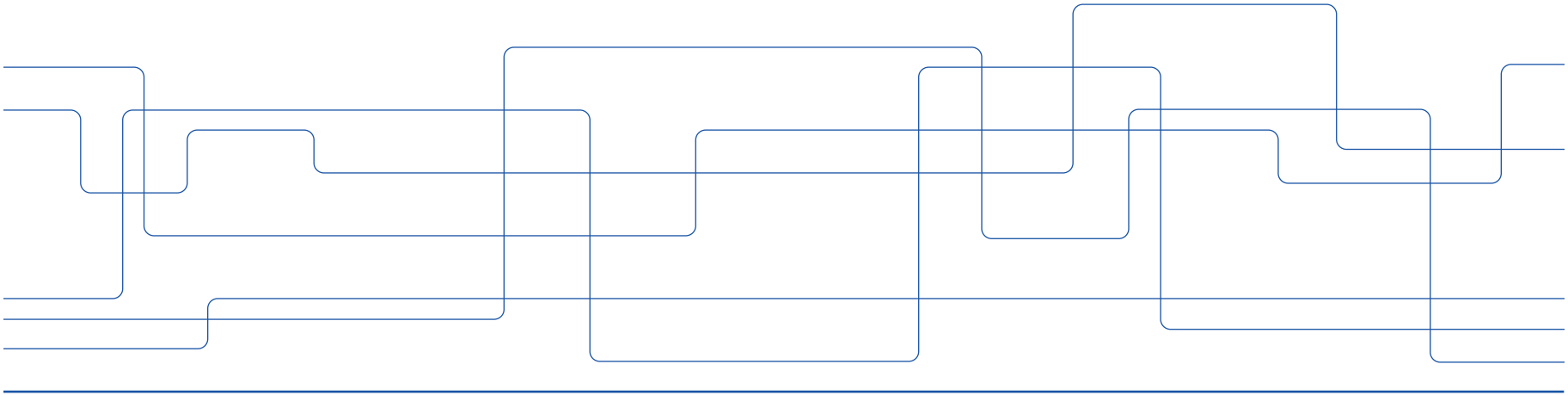
Mobile Application Programming II

Anders Västberg <vastberg@kth.se>



Android User Interface

- Material Design
  - Android UI design guidelines
  - Visual language
  - Classic principles of good design
    - > *Combined with innovation and possibilities of technology and science*
  - https://material.io/design/introduction

Classical Principles of Design

- Balance
- Proximity
- Alignment
- Repetition
- Contrast
- Space
- https://vimeo.com/32944253

Material Design [1]

- A design language developed using grid-based layouts, responsive animations and transitions, padding, and depth effects such as lighting and shadows
- Basic principles:
  - Material is the metaphor
  - Bold, graphic, intentional
  - Motion provides meaning

Material Design [1]

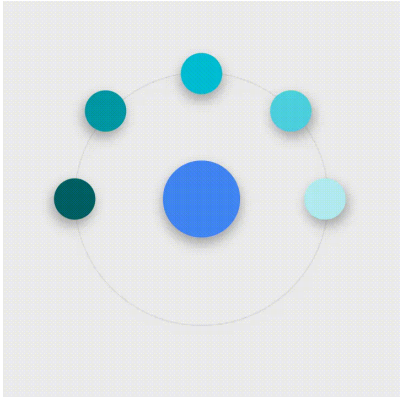
- Elements in your Android app should behave similarly to real world materials
- Cast shadows
- Occupy space
- Interact with each other

Material Design – Bold, Graphical, Intentional [1]

- Choose colors deliberately
- Fill screen edge to edge
- Use large-scale typography
- Use white space intentionally
- Emphasize user action
- Make functionality obvious

Material Design – Motion [1]

- Maintain continuity
- Highlight elements or actions
- Transition naturally between actions or states
- Draw focus
- Organize transitions
- Responsive feedback



Components [2]

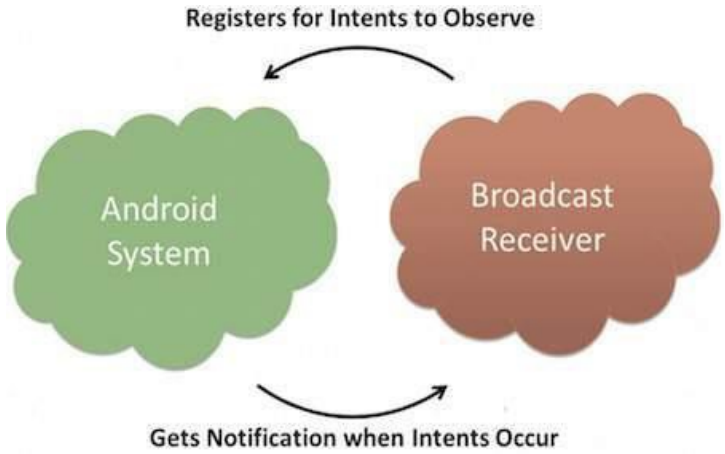
- Interactive building blocks for creating a user interface
- Built in state system
  - Communicate focus, selection, activation, error hover, press, drag and disabled states
- Available for Android, iOS, Flutter, and the web.

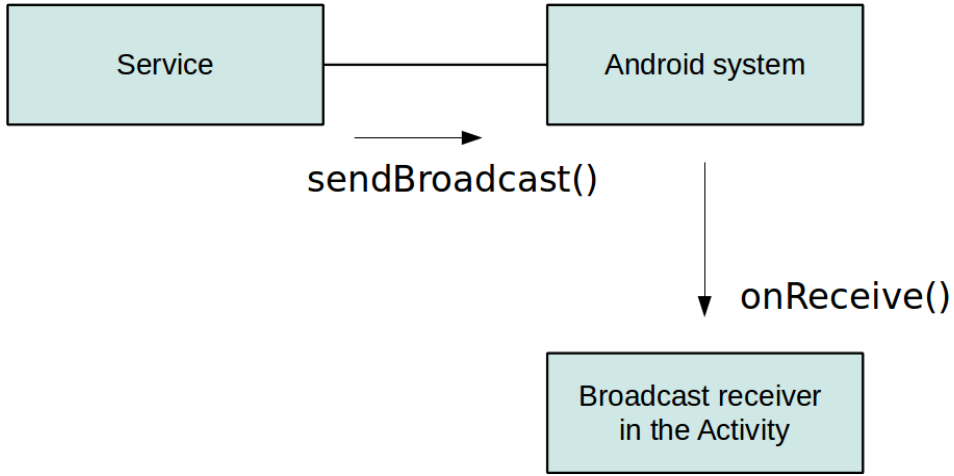
Theming [2]

- Easy to customize Material Design
- Color System
  - primary, secondary, surface, background, and error.
  - Complementary color to indicate stacking of components
- Typography styles
  - 13 typography styles
- Shapes
- Examples: https://material.io/design/material-studies/about-our-material-studies.html

Broadcast Receivers

- A broadcast receiver
  - component that responds to system-wide broadcast announcements
  - Sent when event occur that can affect other apps
    - > *Power connected/disconnected, headphones connected/disconnected, …*





Broadcast Receivers [2]

- broadcasts are messages sent by the Android system and Android apps when an event of interest occurs.
- Broadcasts are wrapped in an `Intent` object
  - Contains event details
- Deliver a broadcast to other apps by passing an Intent to `sendBroadcast()`
- `broadcastReceiver` is the base class for code that receives broadcast intents.
- Types of broadcast:
  - System broadcast
  - Custom broadcast

System broadcasts [3]

- System broadcast are the messages sent by the Android system, when a system event occurs, that might affect your app.

Few examples:

- An Intent with action, `ACTION_BOOT_COMPLETED` is broadcasted when the device boots.
- An Intent with action, `ACTION_POWER_CONNECTED` is broadcasted when the device is connected to the external power.

Custom Broadcasts [3]

- Custom broadcasts are broadcasts that your app sends out, similar to the Android system.
- For example, when you want to let other app(s) know that some data has been downloaded by your app, and its available for their use.

Send a Custom Broadcast [3]

Android provides three ways for sending a broadcast:

- Ordered broadcast.
- Normal broadcast.
- Local broadcast.

Restricting Broadcasts [3]

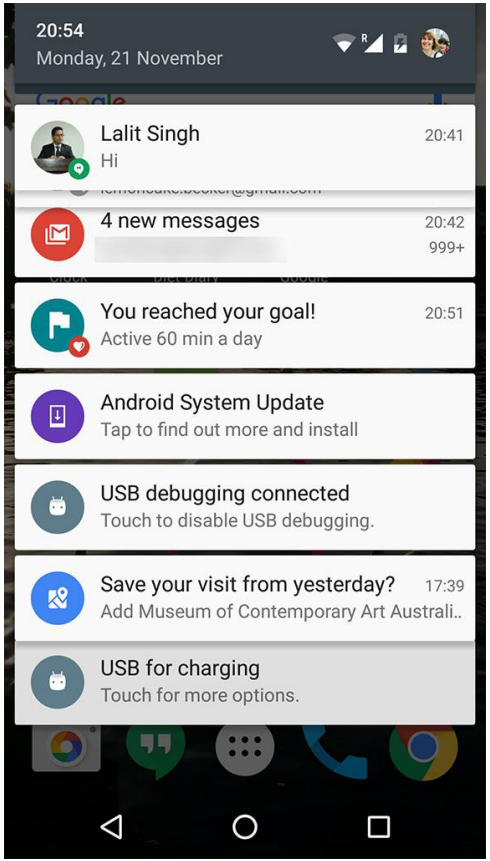
- Restricting your broadcast is strongly recommended.
- An unrestricted broadcast can pose a security threat.
- For example: If your apps' broadcast is not restricted and includes sensitive information, an app that contains malware could register and receive your data.

Broadcasts: Best Practice [3]

- Make sure namespace for intent is unique and you own it.
  - Prefix `String` constants with you app's package name
- Restrict broadcast receivers.
- Other apps can respond to broadcast your app sends —use permissions to control this.
- Prefer dynamic receivers over static receivers.
- Never perform a long running operation inside your broadcast receiver.
  - `onReceive` method runs in the main UI thread
  - use `JobScheduler` or `WorkManger` instead

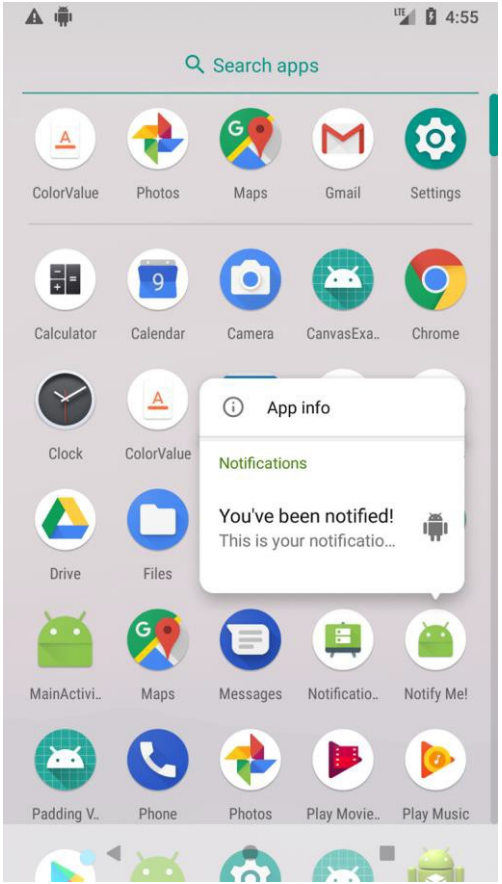
Notifications

- Notification allows apps or services associated with an app to inform the user of an event
- Notification icons appear on the left side of the status bar
- Users can swipe down on the status bar to open the notification drawer, where they can view more details and take actions with the notification.
- Users can tap the notification to open your app or take an action directly from the notification.



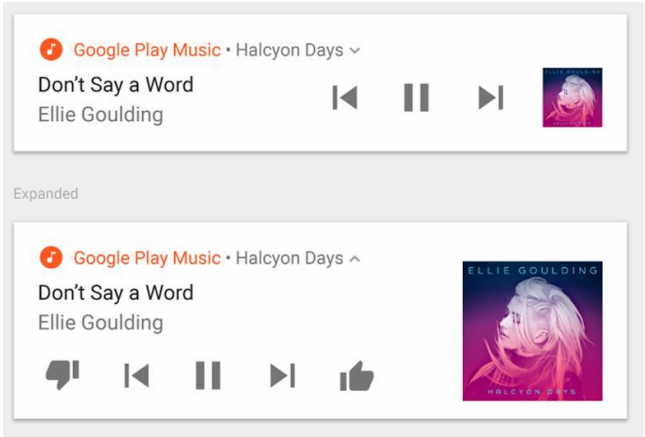
Notifications

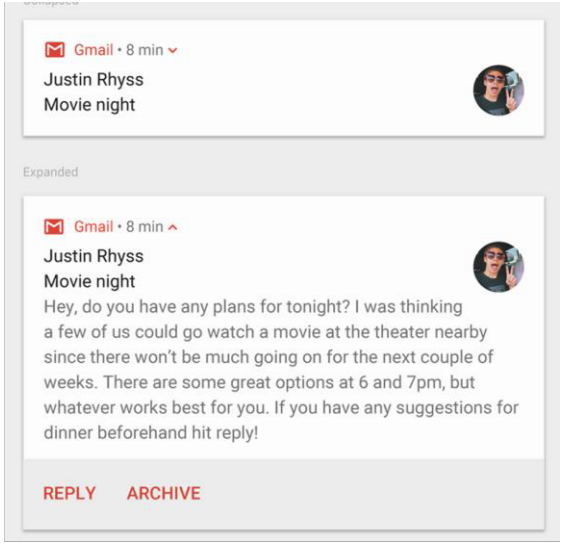
- Badges - notification dots
  - introduced in Android 8.0
  - app icon automatically shows a badge

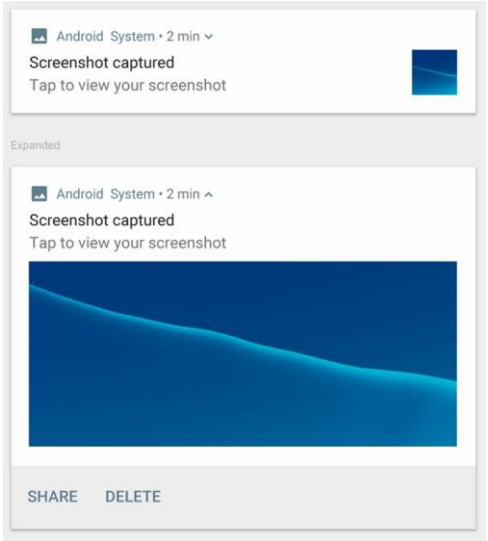


Notifications

- Expandable notifications
  - Introduced in Android 4.1
  - Use them sparingly as they take up more space and attention







Threads

- Threads - Independent path of execution in a running program
  - to run long running threads
  - avoids “Application Non Responsive” dialogs
- background threads - execute long running tasks on a background thread
- asyncTask - use asyncTask to implement basic background tasks
- examples
  - UI/main application thread
  - worker threads (downloads files, read/write db)

Threads

- the main thread
  - app runs on java thread called “main” or “UI thread”
  - responds to user actions by handling UI events
- the main thread must be fast
  - if the UI waits too long for an operation to finish, it becomes unresponsive
- two rules for Android threads
  - do not block the UI thread
  - do not access the Android UI toolkit from outside the UI thread

AsyncTask

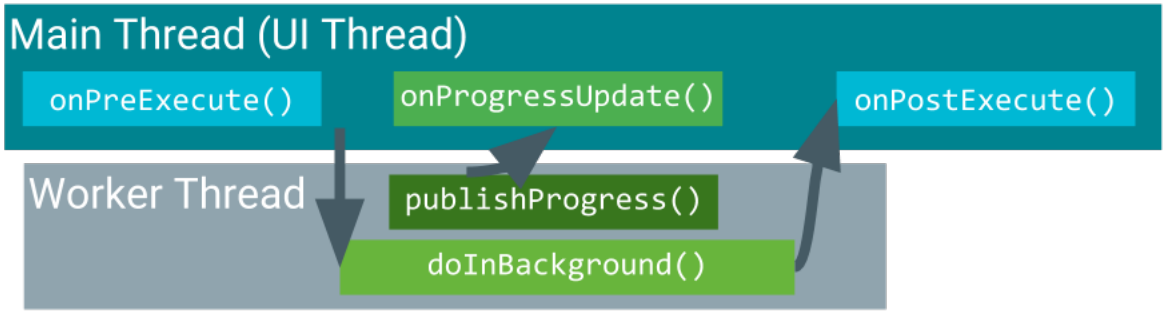
- Why `AsyncTask`
  - to manage your own threads
  - use `AsyncTask` to implement basic background tasks
- What `AsyncTask` does
  - It makes it really convenient to be in an Activity and to test if you can run an operation asynchronously.
  - an `AsyncTask` allows you to execute background tasks sequentially, in parallel, or through your own thread Pool.

AsyncTask

- to keep the user experience (UX) running smoothly, the Android framework provides a helper class called `AsyncTask`, which processes work off of the UI thread
- `AsyncTasks` should ideally be used for short operations (a few seconds at the most)
- The `AsyncTask` class is a wrapper around standard Java Threading; it encapsulates the most common pattern of executing background work on a child Thread, before syncing with the UI Thread to deliver progress and the final result.

AsyncTask

- an asynchronous task important methods
  - doInBackground,
  - onPostExecute



ContentProviders

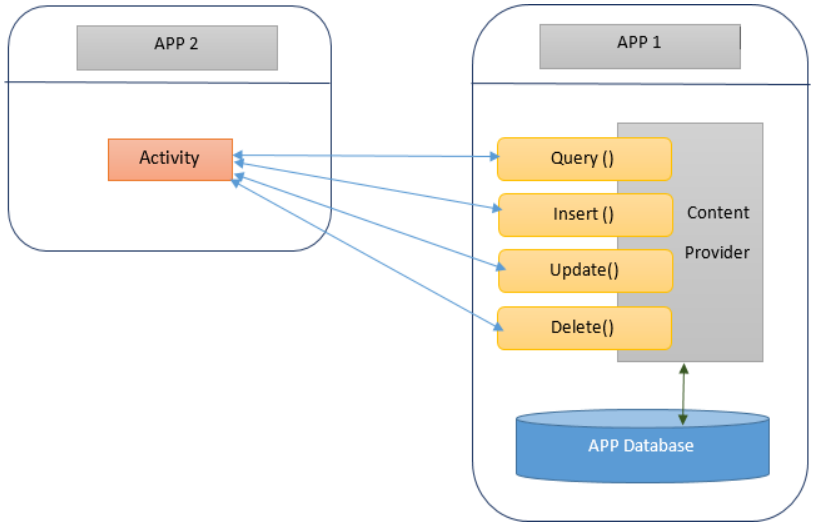
Enables sharing of data
across applications

- E.g. address book, photo gallery

Provides uniform APIs for:

- querying

delete, update and insert.
Content is represented by
URI and MIME type



Data Storage Options in Android

- types of data storage options in android
  - android file system - private data in key-value pairs
    - *internal storage - private data on device memory*
    - *external storage - public data on device or external*
  - SQLite database - structured data in a private database
  - Content providers - store privately and make available publicly
  - other options
    - *firebase real-time database*
    - *cloud backup*

SQLite

- SQLite - a relational database, standards-compliant, implementing most of the SQL standard
- Use - to store and manage complex, structured application data
- SQLite databases are private, accessible only by the application that created them
- SQLite reduces external dependencies, minimizes latency, and simplifies transaction locking and synchronization.
- SQLiteOpenHelper is an abstract class used to help implement the best practice pattern for creating, opening, and upgrading databases.
- SQLite supports Adding, Updating, and Deleting Rows

SQLite

`SQLiteOpenHelper`

- Defining the database contract - public methods and constants required to interact with the underlying data
- SQLiteOpenHelper is an abstract class used to help implement the best practice pattern for creating, opening, and upgrading databases.

```
public class DBHelper extends SQLiteOpenHelper {

    public static final String DATABASE_NAME = "MyDBName.db";
    public static final String CONTACTS_TABLE_NAME = "contacts";
    public static final String CONTACTS_COLUMN_ID = "id";
    public static final String CONTACTS_COLUMN_NAME = "name";
    public static final String CONTACTS_COLUMN_EMAIL = "email";
    public static final String CONTACTS_COLUMN_STREET = "street";
    public static final String CONTACTS_COLUMN_CITY = "place";
    public static final String CONTACTS_COLUMN_PHONE = "phone";
    private HashMap hp;

    public DBHelper(Context context) {
        super(context, DATABASE_NAME , factory: null, version: 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // TODO Auto-generated method stub
        db.execSQL(
                "create table contacts " +
                        "(id integer primary key, name text,phone text,email text, street text,place text)"
        );
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // TODO Auto-generated method stub
        db.execSQL("DROP TABLE IF EXISTS contacts");
        onCreate(db);
    }
```

Firebase [4]

- Web and mobile application platform
- Three basic groups of applications
  - Build
  - Release and Monitor
  - Engage

Example of products [4]

Build products

- Firebase real-time database
  - API that synchronizes application data across iOS, Android and Web devices
  - Stores data in a Firebase's cloud
- Cloud functions and storage
  - Write and run app logic server-side without own server
  - Store user-generated content
- Authentication
  - Authentication and sign-in
- Release and Monitor products
- Engage products

Build your app

- Cloud Firestore — iOS, Android, </>
- ML Kit — iOS, Android
- Cloud Functions — iOS, Android, </>, C++, Unity
- Authentication — iOS, Android, </>, C++, Unity
- Hosting — </>
- Cloud Storage — iOS, Android, </>, C++, Unity
- Realtime Database — iOS, Android, </>, C++, Unity

Improve app quality

- Crashlytics — iOS, Android
- Performance Monitoring — iOS, Android
- Test Lab — iOS, Android

Grow your business

- Analytics — iOS, Android, C++, Unity
- Predictions — iOS, Android, C++, Unity
- Firebase A/B Testing — iOS, Android, C++, Unity
- Cloud Messaging — iOS, Android, </>, C++, Unity
- Remote Config — iOS, Android, C++, Unity
- Dynamic Links — iOS, Android, C++, Unity
- App Indexing — iOS, Android
- Invites — iOS, Android, C++, Unity

Firebase Real-time Database [4]

- Firebase Realtime Database: cloud-hosted NoSQL
  - database data is synced across all clients in real time
- The database is stored locally on each device as JSON files and continuously synchronized across devices including mobile and web clients.
- Firebase Console to define data structure & access rules.
- Actions performed like Adding, Modifying, Deleting, and Querying Data from a Firebase Realtime Database
- The added new data becomes an element in the JSON tree, and accessible using the associated key.
- To write to a Firebase Database within Android app, use
  - the static getInstance method:
  - FirebaseDatabase database = FirebaseDatabase.getInstance();

Cloud Firestore [4]

- Firestore: flexible, scalable NoSQL cloud database to store and sync data for client- and server-side development.
- Designed to be highly scalable and supports more expressive and efficient(doesn’t require retrieving the entire collection)
- Cloud Firestore offers seamless integration with other Firebase and Google Cloud Platform products, including Cloud Functions.
- In addition to Android, web, and iOS SDKs, Firestore APIs
- are available in Node.js, Java, Python, and Go.

React Native [5]

- Open source platform for developing native mobile applications
- Uses JavaScript
  - And syntax extension JSX
- Based on the React JavaScript library for building user interfaces
- Can wrap native code written in Java for Android or Swift for iOS

References

[1] Slides 5.2 Material Design, https://docs.google.com/presentation/d/1TFKxDDS8KZEjbPgr8jLcsqVdKTDkNvsYHQz7vNoMGWs/edit#slide=id.g116d7d9d49_3_13, retrieved 2021-02-08

[2] Material Design, Introduction, https://material.io/design/introduction#theming, retrieved 20201-02-08

[3] Slides 7.3 Broadcasts, https://docs.google.com/presentation/d/1qF9Yeau7uHIP7_aOHWgPU_RnfxACZzGyAZIzcJWz0R0/edit#slide=id.g116d7d9d49_3_13, retrieved 2021-02-08

[4] Firebase webpage, https://firebase.google.com, retrieved 2021-02-08

[5] React Native, https://reactnative.dev, retrieved 2021-02-09