APPLICATION DEVELOPMENT FOR MOBILE AND UBIQUITOUS COMPUTING

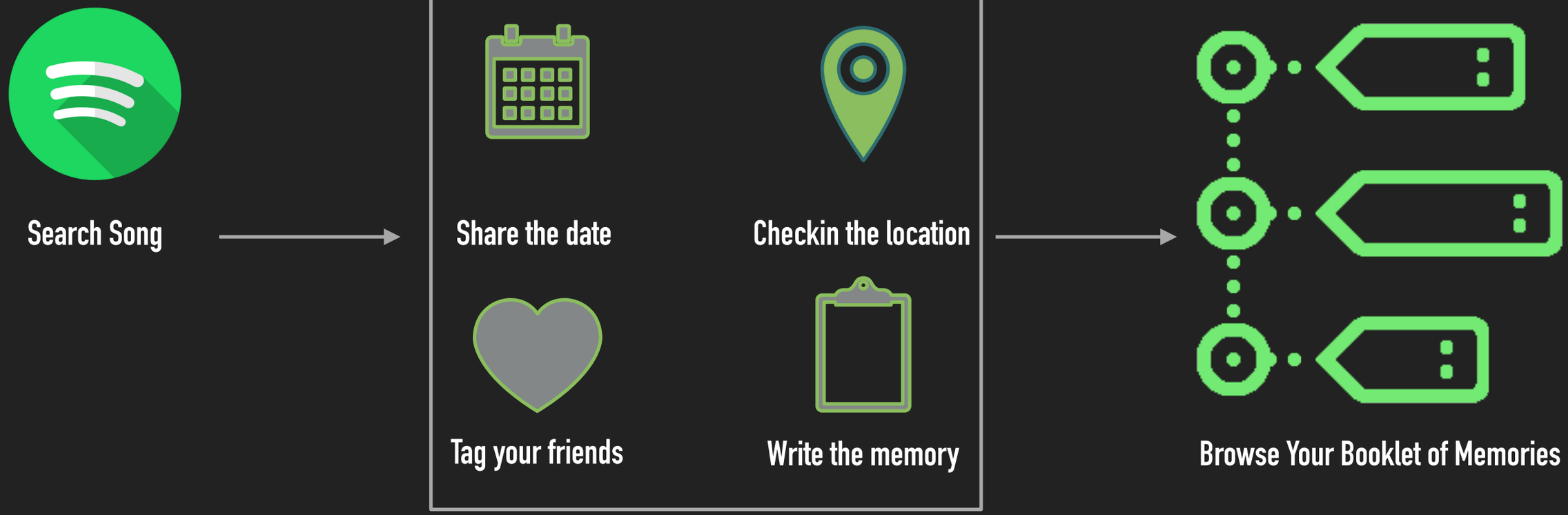
MEMOSONG

Adaptation Concepts

Group No. 9
Team: Shermin Shojaei - Niloofar Gheibi

Dresden - 15.12.17

APP SCENARIO



PHYSICAL CONTEXT

Location Time (Date)

- Use Google places API

- Usage of **Name**

- For more accuracy in future : Latitude/Longitude

PERSONAL CONTEXT

Spotify Account (email)

- Use Spotify Web API
- Usage of Track
- Users

Searching the Tracks

Playing the track via Song's URI

Extracting the Name of the Song

TECHNICAL CONTEXT

Network

Available Web Service

- Tracking Network Status
- Checking Availability of Web Server
- Http POST/PUT requests

OFFLINE CHALLENGE

- Mark the job as requires network : Storing in the Web Server

```
public StoreWeb(String... memo) {

    super(new Params(Priority.HIGH).requireNetwork().persist());
    this.memo = memo;

}
```

Job Manager will not run the job until the device has network connection.

Job Manager queries Connectivity Manager to check if network is available or not

- Triggering Local Storing and Server Storing Jobs

```
StoreInLocalDatabase.StoreInLocalDatabaseTask(Date.getText().toString(),friend.id,Place.getName().toString(),
            Memory.getText().toString(),Privacy,Song.getText().toString(),email);

StoreInWebServerDatabase.StoreInWebServerDatabaseTask(Date.getText().toString(),friend.id,Place.getName().toString(),
            Memory.getText().toString(),Privacy,Song.getText().toString(),email);
```

COMMUNICATION ADAPTATION

- Queuing Requests which suppose to be sent to the REST server
- Library: com.birbit:android-priority-jobqueue

```
public static void StoreInWebServerDatabaseTask(String...memo) {

    App.getInstance().getJobManager().addJobInBackground(new StoreWeb(memo));
}
```

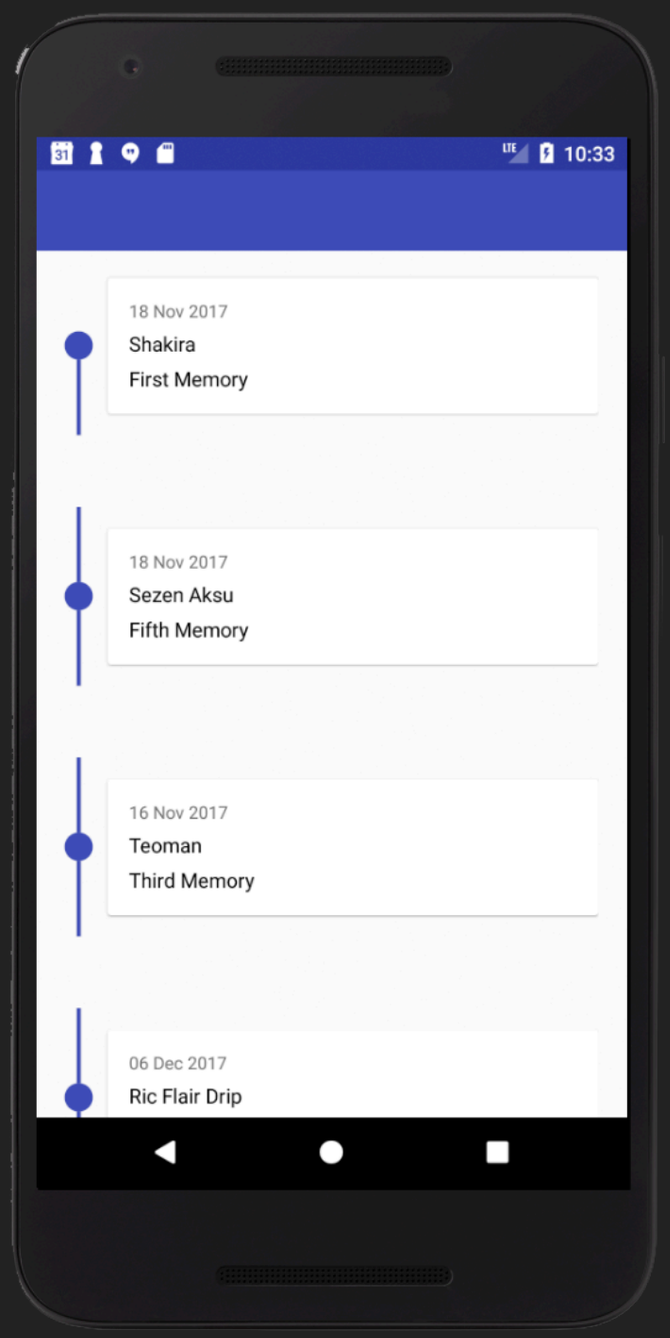
Persist: syncs user's content to server

RequireNetwork

CONNECTIVITY CHALLENGE

- In order to provide a good user experience :
- Converting the App to Memory Booklet based on connectivity status

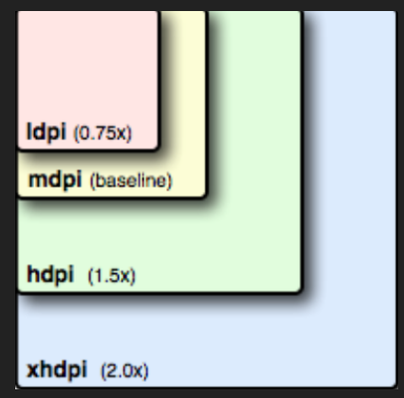
```
private boolean haveNetworkConnection() {
    boolean haveConnectedWifi = false;
    boolean haveConnectedMobile = false;

    ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo netInfo = cm.getActiveNetworkInfo();

        if (netInfo.getTypeName().equalsIgnoreCase( anotherString: "WIFI"))
            if (netInfo.isConnected())
                haveConnectedWifi = true;
        if (netInfo.getTypeName().equalsIgnoreCase( anotherString: "MOBILE"))
            if (netInfo.isConnected())
                haveConnectedMobile = true;

    return haveConnectedWifi || haveConnectedMobile;
}
```



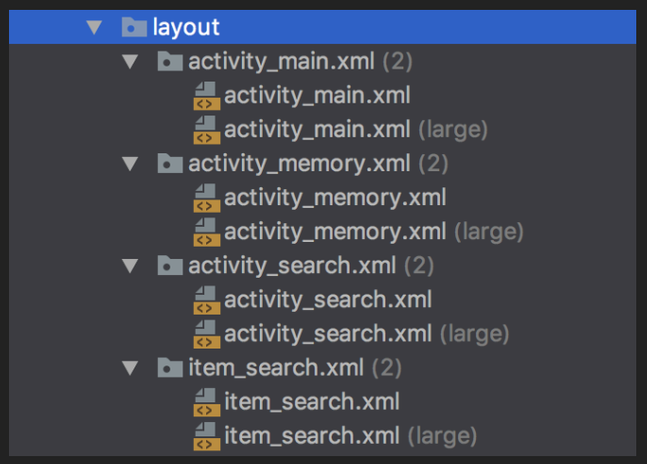
Skipping the **Search** and **Memory Add Activities**

USABILITY CHALLENGE ▸ Supporting Different Screen Sizes

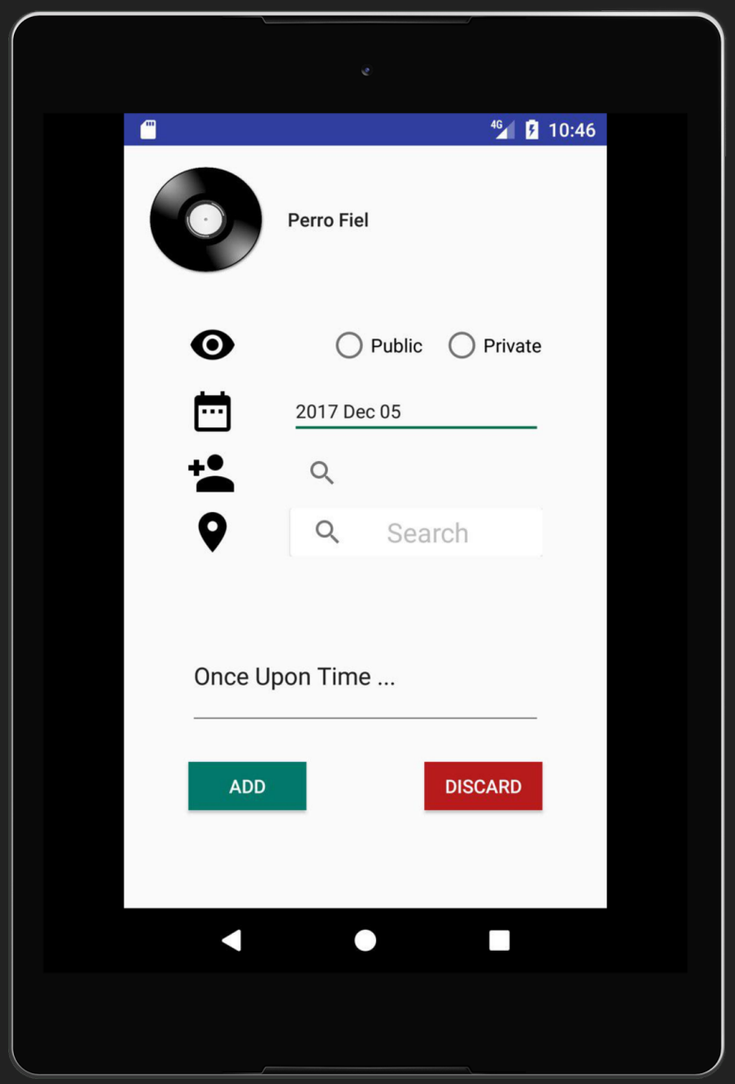
- Using **RelativeLayout** instead of AbsoluteLayout
- Generating density-specific **Resources** (mipmap-drawable)

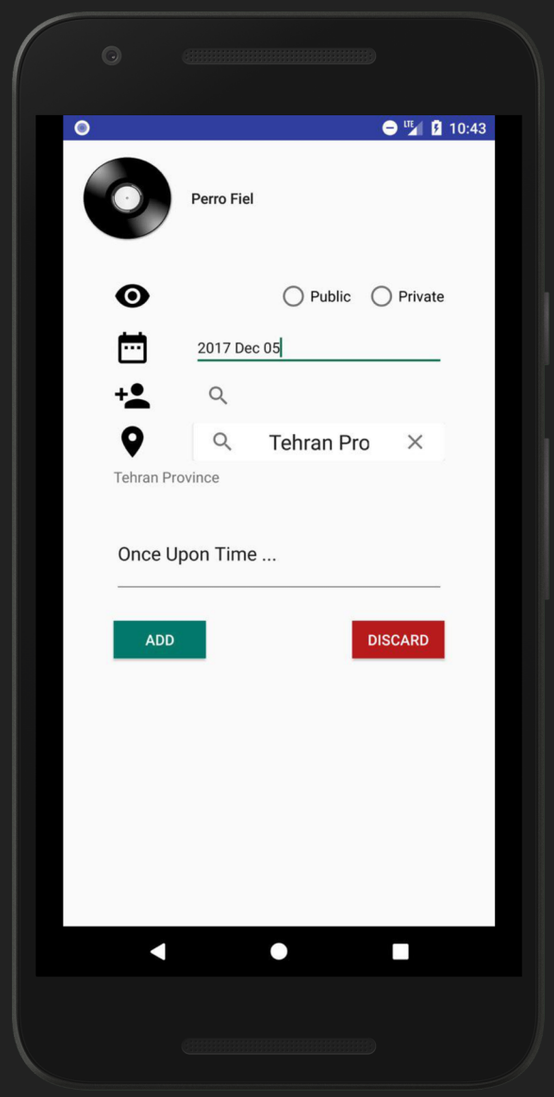


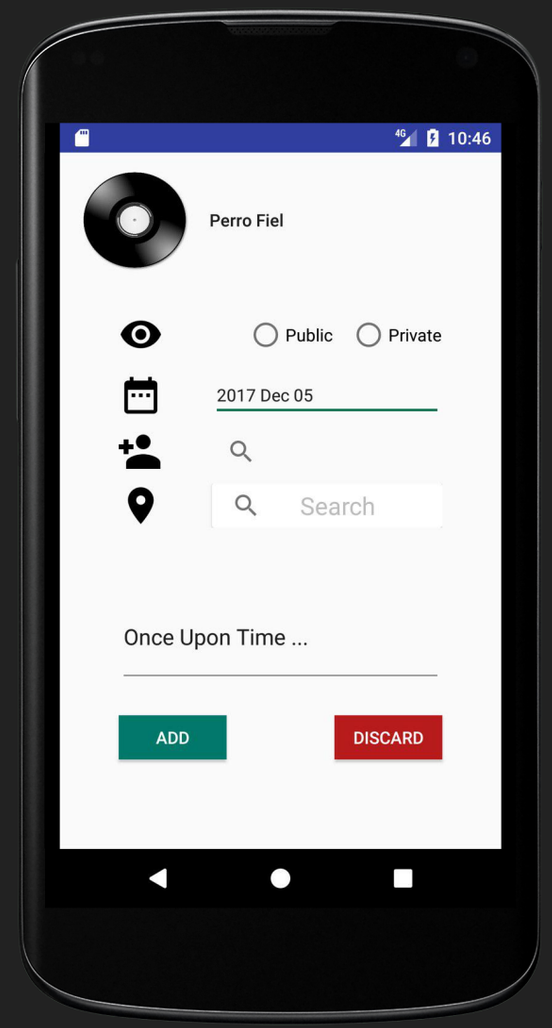
- Avoiding **hard-coded sizes**
- Using **Size Qualifiers**
- ➡ Creating Different layouts for large screens



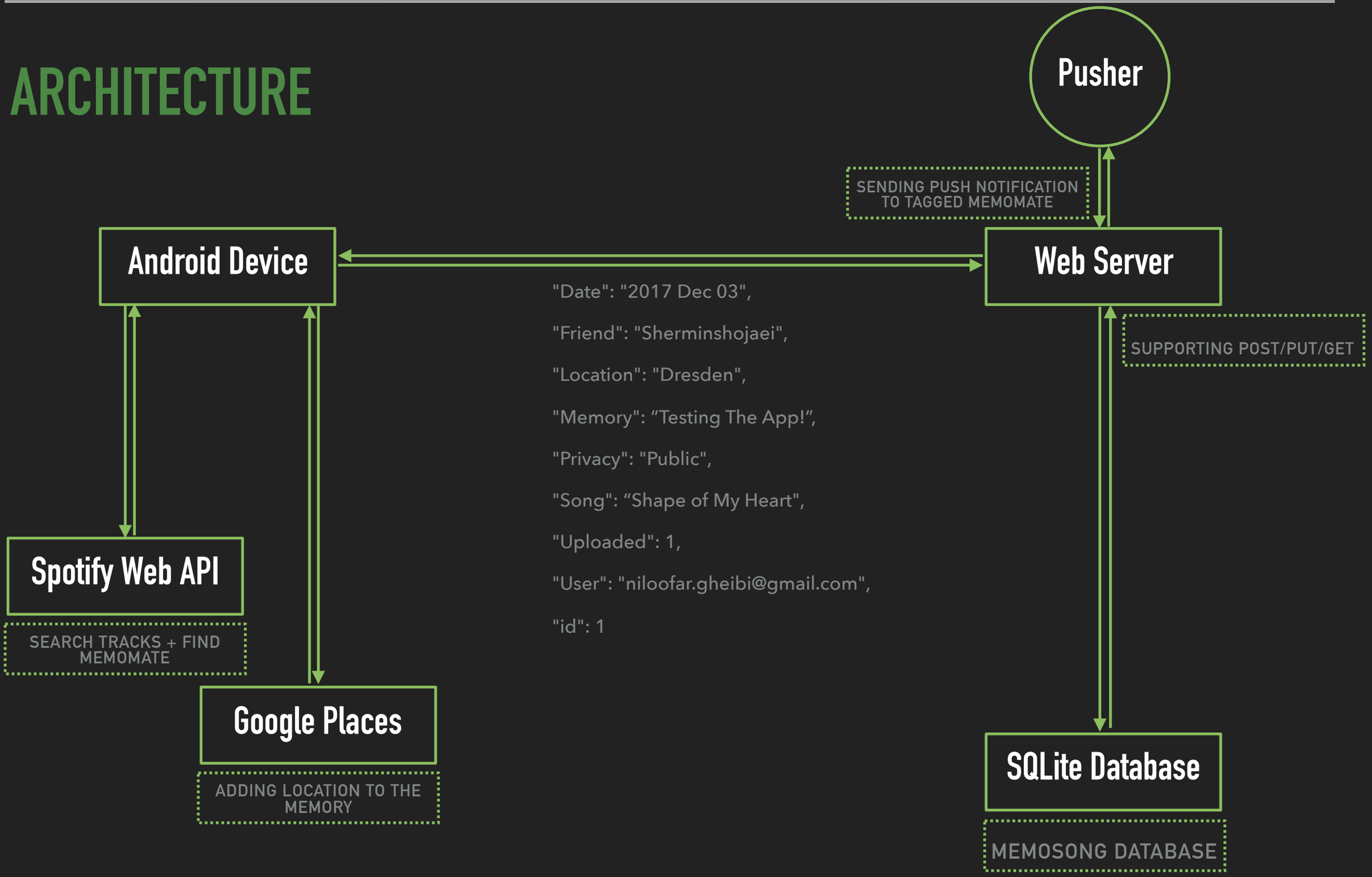
USABILITY CHALLENGE (CONT.)







ARCHITECTURE



WHAT TO DO NEXT?

- Close open topics in network/offline challenge
- Implementation of push notification from server side
- Enhance usability of Memory Booklet (Timeline)
- Enhancing UI
- Test on real devices

QUESTIONS?