Mobile Human-Computer Interaction

Stephen Brewster, Euan Freeman (UG)
Harry Nguyen (UGS)

GLASGOW INTERACTIVE SYSTEMS GROUP

University of Glasgow

Stephen.Brewster@glasgow.ac.uk
Harry.Nguyen@glasgow.ac.uk
moodle2.gla.ac.uk/course/view.php?id=969

GOOGLE FIREBASE

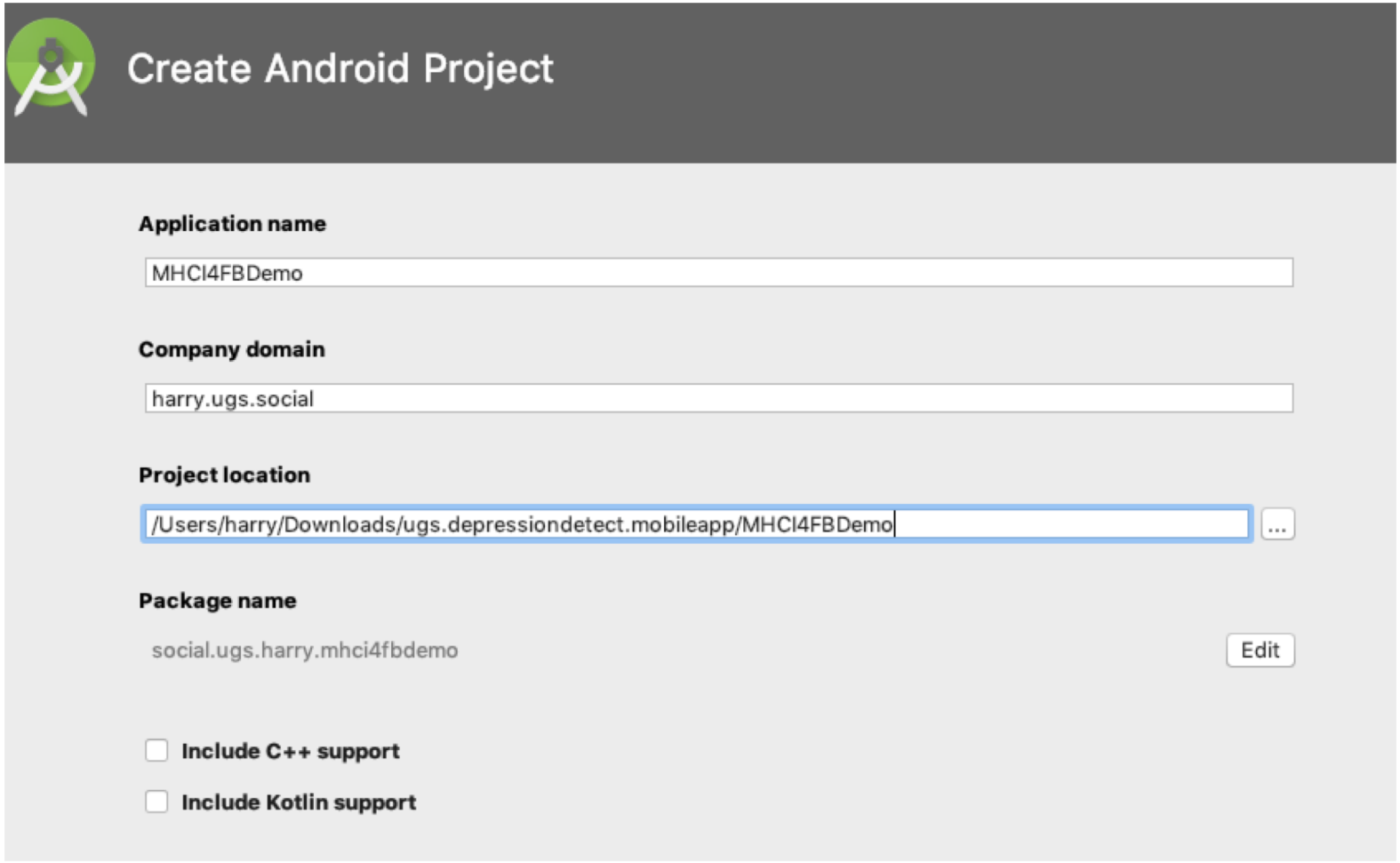
Google Firebase

- A comprehensive mobile development platform
  - Analytics
  - Cloud Messaging
  - Realtime Database
  - Authentication
  - ML Kit

Learning Google Firebase

- Get started for Android - https://firebase.google.com/docs/android/setup
- Realtime Database - https://firebase.google.com/docs/database/
- Machine Learning - https://firebase.google.com/docs/ml-kit/
- Photo Basics - https://developer.android.com/training/camera/photobasics

Firebase – MHCI4 Lab



Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

- [x] **Phone and Tablet**

  API 15: Android 4.0.3 (IceCreamSandwich)

  By targeting **API 15 and later**, your app will run on approximately **100%** of devices. Help me choose

  - [ ] Include Android Instant App support

- [ ] **Wear OS**

  API 23: Android 6.0 (Marshmallow)

- [ ] **TV**

  API 21: Android 5.0 (Lollipop)

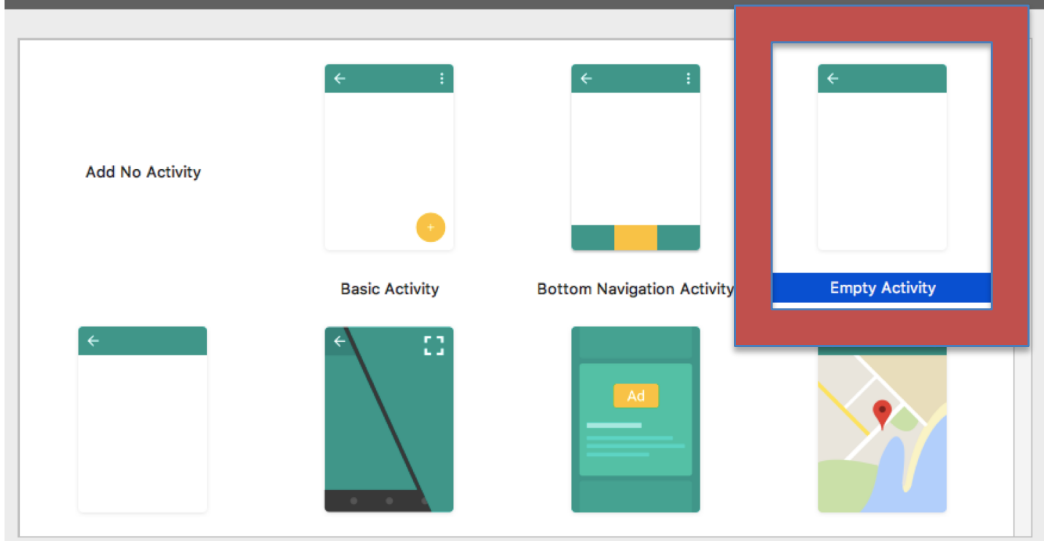
- [ ] **Android Auto**

- [ ] **Android Things**

  API 24: Android 7.0 (Nougat)

Cancel | Previous | Next | Finish

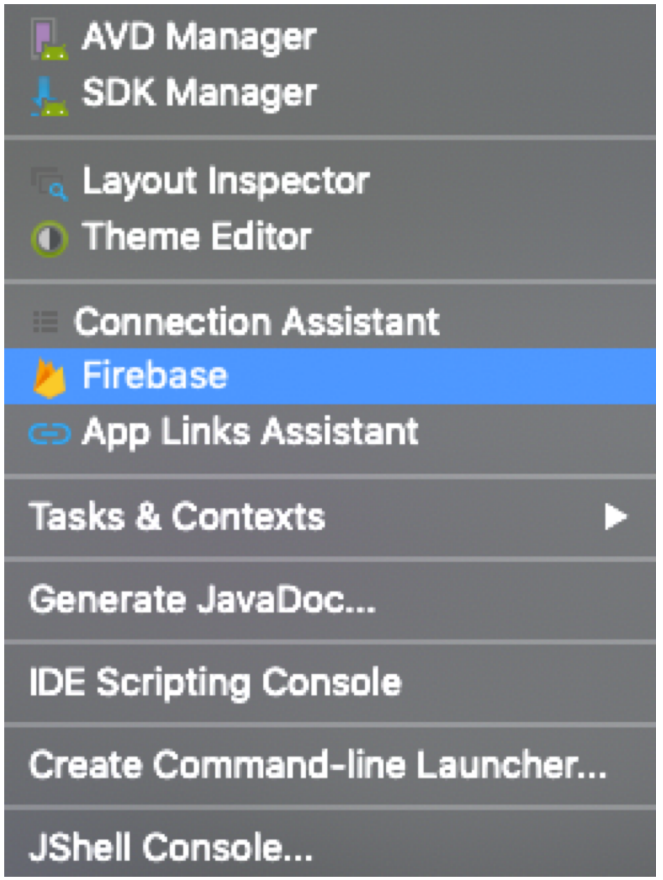
Add an Activity to Mobile



Cancel | Previous | Next | Finish

Launch Firebase Assistant

- Menu
  - Tools
  - Firebase



Firebase

Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. Learn more

- **Analytics**
  Measure user activity and engagement with free, easy, and unlimited analytics. More info
- **Cloud Messaging**
  Deliver and receive messages and notifications reliably across cloud and device. More info
- **Authentication**
  Sign in and manage users with ease, accepting emails, Google Sign-In, Facebook and other login providers. More info
- **Realtime Database**
  Store and sync data in realtime across all connected clients. More info
  - Save and retrieve data
- **Storage**
  Store and retrieve large files like images, audio, and video without writing server-side code. More info
- **Remote Config**
  Customize and experiment with app behavior using cloud-based configuration parameters. More info
- **Test Lab**
  Test your apps against a wide range of physical devices hosted in Google's cloud. More info
- **Crash Reporting**
  Get actionable insights and reports on app crashes, ANRs or other errors. More info
- **App Indexing**
  Get your app content into Google Search. More info
- **Dynamic Links**
  Create web URLs that can be shared to drive app installs and deep-linked into relevant content of your app. More info
- **Invites**
  Let your existing users easily share your app, or their favorite in-app content, via email or SMS. More info

Save and retrieve data

Our cloud database stays synced to all connected clients in realtime and remains available when your app goes offline. Data is stored in a JSON tree structure rather than a table, eliminating the need for complex SQL queries.

Launch in browser

1 Connect your app to Firebase

Connect to Firebase

2 Add the Realtime Database to your app

Add the Realtime Database to your app

3 Configure Firebase Database Rules

The Realtime Database provides a declarative rules language that allows you to define how your data should be structured, how it should be indexed, and when your data can be read from and written to. By default, read and write access to your database is restricted so only authenticated users can read or write data. To get started without setting up Authentication , you can configure your rules for public access . This does make your database open to anyone, even people not using your app, so be sure to restrict your database again when you set up authentication.

4 Write to your database

Retrieve an instance of your database using getInstance() and reference the location you want to write to.

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

You can save a range of data types to the database this way, including Java objects. When you save an object the responses from any getters will be saved as children of this location.

Firebase Setup (Google Services)

- Once connected
  - app/google-services.json should appear
- build.gradle

```
dependencies {
   classpath 'com.android.tools.build:gradle:3.2.1'
   classpath 'com.google.gms:google-services:4.2.0' // google-services plugin
...
}
```

- app/build.gradle (at the bottom)

```
...
apply plugin: 'com.google.gms.google-services'
```

4 Write to your database

Retrieve an instance of your database using getInstance() and reference the location you want to write to.

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

You can save a range of data types to the database this way, including Java objects. When you save an object the responses from any getters will be saved as children of this location.

5 Read from your database

To make your app data update in realtime, you should add a ValueEventListener to the reference you just created.

The onDataChange() method in this class is triggered once when the listener is attached and again every time the data changes, including the children.

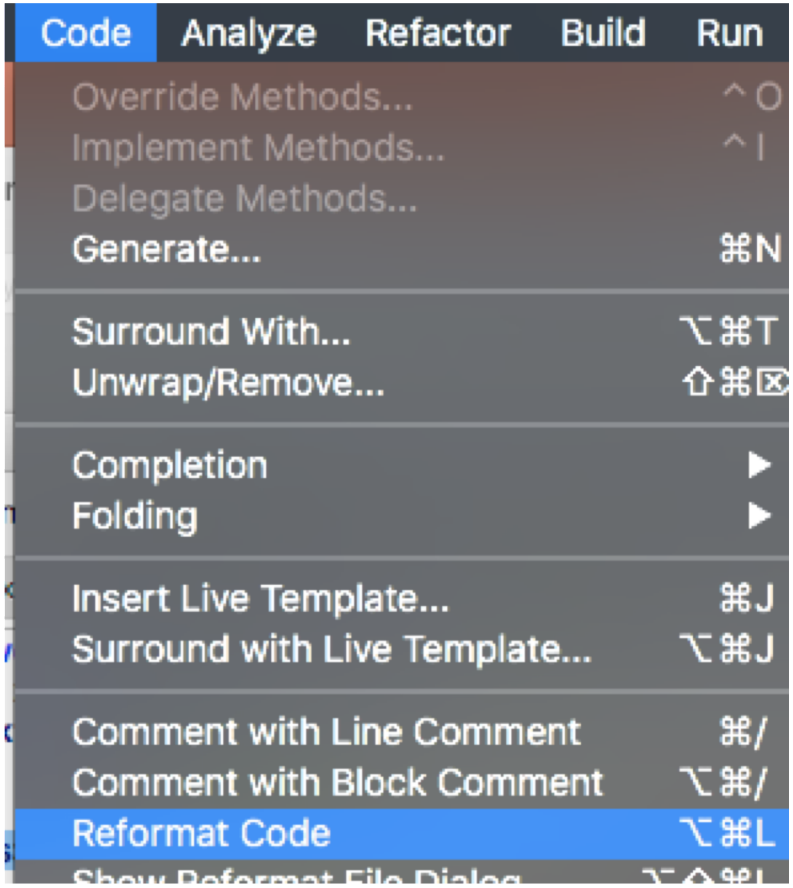
```
// Read from the database
myRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // This method is called once with the initial value and again
        // whenever data at this location is updated.
        String value = dataSnapshot.getValue(String.class);
        Log.d(TAG, "Value is: " + value);
    }

    @Override
    public void onCancelled(DatabaseError error) {
        // Failed to read value
        Log.w(TAG, "Failed to read value.", error.toException());
    }
});
```

Vision API

Feature	On-device	Cloud
Text recognition	✓	✓
Face detection	✓	
Barcode scanning	✓	
Image labeling	✓	✓
Landmark recognition		✓
Language identification	✓	
Custom model inference	✓	

Firebase Vision API

HANDS-ON



app/build.gradle

```
android {
...
    packagingOptions {
        exclude 'META-INF/proguard/androidx-annotations.pro'
        exclude 'META-INF/DEPENDENCIES'
    }
...
}
...
dependencies {
...
    implementation 'com.google.firebase:firebase-core:16.0.7'
    implementation "com.google.firebase:firebase-measurement-connector-impl:17.0.4"
    implementation 'com.google.firebase:firebase-ml-vision:18.0.2'
    implementation 'com.google.firebase:firebase-ml-vision-image-label-model:17.0.2'
...
}
...
apply plugin: 'com.google.gms.google-services'
```

app/src/main/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="social.ugs.harry.mhci4_fb_demo">
   <uses-feature android:name="android.hardware.camera" android:required="true" />
   <uses-permission android:name="android.permission.CAMERA" />
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

   <application ...>
   ...
      <provider
         android:name="android.support.v4.content.FileProvider"
         android:authorities="social.uog.mhci.fileprovider"
         android:exported="false"
         android:grantUriPermissions="true">
         <meta-data
            android:name="android.support.FILE_PROVIDER_PATHS"
            android:resource="@xml/file_paths"></meta-data>
      </provider>
    ...
   </application>
</manifest>
```

XML

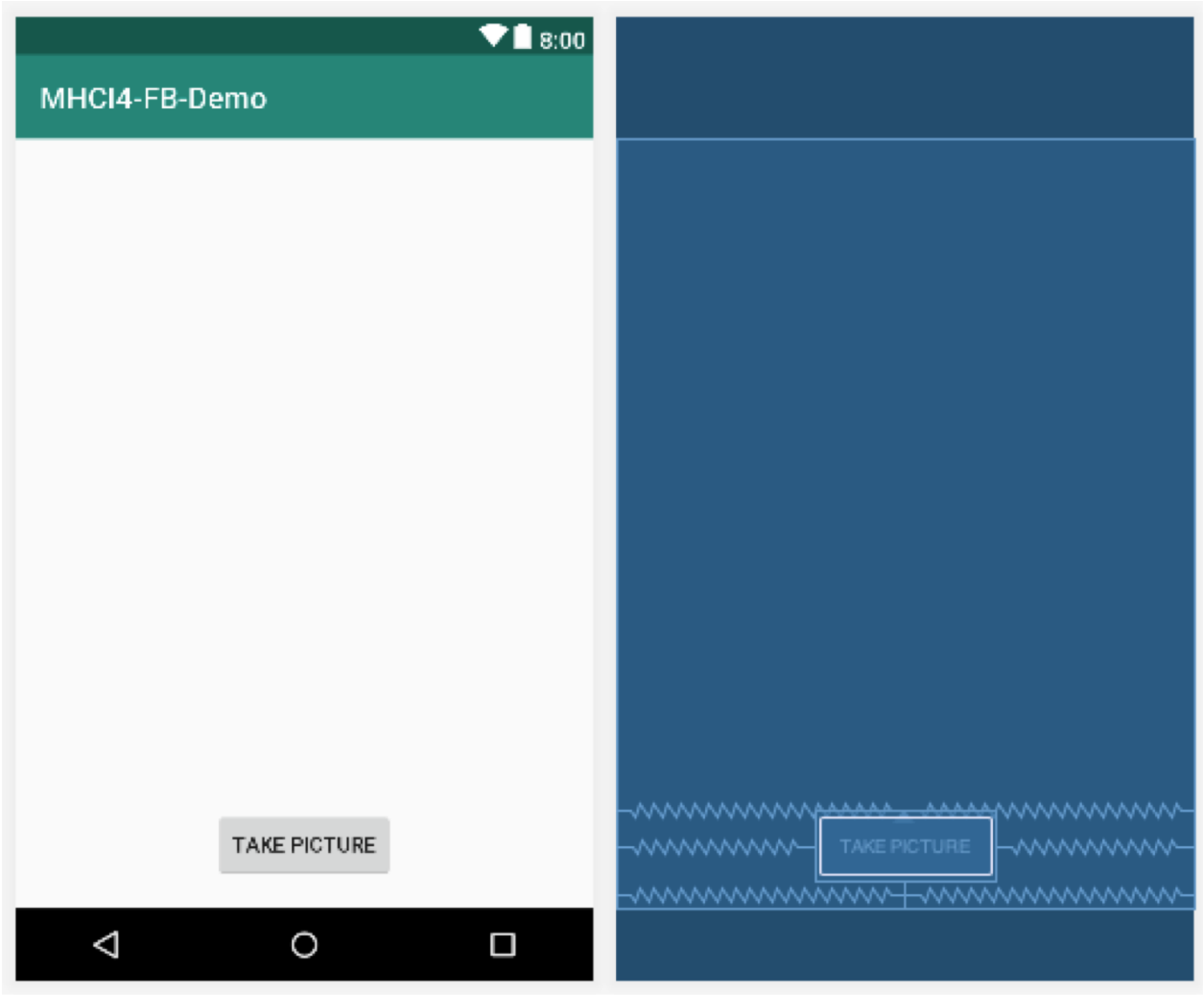
- Create app/src/res/xml/file_paths.xml

```
<?xml version="1.0" encoding="utf-8"?>
<paths xmlns:android="http://schemas.android.com/apk/res/android">
   <external-path name="my_images" path="Android/data" />
</paths>
```

app/src/main/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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">

    <ImageView
        android:id="@+id/image_preview"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_above="@+id/button_take_picture" android:layout_centerHorizontal="true" />

    <Button
        android:id="@+id/button_take_picture"
        android:layout_width="wrap_content" android:layout_height="48dp"
        android:layout_above="@+id/text_info" android:layout_centerHorizontal="true"
        android:onClick="takePicture"
        android:text="Take picture" />

    <TextView
        android:id="@+id/text_info"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_alignParentBottom="true" android:layout_centerHorizontal="true"
        android:layout_marginEnd="152dp" android:layout_marginRight="152dp"
        android:text="" />

</RelativeLayout>
```

app/src/main/res/layout/activity_main.xml



MainActivity.java (1)

```
private String currentPhotoPath; // Class variable

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Button takePictureButton = (Button) findViewById(R.id.button_take_picture);

    if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED) {
        takePictureButton.setEnabled(false);
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA,
Manifest.permission.WRITE_EXTERNAL_STORAGE}, 0);
    }
}

...
```

MainActivity.java (2)

```
@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
   if (requestCode == 0) {
      if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED &&
grantResults[1] == PackageManager.PERMISSION_GRANTED) {
         ((Button) findViewById(R.id.button_take_picture)).setEnabled(true);
      }
   }
}


private File createMediaFile() throws java.io.IOException {
   File storageDir = getExternalFilesDir(Environment.DIRECTORY_PICTURES);
   File image = File.createTempFile(String.valueOf(new Date().getTime()), ".jpg", storageDir);
   currentPhotoPath = image.getAbsolutePath();
   return image;
}
```

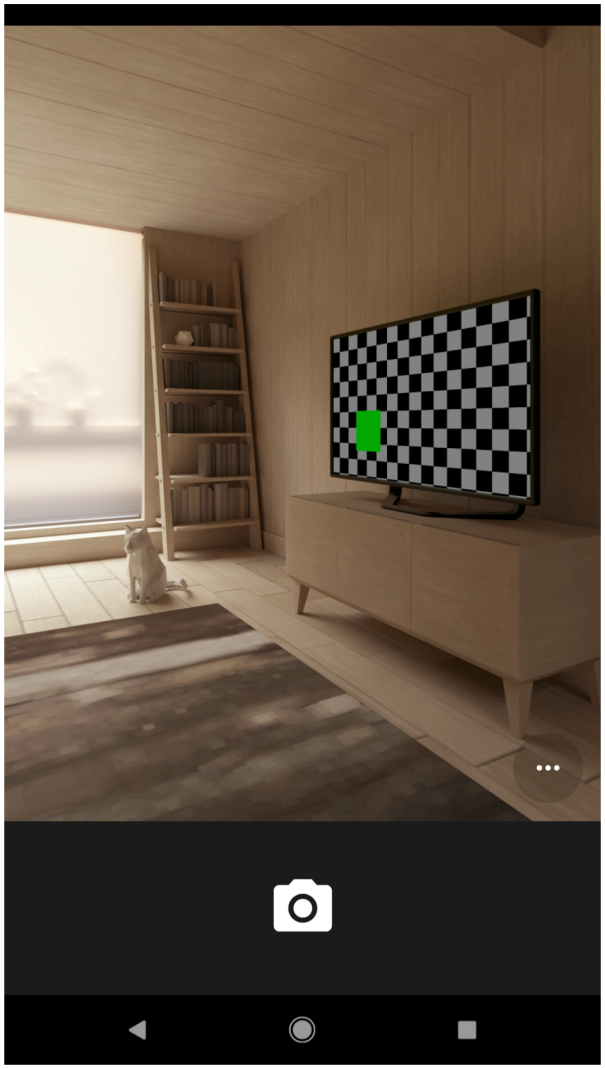
MainActivity.java (3)

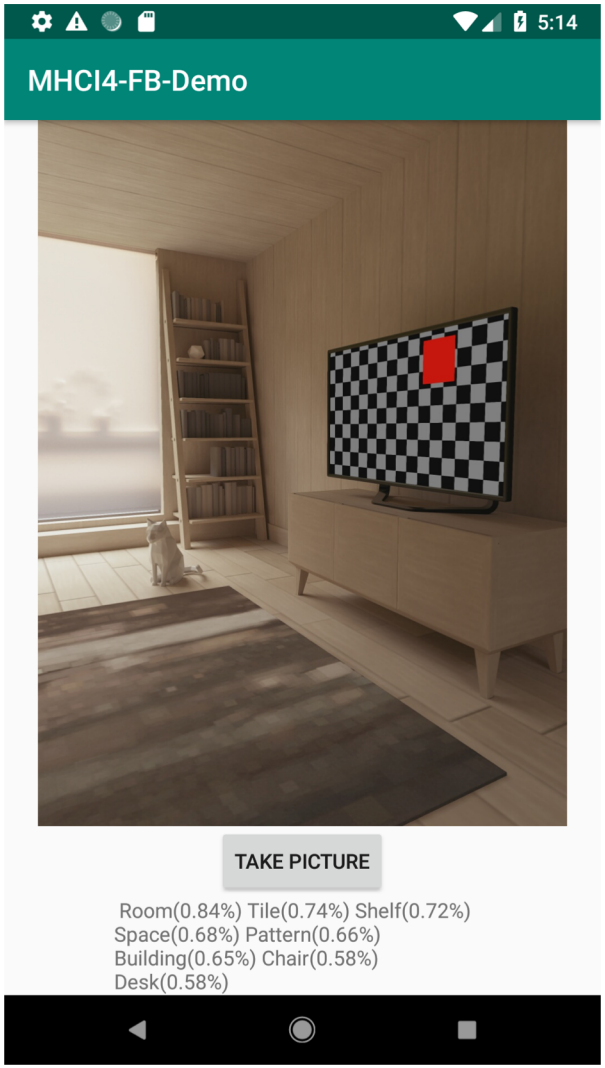
```
public void takePicture(View view) throws IOException {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, FileProvider.getUriForFile(this,
"social.uog.mhci.fileprovider", createMediaFile()));
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, 1);
    }
}
```

MainActivity.java (4)

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
   if (requestCode == 1 && resultCode == RESULT_OK) {
      Uri photoUri = Uri.fromFile(new File(currentPhotoPath));
      ((ImageView) findViewById(R.id.image_preview)).setImageURI(photoUri);
      try {
         FirebaseVisionImage image = FirebaseVisionImage.fromFilePath(this, photoUri);
         FirebaseVisionLabelDetector detector = FirebaseVision.getInstance().getVisionLabelDetector();
         Task<List<FirebaseVisionLabel>> result = detector.detectInImage(image)
                  .addOnSuccessListener(new OnSuccessListener<List<FirebaseVisionLabel>>() {
                     @Override
                     public void onSuccess(List<FirebaseVisionLabel> labels) {
                        TextView infoText = findViewById(R.id.text_info);
                        infoText.setText("");
                        for (FirebaseVisionLabel label: labels) {
infoText.setText(infoText.getText() + " " + label.getLabel() + "(" + String.format("%.02f",
label.getConfidence()) + "%)" );
                        }
                     }
                  });
      } catch (IOException e) { }
   }
}
```

Label Detection





Google Cloud Platform Education Grants

- Students' email domain(s): @student.gla.ac.uk
- Students can request coupons from the URL and redeem them until: 7/5/2019
- Coupons Valid Through: 7/1/2020
- Number of Coupons: 80
- Face Value of Coupon(s): USD 50.00
- http://mhci4-gcp.ugs.social