CS 4518 Mobile and Ubiquitous Computing
Lecture 6: Databases, Camera, Face Detection

Emmanuel Agu
Project 2
- Emailed out last week
- Should be done in groups of 5 or 6
- Due this Thursday, 11.59PM
- Can be done on your own computer. No need to test in zoolab
- Test on REAL PHONE!!

Groups that don’t have access to Android phone for project 2, 3 or final project should talk to me
Android Nerd Ranch
CriminalIntent Chapters
Skipped
Chapter 8: Displaying Lists with RecyclerView

- Skipped several **UI chapters**

- These features are programmed into the **CriminalIntent** code you will be given for project 2

- **RecyclerView** facilitates view of large dataset

- E.g. Allows crimes (title, date) in **CriminalIntent** to be listed
Chapter 9: Creating Android Layouts & Widgets

- Mostly already covered
- Does introduce Constraint Layout (specify widget positions using constraints)
Chapter 11: Using ViewPager

- ViewPager allows users to swipe left-right between screens
  - Similar to Tinder
- E.g. Users can swipe left-right between Crimes in CriminalIntent
Chapter 12: Dialogs

- Dialogs present users with a choice or important information
- DatePicker allows users to pick a date
- Users can pick a date on which a crime occurred in CriminalIntent

DatePicker also exists

TimePicker also exists
Chapter 13: The Toolbar

- Toolbar includes actions user can take
- In CriminalIntent, menu items for adding crime, navigate up the screen hierarchy
Android Nerd Ranch Ch 14
SQLite Databases
Background on Databases

- Relational DataBase Management System (RDBMS)
  - Introduced by E. F. Codd (Turing Award Winner)

- Relational Database
  - data stored in tables
  - relationships among data stored in tables
  - data can be accessed and viewed in different ways
Example Wines Database

- **Relational Data**: Data in different tables can be related

<table>
<thead>
<tr>
<th>Winery Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winery ID</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Region Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region ID</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Ref: Web Database Applications with PHP and MySQL, 2nd Edition, by Hugh E. Williams, David Lane
Keys

- Each table has a key
- **Key:** column used to uniquely identify each row

### Winery Table

<table>
<thead>
<tr>
<th>Winery ID</th>
<th>Winery name</th>
<th>Address</th>
<th>Region ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Moss Brothers</td>
<td>Smith Rd.</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Hardy Brothers</td>
<td>Jones St.</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Penfolds</td>
<td>Arthurton Rd.</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Lindemans</td>
<td>Smith Ave.</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Orlando</td>
<td>Jones St.</td>
<td>1</td>
</tr>
</tbody>
</table>

### Region Table

<table>
<thead>
<tr>
<th>Region ID</th>
<th>Region name</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Barossa Valley</td>
<td>South Australia</td>
</tr>
<tr>
<td>2</td>
<td>Yarra Valley</td>
<td>Victoria</td>
</tr>
<tr>
<td>3</td>
<td>Margaret River</td>
<td>Western Australia</td>
</tr>
</tbody>
</table>
SQL and Databases

- **SQL**: language used to manipulate Relational Database (RDBMS)

- SQL Commands:
  - **CREATE TABLE** - creates new database table
  - **ALTER TABLE** - alters a database table
  - **DROP TABLE** - deletes a database table
  - **SELECT** - get data from a database table
  - **UPDATE** - change data in a database table
  - **DELETE** - remove data from a database table
  - **INSERT INTO** - insert new data in a database table

### Region Table

<table>
<thead>
<tr>
<th>Region ID</th>
<th>Region name</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Barossa Valley</td>
<td>South Australia</td>
</tr>
<tr>
<td>2</td>
<td>Yarra Valley</td>
<td>Victoria</td>
</tr>
<tr>
<td>3</td>
<td>Margaret River</td>
<td>Western Australia</td>
</tr>
</tbody>
</table>
CriminalIntent Database

- **SQLite**: open source relational database
- SQLite implements subset of SQL (most but not all)
  - [http://www.sqlite.org/](http://www.sqlite.org/)
- Android includes a SQLite database
- **Goal**: Store crimes in CriminalIntent in SQLite database
- First step, define database table of **crimes**

<table>
<thead>
<tr>
<th>_id</th>
<th>uuid</th>
<th>title</th>
<th>date</th>
<th>solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13090636733242</td>
<td>Stolen yogurt</td>
<td>13090636733242</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>13090732131909</td>
<td>Dirty sink</td>
<td>13090732131909</td>
<td>1</td>
</tr>
</tbody>
</table>
CriminalIntent Database Schema

- Create **CrimeDbSchema** class to store crime database
- Define fields/columns of the Crimes database table

```java
public class CrimeDbSchema {
    public static final class CrimeTable {
        public static final String NAME = "crimes"; // Name of Table

        public static final class Cols {
            public static final String UUID = "uuid"; // uuid
            public static final String TITLE = "title"; // title
            public static final String DATE = "date"; // date
            public static final String SOLVED = "solved"; // solved
        }
    }
}
```

Each Crimes Table has the following fields/columns:

<table>
<thead>
<tr>
<th>id</th>
<th>uuid</th>
<th>title</th>
<th>date</th>
<th>solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13090636733242</td>
<td>Stolen yogurt</td>
<td>13090636733242</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>13090732131909</td>
<td>Dirty sink</td>
<td>13090732131909</td>
<td>1</td>
</tr>
</tbody>
</table>
SQLiteOpenHelper

- **SQLiteOpenHelper** class used for database creation, opening and updating a **SQLiteDatabase**
- In **CriminalIntent**, create subclass of **SQLiteOpenHelper** called **CrimeBaseHelper**

```java
public class CrimeBaseHelper extends SQLiteOpenHelper {
    private static final int VERSION = 1;
    private static final String DATABASE_NAME = "crimeBase.db";

    public CrimeBaseHelper(Context context) {
        super(context, DATABASE_NAME, null, VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
}
```

- Used to create the database (to store Crimes)
- Called the first time database is created
Use CrimeBaseHelper to open SQLite Database

```java
public class CrimeLab {
    private static CrimeLab sCrimeLab;

    private List<Crime> mCrimes;
    private Context mContext;
    private SQLiteDatabase mDatabase;

    private CrimeLab(Context context) {
        mContext = context.getApplicationContext();
        mDatabase = new CrimeBaseHelper(mContext)
            .getWritableDatabase();
        mCrimes = new ArrayList<>();
    }
}
```
Create CrimeTable in onCreate()
In Android, writing to databases is done using class `ContentValues`

`ContentValues` is key-value pair

Create method to create `ContentValues` instance from a `Crime`

```java
public Crime getCrime(UUID id) {
    return null;
}

private static ContentValues getContentValues(Crime crime) {
    ContentValues values = new ContentValues();
    values.put(CrimeTable.Cols.UUID, crime.getId().toString());
    values.put(CrimeTable.Cols.TITLE, crime.getTitle());
    values.put(CrimeTable.Cols.DATE, crime.getDate().getTime());
    values.put(CrimeTable.Cols.SOLVED, crime.isSolved() ? 1 : 0);
    return values;
}
```
Firebase Cloud API
Firebase

- Mobile cloud backend service for
  - Analytics
  - Messaging
  - Authentication
  - Database
  - Crash reporting, etc

- Previously 3rd party company
- Acquired by Google in 2014
  - Now part of Google. See https://firebase.google.com/
  - Fully integrated, could speed up development. E.g. final project
Firebase

- Relatively easy programming, few lines of code
- E.g. to create database

```java
FirebaseDatabase database = FirebaseDatabase.getInstance();
// write
database.child("users").child("userId").setValue(user);

// read / listen
database.child("users").addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // ...
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {}
});
```
The Mobile Camera
Interesting application
Word Lens Feature of Google Translate

- Word Lens: translates text/signs in foreign Language in real time
- Example use case: tourist can understand signs, restaurant menus
- Uses Optical Character Recognition technology
- Google bought company in 2014, now part of Google Translate
Camera: Taking Pictures
Taking Pictures with Camera

Ref: https://developer.android.com/training/camera/photobasics.html

- How to take photos from your app using Android Camera app
- 4 Steps:
  1. Request the camera feature
  2. Take a Photo with the Camera App
  3. Get the Thumbnail
  4. Save the Full-size Photo
1. Request the Smartphone Camera Feature
Ref: https://developer.android.com/training/camera/photobasics.html

- If your app takes pictures using the phone’s Camera, you can allow only devices with a camera find your app while searching Google Play Store
- How?
- Make the following declaration in AndroidManifest.xml

```xml
<manifest ... >
  <uses-feature android:name="android.hardware.camera"
    android:required="true" />
  ...
</manifest>
```
2. Capture an Image with the Camera App

Ref: https://developer.android.com/training/camera/photobasics.html

- To take a picture, your app needs to send an implicit Intent requesting for a picture to be taken (i.e. action = capture an image).
- Call `startActivityForResult()` with a Camera intent since a picture will be sent back.
- Potentially, multiple apps/activities can handle this/take a picture.
- Check that at least one Activity that can handle the request to take a picture using `resolveActivity`.

```
startActivityForResult

Your App

Android Camera app

onActivityResult
```

Big picture: taking a picture
Code to Take a Photo with the Camera App

Ref: https://developer.android.com/training/camera/photobasics.html

```java
static final int REQUEST_IMAGE_CAPTURE = 1;

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
    }
}
```

1. Build Intent, action = capture an image

2. Check that there’s at least 1 Activity that can handle request to capture an image (Avoids app crashing if no camera app available)

3. Send Intent requesting an image to be captured (usually handled by Android’s Camera app)

Your App

startActivityForResult

Android Camera app

onActivityResult

onActivityResult
3. Get the Thumbnail

Ref: https://developer.android.com/training/camera/photobasics.html

- Android Camera app returns thumbnail of photo (small bitmap)

- Thumbnail bitmap returned in “extra” of Intent delivered to onActivityResult()
4. Save Full-Sized Photo

Ref: https://developer.android.com/training/basics/data-storage/files.html

- Android Camera app saves full-sized photo in a filename you give it
- We need phone owner’s permission to write to external storage
- Android systems have:
  - **Internal storage**: data stored here is available by only your app
  - **External storage**: available stored here is available to all apps
- Would like all apps to read pictures this app takes, so use external storage
Save Full-Sized Photo
Ref: https://developer.android.com/training/basics/data-storage/files.html

Android Camera app can save full-size photo to

1. **Public external storage** (shared by all apps)
   - `getExternalStoragePublicDirectory()`
   - Need to get permission

2. **Private storage** (Seen by only your app, deleted when your app uninstalls):
   - `getExternalFilesDir()`

Either way, need phone owner’s permission to write to external storage

In AndroidManifest.xml, make the following declaration

```xml
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
...
</manifest>
```
Saving Full Sized Photo

Ref: https://developer.android.com/training/camera/photobasics.html

```java
static final int REQUEST_TAKE_PHOTO = 1;

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    // Ensure that there's a camera activity to handle the intent
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        // Create the File where the photo should go
        File photoFile = null;
        try {
            photoFile = createImageFile();
        } catch (IOException ex) {
            // Error occurred while creating the File
            ...
        }
        // Continue only if the File was successfully created
        if (photoFile != null) {
            Uri photoURI = FileProvider.getUriForFile(this,
                                                        "com.example.android.fileprovider",
                                                        photoFile);
            takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
            startActivityForResult(takePictureIntent, REQUEST_TAKE_PHOTO);
        }
    }
}
```
Taking Pictures: Bigger Example
Taking Pictures with Intents
Ref: Ch 16 Android Nerd Ranch 3rd edition

- Would like to take picture of “Crime” to document it
- Use implicit intent to start Camera app from our CrimeIntent app
- Recall: Implicit intent used to call component in different activity

![Camera app launch](image.png)

Click here to take picture
Launches Camera app
Create Placeholder for Picture

- Modify layout to include
  - ImageView for picture
  - Button to take picture
Create Layout for Thumbnail and Button

- First, build out left side
Create Title and Crime Entry EditText

- Build out right side

![Diagram of UI layout with EditText and other UI components]
Get Handle of Camera Button and ImageView

- To respond to Camera Button click, in camera fragment, need handles to:
  - Camera button
  - ImageView

```java
private Button mSuspectButton;
private Button mReportButton;
private ImageButton mPhotoButton;
private ImageView mPhotoView;
...
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    ...
    PackageManager packageManager = getActivity().getPackageManager();
    if (packageManager.resolveActivity(pickContact,
        PackageManager.MATCH_DEFAULT_ONLY) == null) {
        mSuspectButton.setEnabled(false);
    }
    mPhotoButton = (ImageButton) v.findViewById(R.id.crime_camera);
    mPhotoView = (ImageView) v.findViewById(R.id.crime_photo);
    return v;
}
```
Firing Camera Intent

private static final int REQUEST_DATE = 0;
private static final int REQUEST_CONTACT = 1;
private static final int REQUEST_PHOTO = 2;
...
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

    mPhotoButton = (ImageButton) v.findViewById(R.id.crime_camera);
    final Intent captureImage = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    boolean canTakePhoto = mPhotoFile != null &&
        captureImage.resolveActivity(packageManager) != null;
    mPhotoButton.setEnabled(canTakePhoto);

    mPhotoButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Uri uri = FileProvider.getUriForFile(getActivity(),
                "com.bignerdranch.android.criminalintent.fileprovider",
                mPhotoFile);
            captureImage.putExtra(MediaStore.EXTRA_OUTPUT, uri);

            List<ResolveInfo> cameraActivities = getActivity().
                getPackageManager().queryIntentActivities(captureImage,
                PackageManager.MATCH_DEFAULT_ONLY);

            for (ResolveInfo activity : cameraActivities) {
                getActivity().grantUriPermission(activity.activityInfo.packageName,
                    uri, Intent.FLAG_GRANT_WRITE_URI_PERMISSION);
            }

            startActivityForResult(captureImage, REQUEST_PHOTO);
        }
    });

    mPhotoView = (ImageView) v.findViewById(R.id.crime_photo);
    return v;
}
Declaring Features

- Declaring “uses-features”. But “android:required=false” means app prefers to use this feature.
- Phones without a camera will still “see” and on Google Play Store and can download this app.

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.bignerdranch.android.criminalintent">

  <uses-feature android:name="android.hardware.camera"
    android:required="false"/>

</manifest>
```
Face Recognition
Face Recognition

- Answers the question:

  **Who** is this person in this picture?

  **Example answer:** John Smith

- Compares unknown face to database of faces with known identity

- Neural networks/deep learning now makes comparison faster
FindFace App: Stalking on Steroids?

- See stranger you like? Take a picture
- App searches 1 billion pictures using neural networks < 1 second
- Finds person’s picture, identity, link on VK (Russian Facebook)
  - You can send friend Request
- ~ 70% accurate!
- Can also upload picture of celebrity you like
- Finds 10 strangers on Facebook who look similar, can send friend request
FindFace App

- Also used in law enforcement
  - Police identify criminals on watchlist

Face Detection
Mobile Vision API
https://developers.google.com/vision/

- **Face Detection**: Are there [any] faces in this picture?
- **How?** Locate face in photos and video and
  - **Facial landmarks**: Eyes, nose and mouth
  - **State of facial features**: Eyes open? Smiling?
Face Detection: Google Mobile Vision API
Ref: https://developers.google.com/vision/face-detection-concepts

- Detects faces:
  - reported at a position, with size and orientation
  - Can be searched for landmarks (e.g. eyes and nose)

<table>
<thead>
<tr>
<th>Orientation</th>
<th>Landmarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euler Y angle</td>
<td>detectable landmarks</td>
</tr>
<tr>
<td>&lt; -36 degrees</td>
<td>left eye, left mouth, left ear, nose base, left cheek</td>
</tr>
<tr>
<td>-36 degrees to -12 degrees</td>
<td>left mouth, nose base, bottom mouth, right eye, left eye, left cheek, left ear tip</td>
</tr>
<tr>
<td>-12 degrees to 12 degrees</td>
<td>right eye, left eye, nose base, left cheek, right cheek, left mouth, right mouth, bottom mouth</td>
</tr>
<tr>
<td>12 degrees to 36 degrees</td>
<td>right mouth, nose base, bottom mouth, left eye, right eye, right cheek, right ear tip</td>
</tr>
<tr>
<td>&gt; 36 degrees</td>
<td>right eye, right mouth, right ear, nose base, right cheek</td>
</tr>
</tbody>
</table>
Google Mobile Vision API

- Mobile Vision API also does:
  - **Face tracking**: detects faces in consecutive video frames
  - **Classification**: Eyes open? Face smiling?

- **Classification**:
  - Determines whether a certain facial characteristic is present
  - API currently supports 2 classifications: eye open, smiling
  - Results expressed as a confidence that a facial characteristic is present
    - Confidence > 0.7 means facial characteristic is present
    - E.g. > 0.7 confidence means it’s likely person is smiling

- Mobile vision API does face **detection** but **NOT recognition**
Face Detection

- **Face detection**: Special case of object-class detection
- **Object-class detection task**: find locations and sizes of all objects in an image that belong to a given class.
  - E.g: bottles, cups, pedestrians, and cars
- **Object matching**: Objects in picture compared to objects in database of labelled pictures
Mobile Vision API: Other Functionality

- Barcode scanner
- Recognize text
Face Detection Using Google’s Mobile Vision API
Getting Started with Mobile Vision Samples
https://developers.google.com/vision/android/getting-started

- Get **Android Play Services SDK** level 26 or greater
- Download mobile vision samples from github
**Creating the Face Detector**
Ref: https://developers.google.com/vision/android/detect-faces-tutorial

- In app’s **onCreate** method, create face detector

```java
FaceDetector detector = new FaceDetector.Builder(context)
    .setTrackingEnabled(false) \[
    \text{Don’t track points} \]
    .setLandmarkType(FaceDetector.ALL_LANDMARKS) \[
    \text{Detect all landmarks} \]
    .build();
```

- **detector** is base class for implementing specific detectors. E.g. face detector, bar code detector
- Tracking finds same points in multiple frames (continuous)
- Detection works best in single images when **trackingEnabled** is false
Detecting Faces and Facial Landmarks

- Create Frame (image data, dimensions) instance from bitmap supplied
  
  ```java
  Frame frame = new Frame.Builder().setBitmap(bitmap).build();
  ```

- Call detector synchronously with frame to detect faces
  
  ```java
  SparseArray<Face> faces = detector.detect(frame);
  ```

- Detector takes **Frame** as input, outputs array of **Faces** detected
- **Face** is a single detected human face in image or video
- Iterate over array of faces, landmarks for each face, and draw the result based on each landmark’s position

```
for (int i = 0; i < faces.size(); ++i) {
    Face face = faces.valueAt(i);
    for (Landmark landmark : face.getLandmarks()) {
        int cx = (int) (landmark.getPosition().x * scale);
        int cy = (int) (landmark.getPosition().y * scale);
        canvas.drawCircle(cx, cy, 10, paint);
    }
}
```
Other Stuff

- To count faces detected, call `faces.size()`. E.g.

```java
TextView faceCountView = (TextView) findViewById(R.id.face_count);
faceCountView.setText(faces.size() + " faces detected");
```

- Querying Face detector’s status

```java
if (!detector.isOperational()) {
    // ...
}
```

- Releasing Face detector (frees up resources)

```java
detector.release();
```
Detect & Track Multiple Faces in Video

- Can also track multiple faces in image sequences/video, draw rectangle round each one
Face Interpretation
Visage Face Interpretation Engine

- Real-time face interpretation engine for smartphones
  - Tracking user’s 3D head orientation + facial expression

- Facial expression?
  - angry, disgust, fear, happy, neutral, sad, surprise
  - Use? Can be used in Mood Profiler app

Facial Expression Inference

- Active appearance model
  - Describes 2D image as triangular mesh of landmark points
- 7 expression classes: angry, disgust, fear, happy, neutral, sad, surprise
- Extract triangle shape, texture features
- Classify features using Machine learning
### Classification Accuracy

<table>
<thead>
<tr>
<th>Expressions</th>
<th>Anger</th>
<th>Disgust</th>
<th>Fear</th>
<th>Happy</th>
<th>Neutral</th>
<th>Sadness</th>
<th>Surprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy(%)</td>
<td>82.16</td>
<td>79.68</td>
<td>83.57</td>
<td>90.30</td>
<td>89.93</td>
<td>73.24</td>
<td>87.52</td>
</tr>
</tbody>
</table>
References

- Busy Coder’s guide to Android version 4.4
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014