Welcome, नमस्कार

AITI IIT Bombay Class 2013

Lecture 9

Agenda

- Design Documents
- Revenue:
  - Advertising
  - In App Purchases
  - Google Play

Tomorrow

- Location Services
- Multimedia
- Graphics

Video

- http://www.youtube.com/watch?v=O8i4HUw7JYA

Design Documents

- 3 Power point slides
  - 1 with functional requirements
  - 1 with labeled gui activities, buttons and processes
  - 1 with labeled information flow

Design Document Examples: India Votes (Requirements)

Description: India votes is an app that allows users to vote on various current pop culture topics for fun

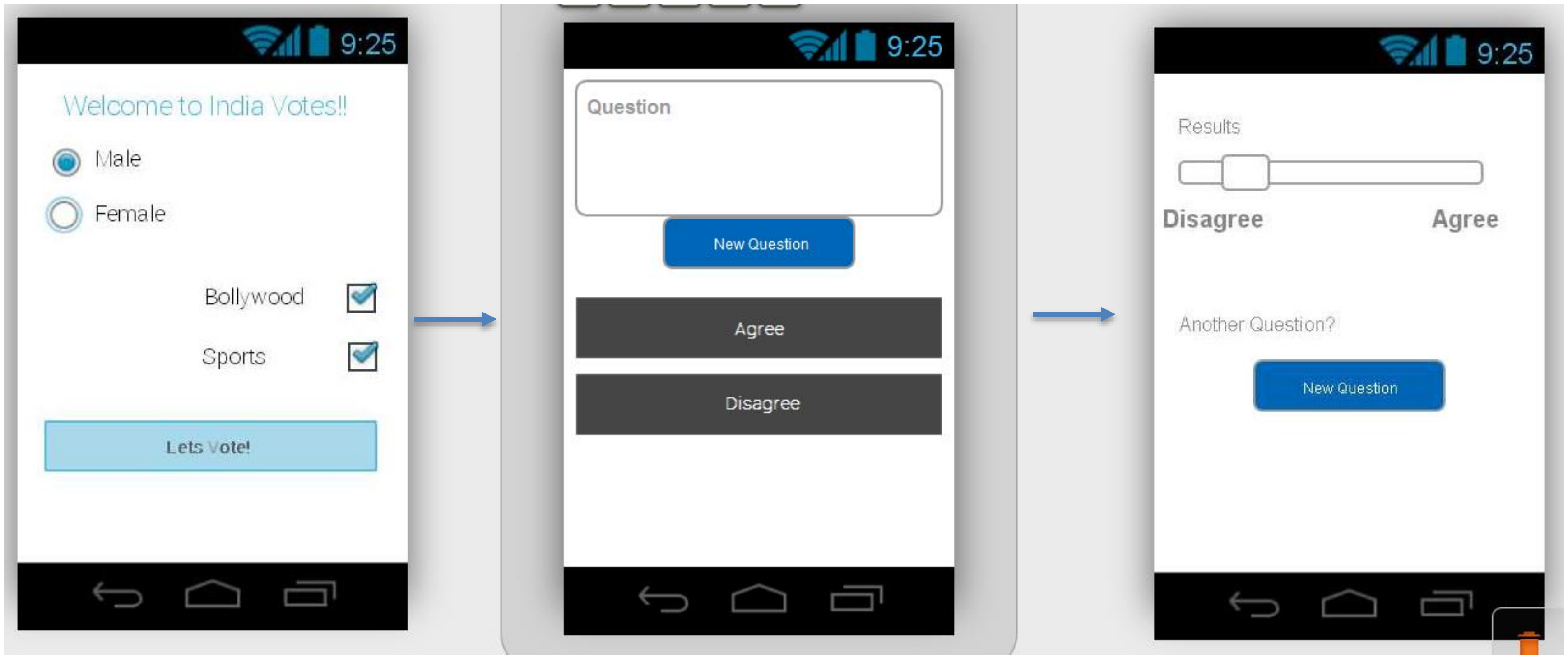
- The user must be able to login and set preferences for questions, and age group
- The app must generate questions based on the user's preferences and send the voting results to a central server
- The app must display the current voting results to the user and allow the user to respond to further questions if he/she chooses to do so

Design Document Examples: India Votes (GUI)

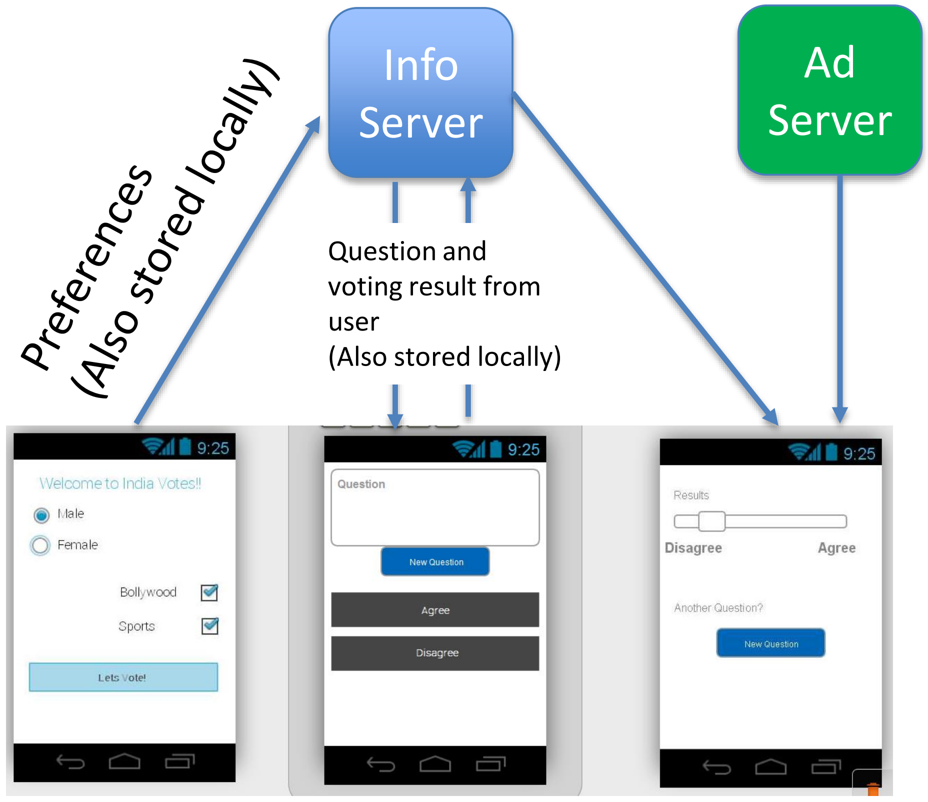
Enter Preferences

Vote on a question

See results from other voters



Design Document Examples: India Votes (Information Flow)

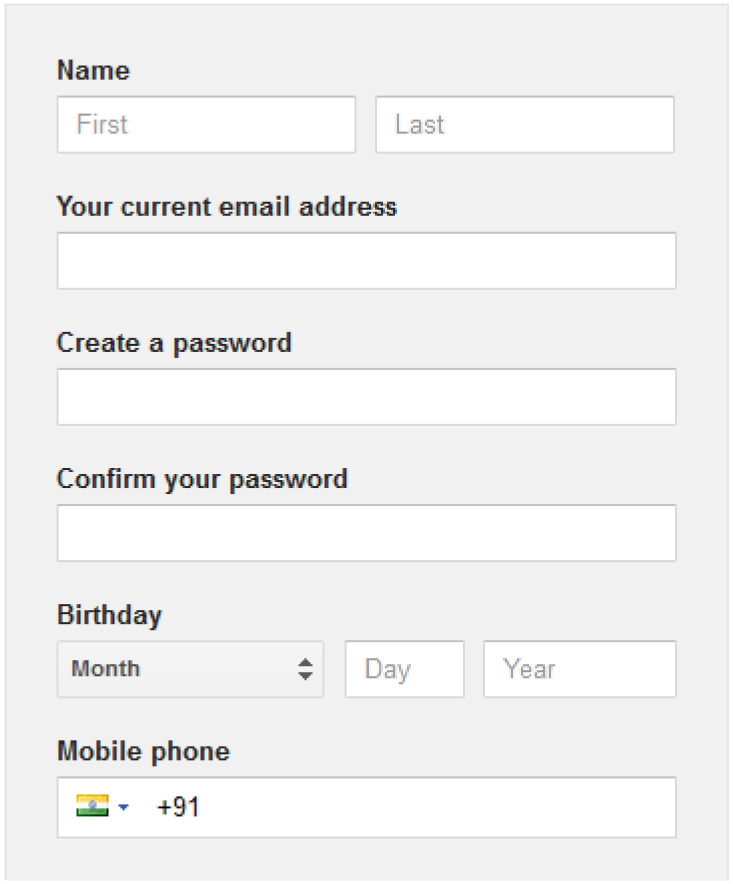


Android Revenue ☺

- What are the main technical tasks required to set up:
  - Advertising
  - In-App purchases
  - Purchases from google play

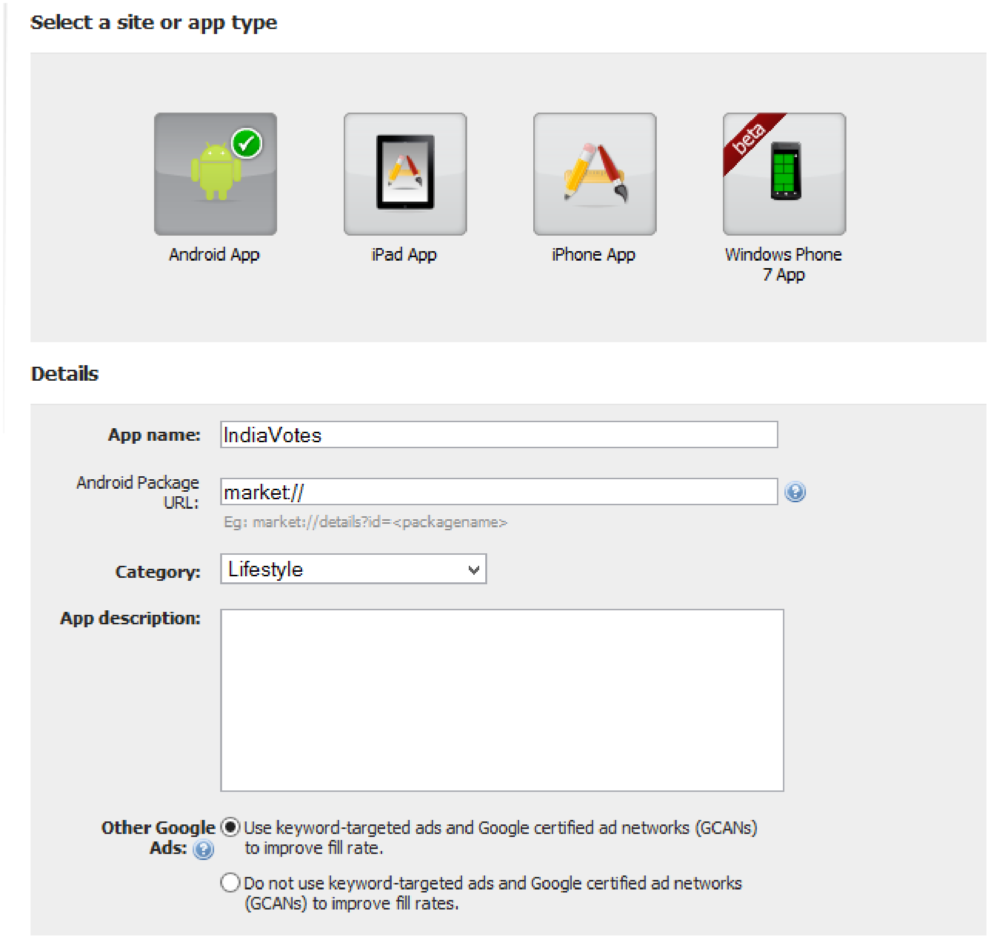
Advertising

- Sign up with Admob
- You will need an Indian bank account, Indian mobile phone number



Name	First / Last
Your current email address	
Create a password	
Confirm your password	
Birthday	Month / Day / Year
Mobile phone	+91

Admob (Developers)

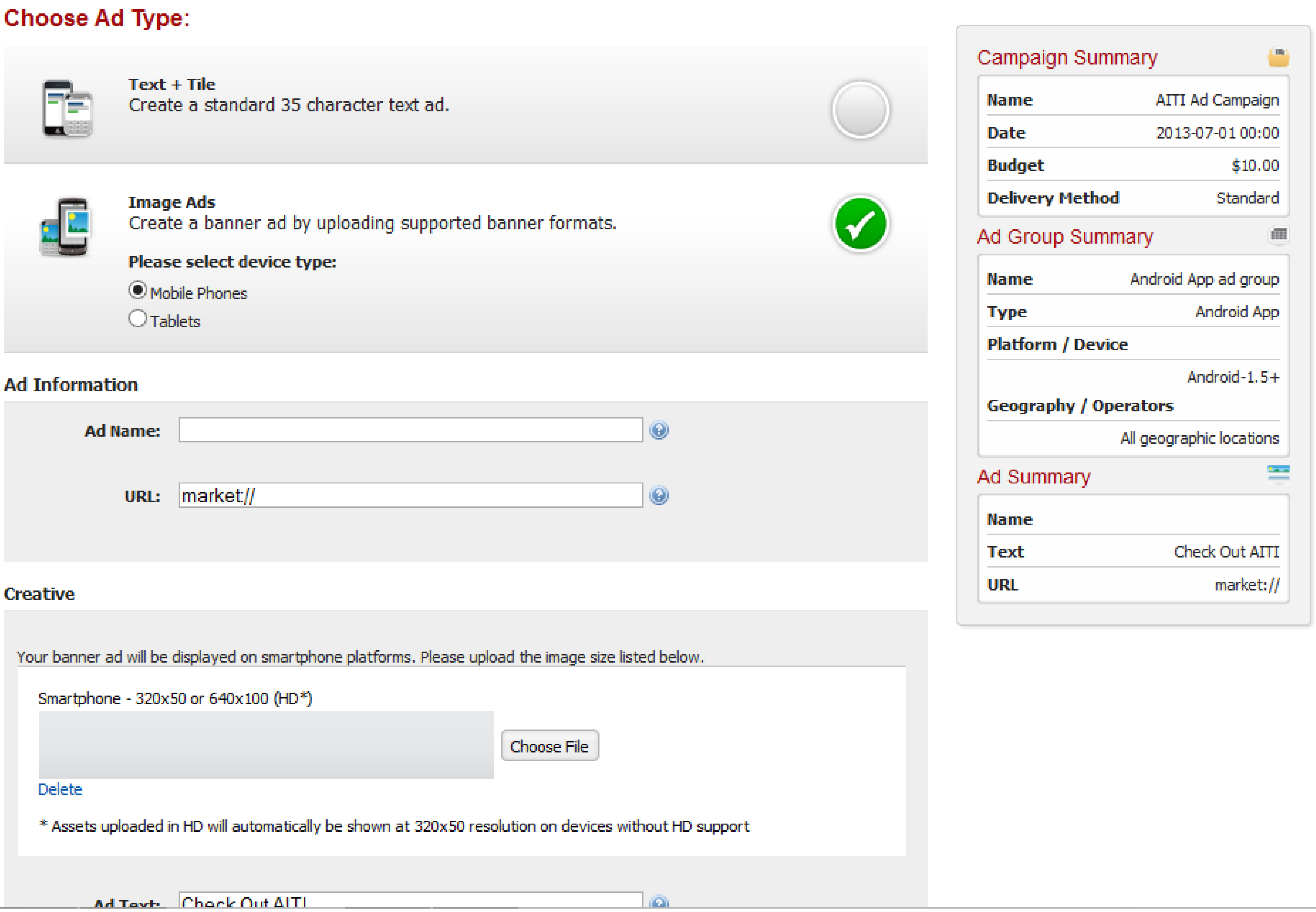


Select a site or app type

Details

App name:	IndiaVotes
Android Package URL:	market:// Eg: market://details?id=<packagename>
Category:	Lifestyle
App description:	
Other Google Ads:	[✓ Use keyword-targeted ads and Google certified ad networks (GCANs) to improve fill rate.] Do not use keyword-targeted ads and Google certified ad networks (GCANs) to improve fill rates.

Admob (Advertisers)



Choose Ad Type:

Text + Tile
Create a standard 35 character text ad.

Image Ads ✓
Create a banner ad by uploading supported banner formats.

Please select device type:
[✓ Mobile Phones]
Tablets

Ad Information

Ad Name:	
URL:	market://

Creative

Your banner ad will be displayed on smartphone platforms. Please upload the image size listed below.

Smartphone - 320x50 or 640x100 (HD*)

Choose File

Delete

* Assets uploaded in HD will automatically be shown at 320x50 resolution on devices without HD support

Ad Text: Check Out AITI

Campaign Summary

Name	AITI Ad Campaign
Date	2013-07-01 00:00
Budget	$10.00
Delivery Method	Standard

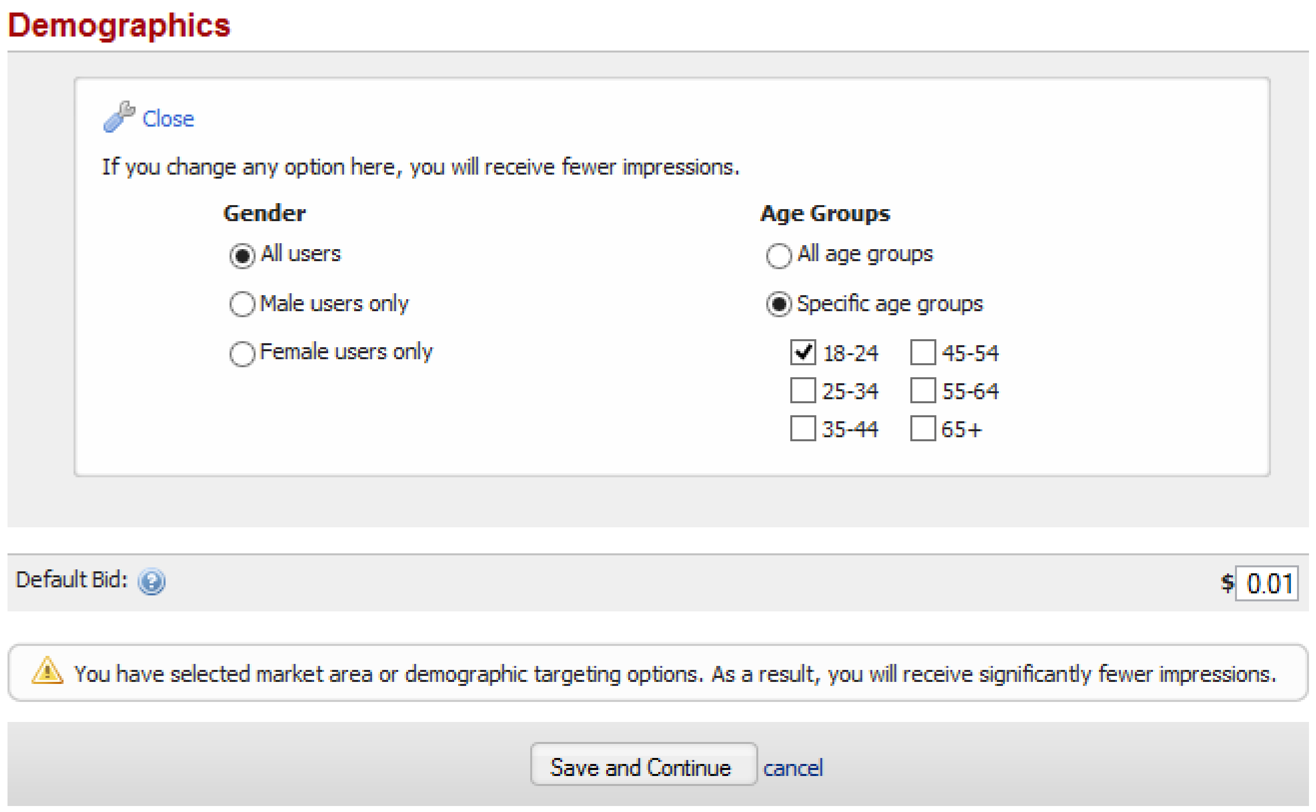
Ad Group Summary

Name	Android App ad group
Type	Android App
Platform / Device	Android-1.5+
Geography / Operators	All geographic locations

Ad Summary

Name	
Text	Check Out AITI
URL	market://

Admob (Advertisers)



Demographics

Close

If you change any option here, you will receive fewer impressions.

Gender

- [✓ All users]
- Male users only
- Female users only

Age Groups

- All age groups
- [✓ Specific age groups]

[✓ 18-24]	45-54
25-34	55-64
35-44	65+

Default Bid: $ 0.01

You have selected market area or demographic targeting options. As a result, you will receive significantly fewer impressions.

Admob

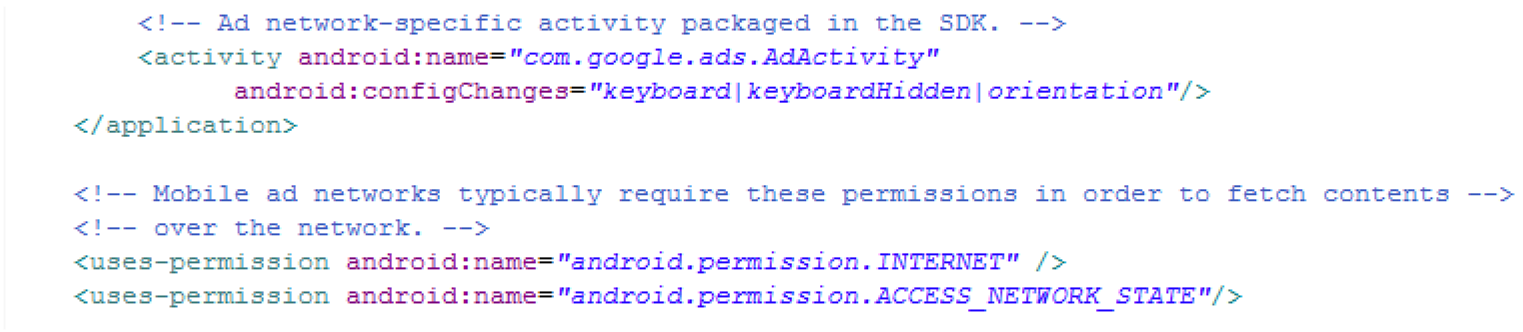
- Include key words to aid in targeting in your ad description
- Easy to implement but be careful not to be too intrusive to the user.

Implementing ads

- Manifest
- Activity
- Listeners
- Formatting

Implement Ads: Android Manifest

- Set permissions: you can get fine or coarse locations

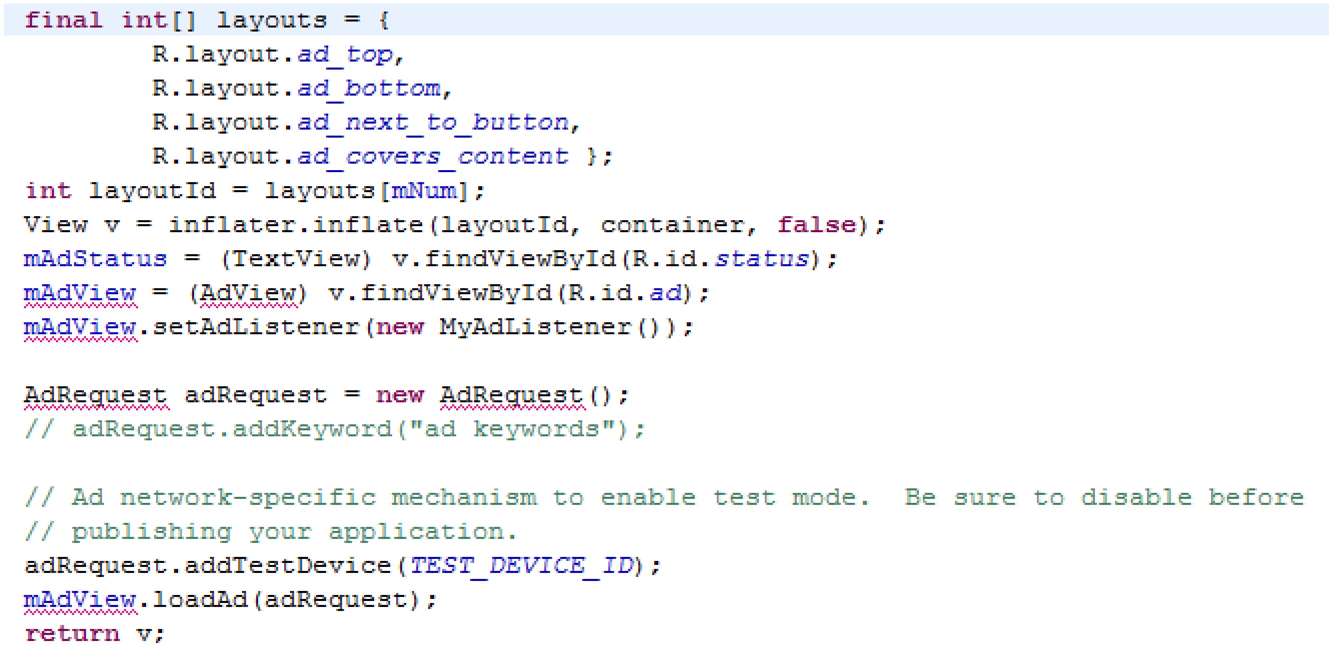


```
    <!-- Ad network-specific activity packaged in the SDK. -->
    <activity android:name="com.google.ads.AdActivity"
          android:configChanges="keyboard|keyboardHidden|orientation"/>
</application>

<!-- Mobile ad networks typically require these permissions in order to fetch contents -->
<!-- over the network. -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

Implementing Ads: Activity (onCreateView)

- Initialize your AdView Object
- Make a add request
- Load the request into your add view



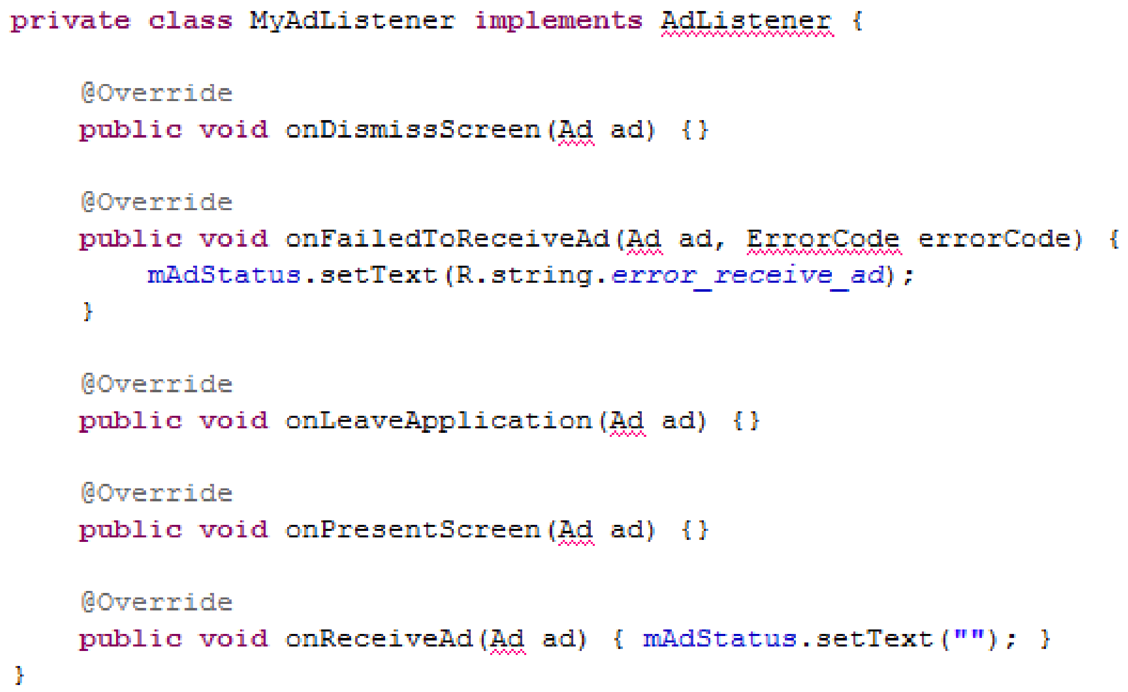
```
final int[] layouts = {
        R.layout.ad_top,
        R.layout.ad_bottom,
        R.layout.ad_next_to_button,
        R.layout.ad_covers_content };
int layoutId = layouts[mNum];
View v = inflater.inflate(layoutId, container, false);
mAdStatus = (TextView) v.findViewById(R.id.status);
mAdView = (AdView) v.findViewById(R.id.ad);
mAdView.setAdListener(new MyAdListener());

AdRequest adRequest = new AdRequest();
// adRequest.addKeyword("ad keywords");

// Ad network-specific mechanism to enable test mode.  Be sure to disable before
// publishing your application.
adRequest.addTestDevice(TEST_DEVICE_ID);
mAdView.loadAd(adRequest);
return v;
```

Implementing Ads: Activity (listeners)

- Add a listener to determine behavior for different ad events:



```
private class MyAdListener implements AdListener {

    @Override
    public void onDismissScreen(Ad ad) {}

    @Override
    public void onFailedToReceiveAd(Ad ad, ErrorCode errorCode) {
        mAdStatus.setText(R.string.error_receive_ad);
    }

    @Override
    public void onLeaveApplication(Ad ad) {}

    @Override
    public void onPresentScreen(Ad ad) {}

    @Override
    public void onReceiveAd(Ad ad) { mAdStatus.setText(""); }
}
```

Admob manual

- http://mm.admob.com/web/pdf/AdMob_API_Documentation.pdf

In App Purchasing (setup)

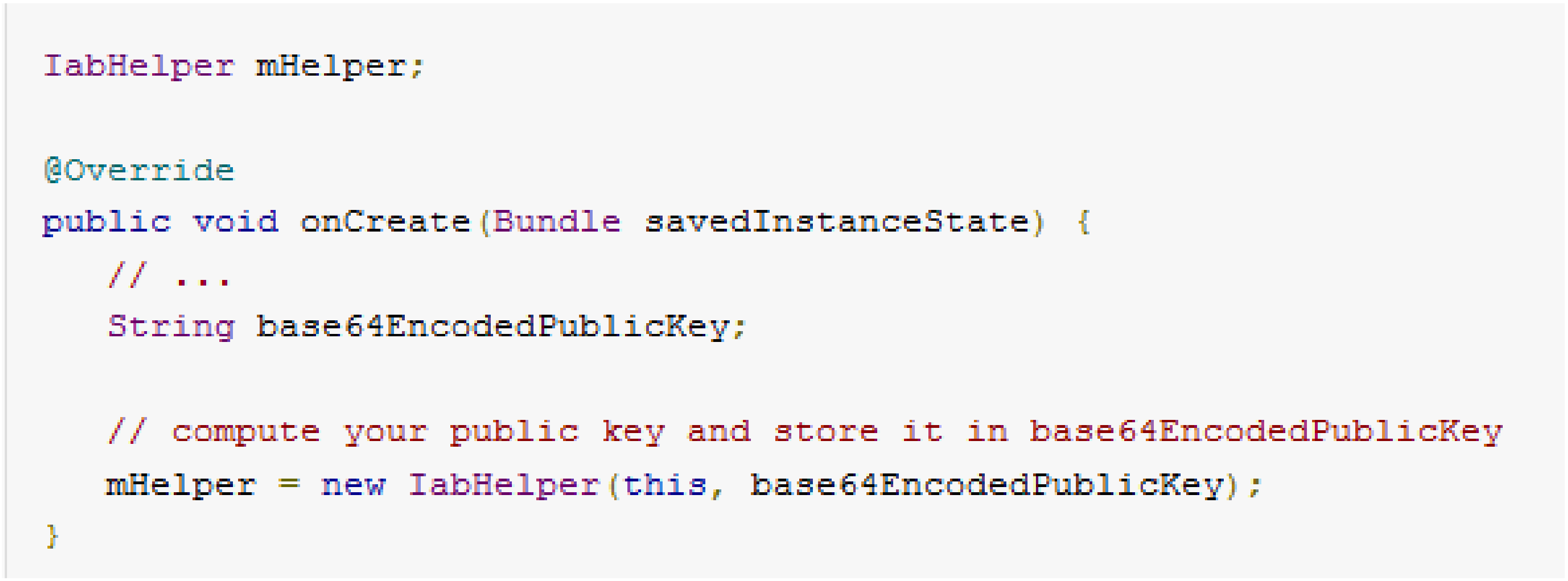
- Setup an account with Google Play Developer Console, get public key
- Create a Google Wallet Merchant Account
- Add billing libraries and permissions to your app (in app billing library to src directory)

```
<uses-permission android:name="com.android.vending.BILLING" />
```

- Bind your app to google play

In App Purchasing (setup)

- Binding your app to google play (the IabHelper object will be your main tool for billing)



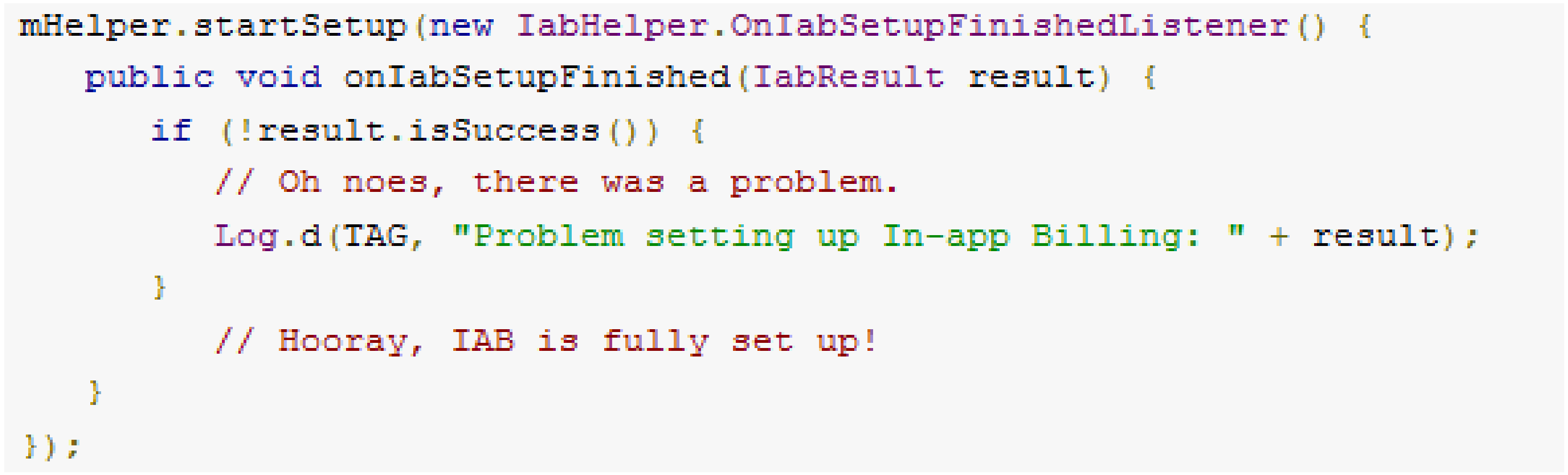
```
IabHelper mHelper;

@Override
public void onCreate(Bundle savedInstanceState) {
   // ...
   String base64EncodedPublicKey;

   // compute your public key and store it in base64EncodedPublicKey
   mHelper = new IabHelper(this, base64EncodedPublicKey);
}
```

In App Purchasing (setup)

- Complete the bind by calling startSetup()



```
mHelper.startSetup(new IabHelper.OnIabSetupFinishedListener() {
   public void onIabSetupFinished(IabResult result) {
      if (!result.isSuccess()) {
         // Oh noes, there was a problem.
         Log.d(TAG, "Problem setting up In-app Billing: " + result);
      }
         // Hooray, IAB is fully set up!
   }
});
```

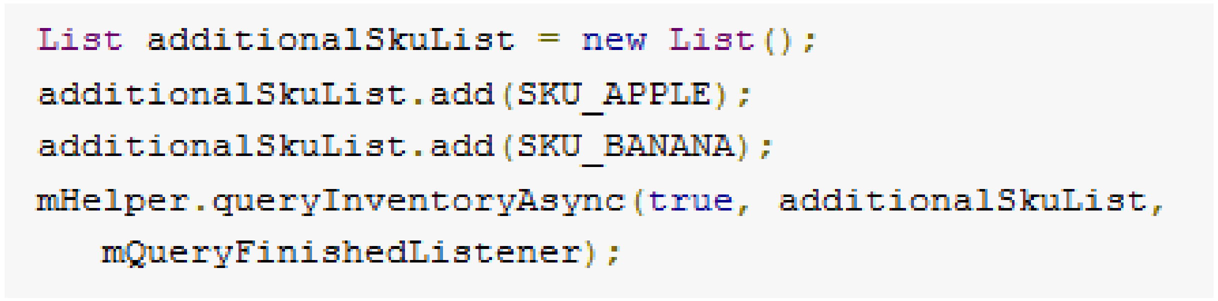
Implementing In-App Purchasing

- Creating Goods (Use developer console)
- Querying Goods
- Purchasing Goods

Implementing In-App Purchasing: Querying Goods

- Querying Goods: use queryInventorAsync() method:

  queryInventoryAsync(boolean, List, QueryInventoryFinishedListener)



```
List additionalSkuList = new List();
additionalSkuList.add(SKU_APPLE);
additionalSkuList.add(SKU_BANANA);
mHelper.queryInventoryAsync(true, additionalSkuList,
   mQueryFinishedListener);
```

Implementing In-App Purchasing: Querying Goods

- The query will come back in an inventory object:

```
IabHelper.QueryInventoryFinishedListener
   mQueryFinishedListener = new IabHelper.QueryInventoryFinishedListener() {
   public void onQueryInventoryFinished(IabResult result, Inventory inventory)
   {
      if (result.isFailure()) {
         // handle error
         return;
       }

       String applePrice =
          inventory.getSkuDetails(SKU_APPLE).getPrice();
       String bananaPrice =
          inventory.getSkuDetails(SKU_BANANA).getPrice();

       // update the UI
   }
```

MIT AITI

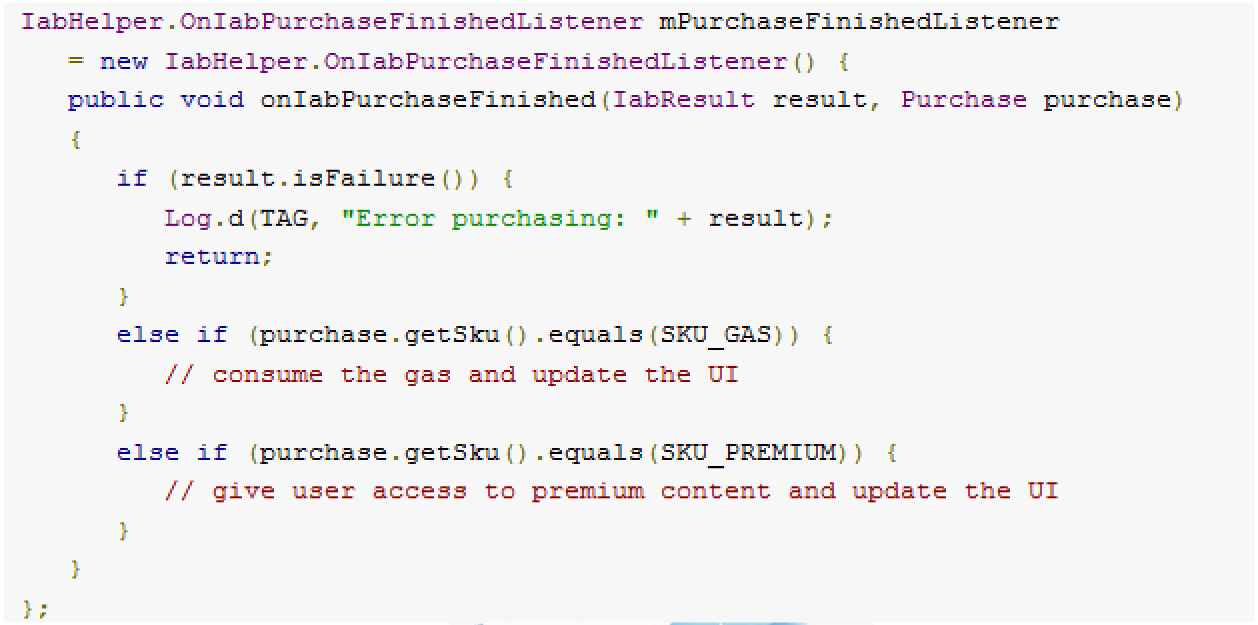
Implementing In-App Purchasing: Purchasing Goods

- Once you have your SKU id, use the launchPurchaseFlow() method…

```
mHelper.launchPurchaseFlow(this, SKU_GAS, 10001,
   mPurchaseFinishedListener, "bGoa+V7g/yqDXvKRqq+JTFn4uQZbPiQJo4pf9RzJ");
```

Implementing In-App Purchasing: Purchasing Goods

- Use the listener to process the response
- Goods that can be purchased more than once need to be “consumed”, premium goods can only be purchased once.



```
IabHelper.OnIabPurchaseFinishedListener mPurchaseFinishedListener
   = new IabHelper.OnIabPurchaseFinishedListener() {
   public void onIabPurchaseFinished(IabResult result, Purchase purchase)
   {
      if (result.isFailure()) {
         Log.d(TAG, "Error purchasing: " + result);
         return;
      }
      else if (purchase.getSku().equals(SKU_GAS)) {
         // consume the gas and update the UI
      }
      else if (purchase.getSku().equals(SKU_PREMIUM)) {
         // give user access to premium content and update the UI
      }
   }
};
```

Implementing In-App Purchasing: References

- Android Tutorial: http://developer.android.com/training/in-app-billing/index.html
- Sample App (within the tutorial)

A Few words about revenue

- Try the test apps first for ads and in-app purchasing
- Wait until you are ready to publish your app to implement ads and in-app purchasing (they require bank accounts and google play accounts)
- Feel free to ad place holders for ads and in-app purchases if you are not quite ready by demo day