adipiscing
elit
CAPITALIZE
REMOVE

Action Bar Navigation

- Option #1: Tabs
  - Use `setNavigationMode()` on ActionBar
    - `NAVIGATION_MODE_TABS`
  - Call `addTab()` to add a tab
  - Pros: easy to set up, automatic fragment support
  - Cons
    - **DEPRECATED**
    - May appear on separate row
    - May be converted into list navigation

Action Bar Navigation

- Option #2: List
  - Use `setNavigationMode()` on ActionBar
    - `NAVIGATION_MODE_LIST`
  - Call `setListNavigationCallbacks()` to define Spinner contents and listener
  - **DEPRECATED**

Action Bar Navigation

- Option #3: `setCustomView()`
  - You supply your own View or layout resource ID
  - Used in the navigation space on the action bar, instead of tabs or Spinner
  - Example: AutoCompleteTextView for browser
  - `getCustomView()` to retrieve inflated layout for runtime configuration

What The L's Going On Here?

- Toolbar
  - Separate class, wrapping up basic action bar functionality
  - Designed to be placed wherever it is needed
- ActionBar
  - Wrapper around Toolbar for classic top-of-the-activity positioning
  - Provides basic support for deprecated features

L Is So Materialistic

- Material Design: Custom Action Bar Colors
  - Themed
  - Palette-Driven
    - Adapt the theme based upon the colors detected in a photo that the activity will be showing

What Else Is There?

- Custom ActionProviders and ActionViews
- ActionBarDrawerToggle
- Transparent/Translucent Action Bars
- Full-Screen/Immersive Modes
- Checkable Action Items
- Long-Press “Tooltip” Help
- And more!