APPLICATION DEVELOPMENT FOR MOBILE AND UBIQUITOUS COMPUTING

AGRIPLANNER

SEMINAR PRESENTATION – 3

Group No. – 3
Tomasz Krol , Nikhil Ambardar
01.02.2019

CONTENTS

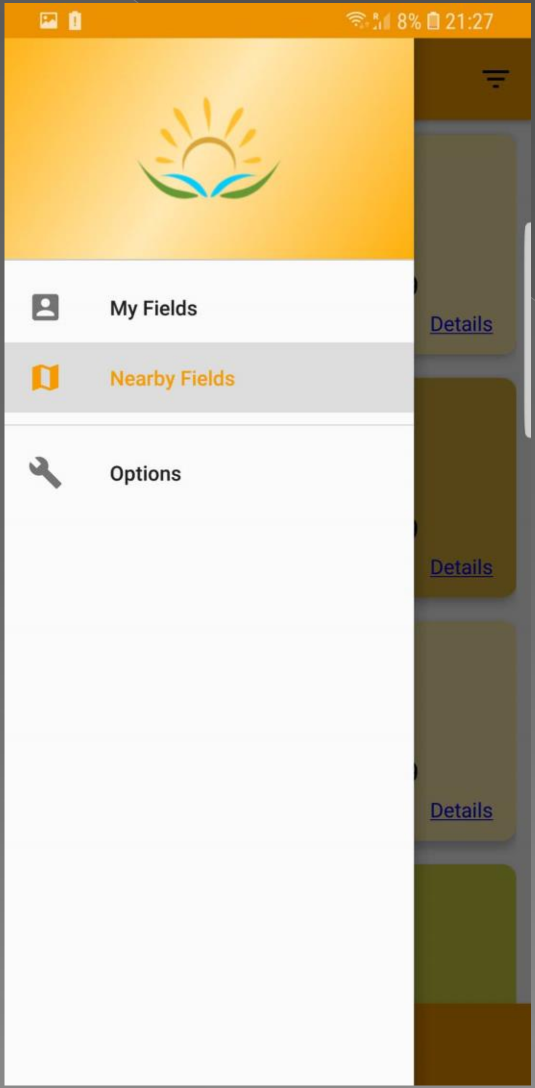
- Application scenario and screenshots of running application
- Final architecture, components and interaction between components
- Technologies for each component
- The challenges of mobile computing we have tackled

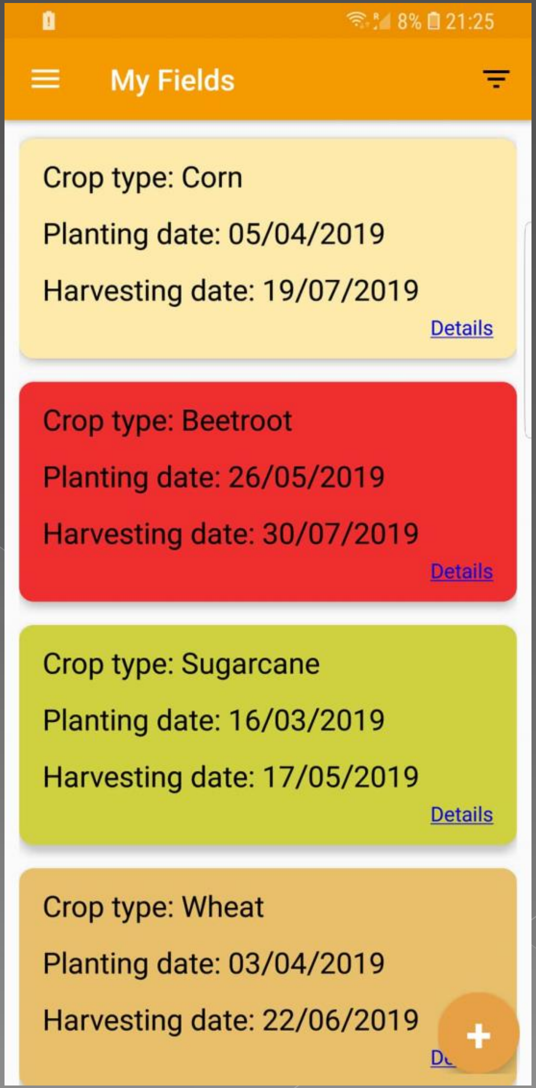
Adaptation mechanisms used

Context information involved

- Open issues and lessons learned

Screenshots of App





Crop type: Corn

Planting date: 05/04/2019

Harvesting date: 19/07/2019

Details

Crop type: Beetroot

Planting date: 26/05/2019

Harvesting date: 30/07/2019

Details

Crop type: Sugarcane

Planting date: 16/03/2019

Harvesting date: 17/05/2019

Details

Crop type: Wheat

Planting date: 03/04/2019

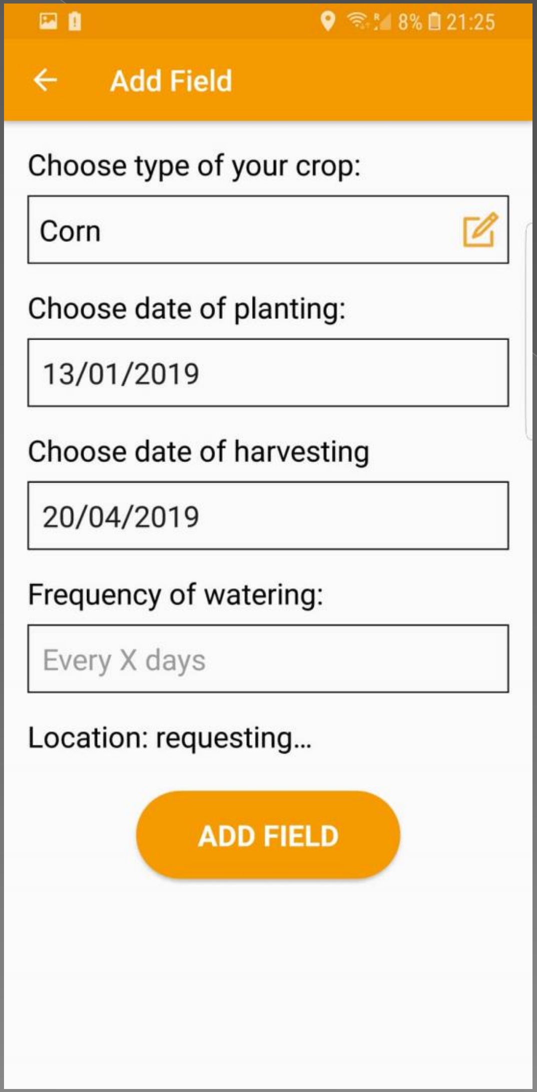
Harvesting date: 22/06/2019

De

Components And Technologies

- Menu – NavigationDrawerMenu (MyActivity)
- List of Users Fields – MyFieldsAdapter with RecyclerView (MyFieldsFragment)
- List of Fields – saved with SQLite Database

ADD FIELD/EDIT FIELD



Add Field

Choose type of your crop:

Corn

Choose date of planting:

13/01/2019

Choose date of harvesting

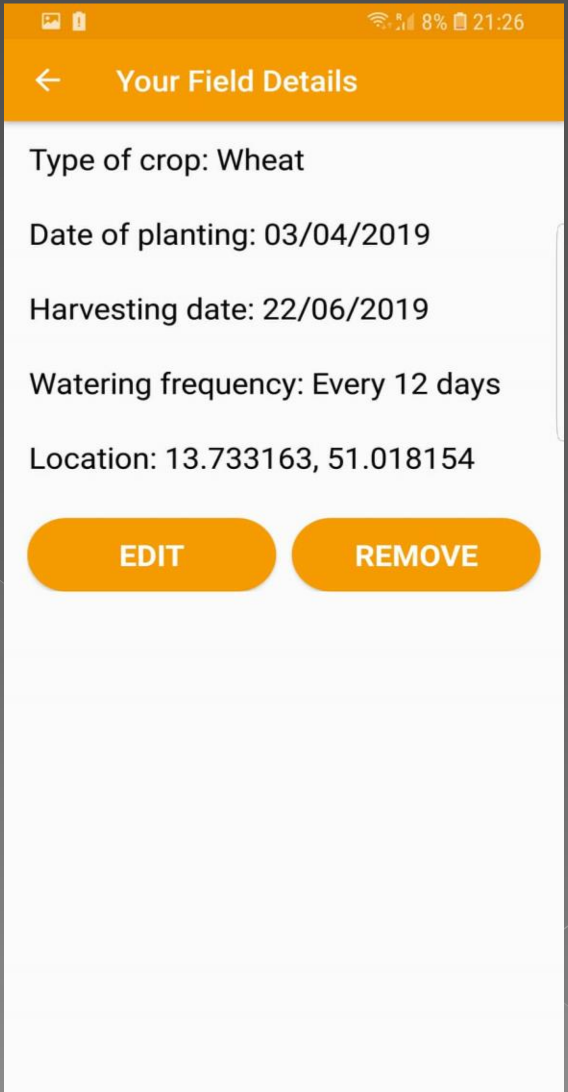
20/04/2019

Frequency of watering:

Every X days

Location: requesting...

ADD FIELD



Your Field Details

Type of crop: Wheat

Date of planting: 03/04/2019

Harvesting date: 22/06/2019

Watering frequency: Every 12 days

Location: 13.733163, 51.018154

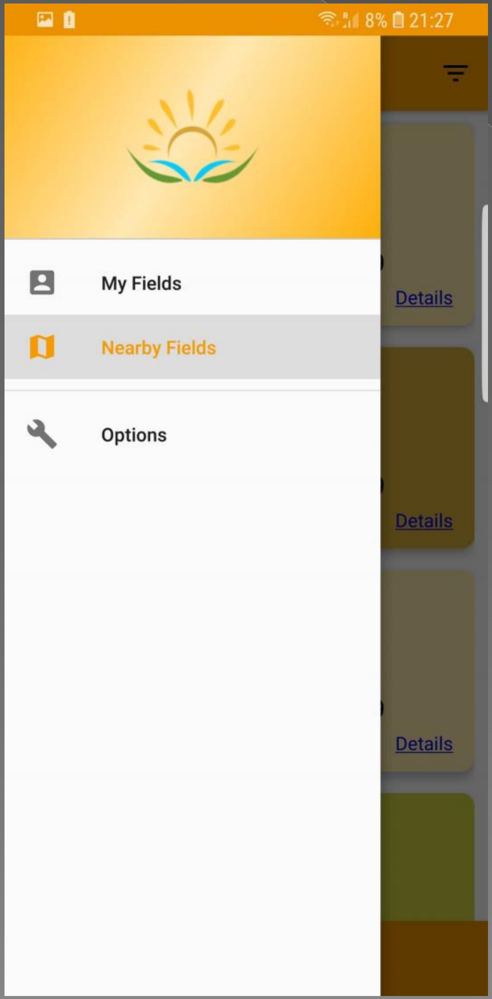
EDIT

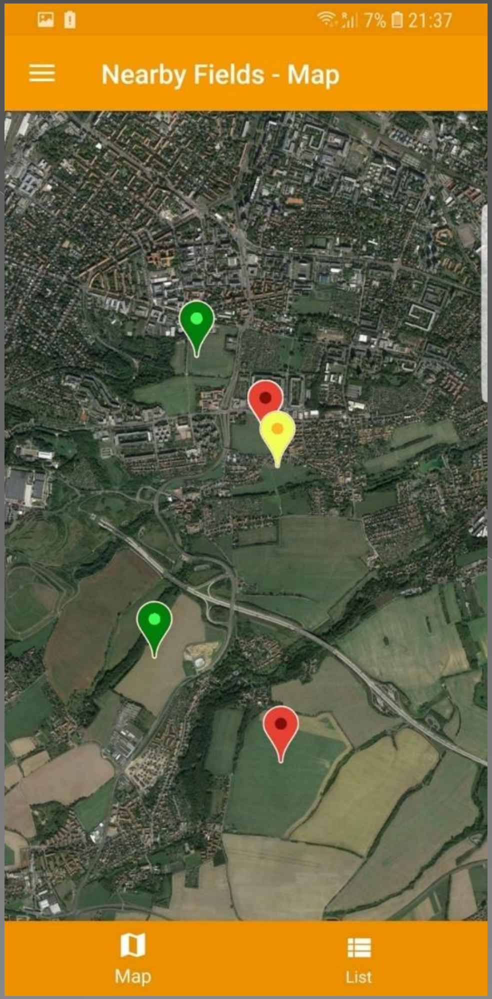
REMOVE

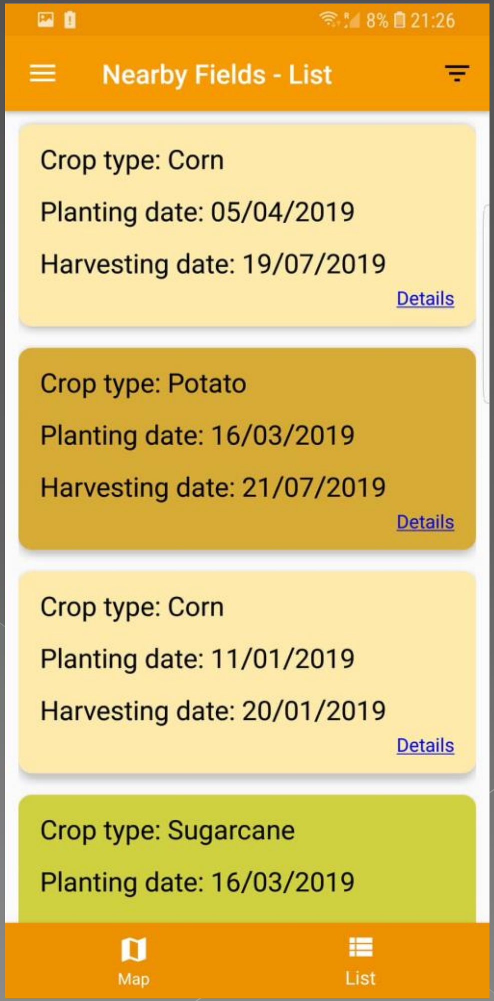
Components And Technologies

- Add Field – Spinner with crop types, DatePickerDialogs for choosing dates, Location from FusedLocationProvider/LocationService (AddFieldActivity)
- Field Details – OnClickListener in MyFieldsAdapter
- Options: Add/Edit/Remove Field – Requests to Firebase Storage

NEARBY FIELDS







Crop type: Corn

Planting date: 05/04/2019

Harvesting date: 19/07/2019

Details

Crop type: Potato

Planting date: 16/03/2019

Harvesting date: 21/07/2019

Details

Crop type: Corn

Planting date: 11/01/2019

Harvesting date: 20/01/2019

Details

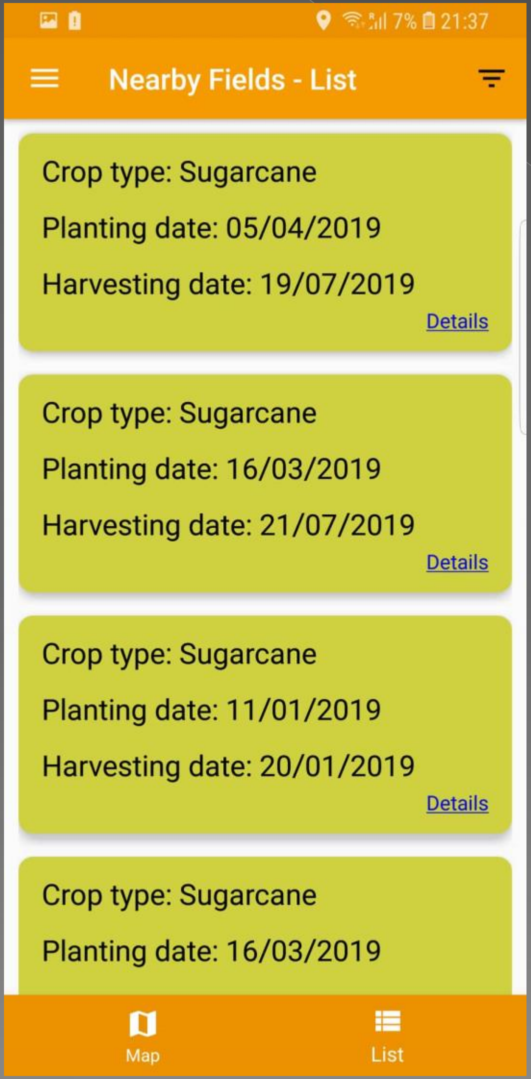
Crop type: Sugarcane

Planting date: 16/03/2019

Components And Technologies

- Nearby Fields Map – GoogleMap/MapView with User`s location in the center of the map (location from FusedLocationProvider/LocationService), markers/pins of nearby fields from Firebase(online)/SQLite(offline)
- Nearby Fields List – RecyclerView with NearbyFieldsAdapter
- BottomNavigationMenu

FILTERED FIELDS



Crop type: Sugarcane

Planting date: 05/04/2019

Harvesting date: 19/07/2019

Details

Crop type: Sugarcane

Planting date: 16/03/2019

Harvesting date: 21/07/2019

Details

Crop type: Sugarcane

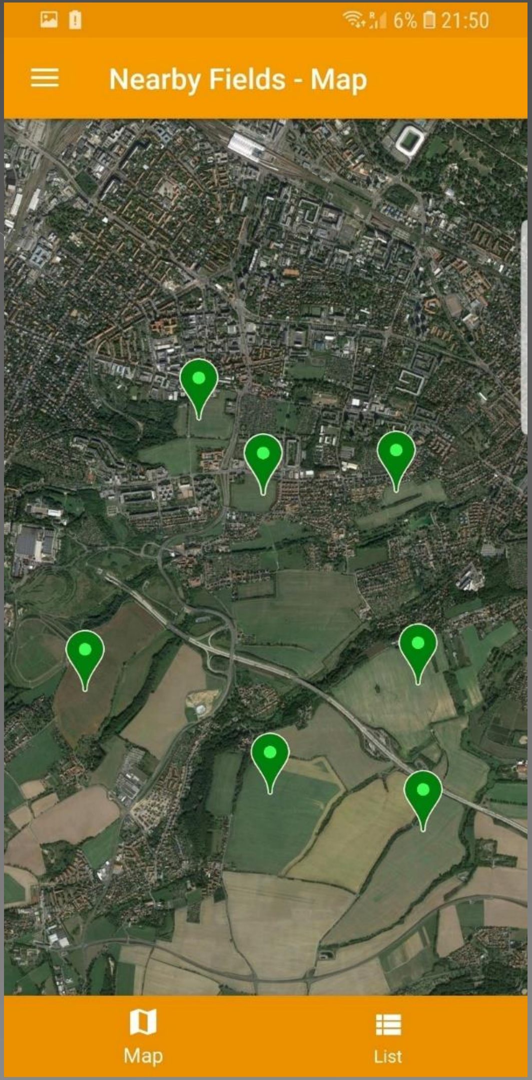
Planting date: 11/01/2019

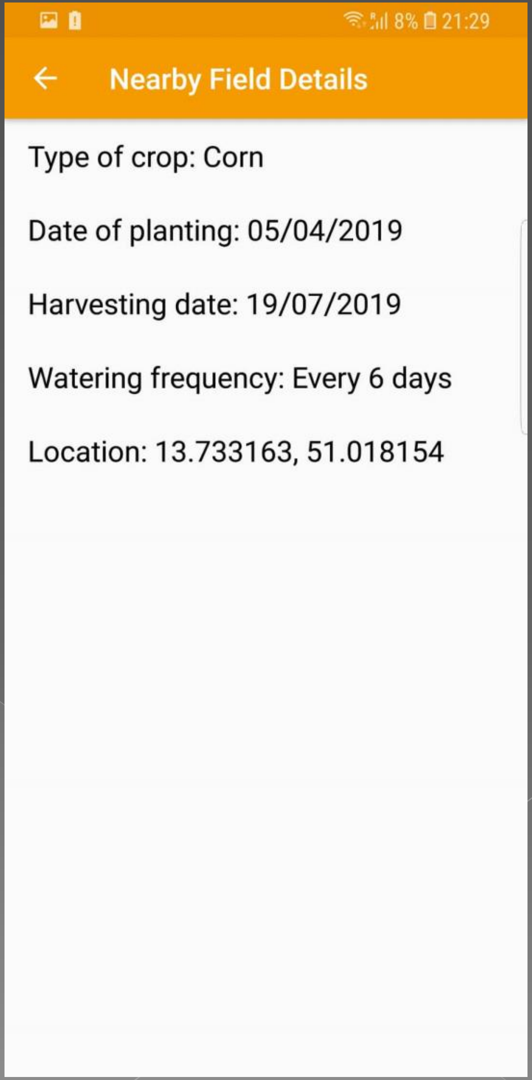
Harvesting date: 20/01/2019

Details

Crop type: Sugarcane

Planting date: 16/03/2019





Type of crop: Corn

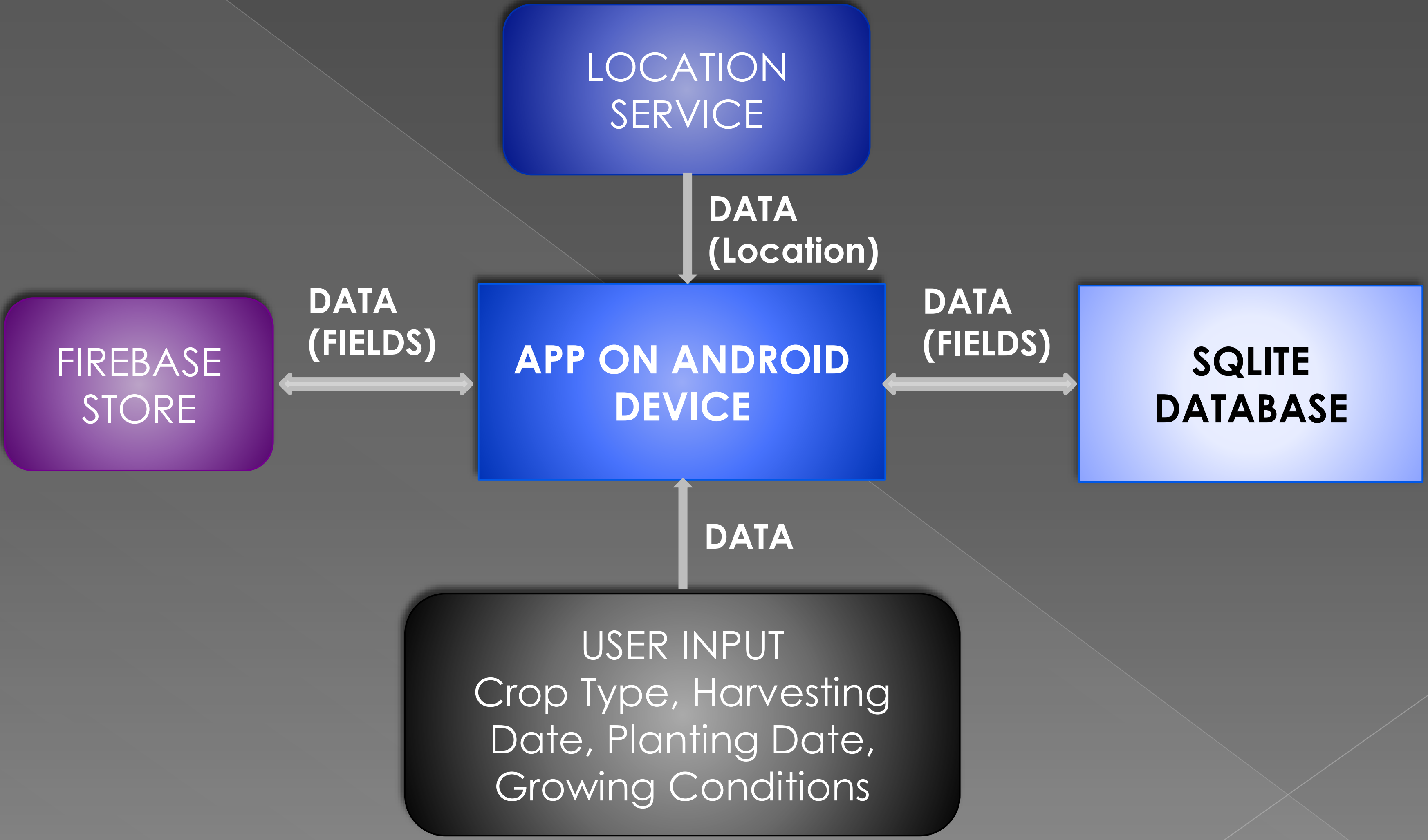
Date of planting: 05/04/2019

Harvesting date: 19/07/2019

Watering frequency: Every 6 days

Location: 13.733163, 51.018154

ARCHITECTURE



LOCATION SERVICE

DATA (Location)

FIREBASE STORE

DATA (FIELDS)

APP ON ANDROID DEVICE

DATA (FIELDS)

SQLITE DATABASE

DATA

USER INPUT
Crop Type, Harvesting Date, Planting Date, Growing Conditions

Location

Context:

- Obtain the current location using `android.gms.location.FusedLocationProviderClient`

```
if(canRequestLocation && hasLocationPermission) {
        ➢ Show nearby fields on the map arround the user in given radius
        ➢ Save user`s location to cache
}
else {
        ➢ Load location from cache or use default location set by user
}
```

Battery

Context:

- Determine the current battery level,using **BatteryManager.EXTRA_LEVEL** and **BatteryManager.EXTRA_SCALE**

```
if(batteryLevel <= 25%) {
        ➢ reduce the rate of background updates to reduce battery
          consumption through asking user if he wants to load data from locally
          database instead of loading data from cloud
        ➢ use last saved location without using GPS to reduce battery consumption
}
else {
        ➢ Load data from Firestore
}
```

Network

Context:

- Detect if app is offline, using **Android.net.ConnectivityManager** and **android.net.NetworkInfo**

```
if(isOnline) {
        ➢ Get fields data from cloud store
        ➢ Save this data to locally SQLite database
}
else {
        ➢ Inform user that he is in offline mode and data can be out of date
        ➢ Use pre-fetched data from locally SQLite database, not from Firebase
          caching
}
```

EXTRA CONTEXT FEATURES/IDEAS

- Technical Context – Weather Data from Open Weather Map API - min/ max temperature, humidity, rain, clouds
- Contextual Info -Info by buyers – price of crops , Info by sellers – crop type , quantity sowed, date of sowing
- Personal Context – App user id
- Social Context – Nearby field users data shared to social networks
- Operational Context – Users as Buyers / Sellers i.e. farmers

Open Issues and Lessons Learnt

- Lessons Learned

  Using Android Studio

  Implementing Cloud How to use Google Firebase

  Implementing intuitive UI and seamless data from DB
- For the future…

  User experience – improving user interface - Better implementation of features

  better case handling from backend , Adding social networking dimension

QUESTIONS ?

OR ELSE

THANK YOU FOR ATTENTION