

LAPORAN TUGAS AKHIR
APLIKASI MONITORING KEBUN HIDROPONIK
BERBASIS MOBILE



Disusun oleh :
Fernanda Daymara Hasna

MOBILE APPLICATION AND TECHNOLOGY
KEJURUAN TEKNOLOGI INFORMASI DAN KOMUNIKASI
BALAI BESAR PENGEMBANGAN LATIHAN KERJA
BBPLK BEKASI
2021

KATA PENGANTAR

Assalamualaikum Wr. Wb.

Puji syukur kehadirat Tuhan Yang Maha Esa atas segala rahmat-Nya, penulis dapat menyelesaikan laporan tugas akhir dengan judul **Laporan Tugas Akhir Aplikasi Monitoring Kebun Hidroponik Berbasis Mobile**. Pengembangan ini disusun dalam rangka memenuhi salah satu persyaratan untuk menyelesaikan Pelatihan Berbasis Kompetensi (PBK) Tahap 1 “*Mobile Application and Technology*” yang diselenggarakan oleh Balai Besar Pengembangan Latihan Kerja (BBPLK) Bekasi sejak Februari 2021 s.d. Oktober 2021.

Penulis menyadari bahwa dalam penyusunan proposal ini jauh dari kata sempurna, untuk itu penulis memohon kritik dan saran yang membangun sehingga dapat menjadi lebih baik lagi. Harapannya penelitian ini dapat berguna sebagai acuan pengembangan selanjutnya dan bermanfaat bagi kita semua. Aamiin.

Wassalamualaikum Wr. Wb.

Bekasi, 05 Oktober 2021
Penulis

Fernanda Daymara Hasna

DAFTAR ISI

KATA PENGANTAR	ii
DAFTAR ISI.....	iii
DAFTAR GAMBAR	v
DAFTAR TABEL.....	v
DAFTAR KODE.....	v
BAB I : PENDAHULUAN	7
A. Latar Belakang.....	7
B. Permasalahan	7
C. Batasan Masalah	8
D. Tujuan dan Manfaat.....	8
BAB II : LANDASAN TEORI	9
A. Android.....	9
1. Arsitektur MVVM	10
2. <i>View Binding</i>	11
B. Firebase.....	12
1. Firebase Authentication.....	12
2. Firebase Cloud Firestore	13
3. Firebase Storage.....	14
BAB III : METODE PENGEMBANGAN.....	15
A. Metode Pengembangan.....	15
1. <i>Initiation (Requirements)</i>	15
2. <i>Planning (Design)</i>	16
3. <i>Executing (Development, Testing, Review & Feedback)</i>	16
4. <i>Closure (Deployment & Release)</i>	16
B. Rencana Jadwal Pengembangan.....	17

C. Sumber Daya	18
1. Perlengkapan Perangkat Keras	18
2. Perlengkapan Perangkat Lunak	19
BAB IV : HASIL DAN PEMBAHASAN.....	20
A. <i>Initiation (Requirements)</i>	20
B. <i>Planning (Design)</i>	20
1. Desain <i>Storyboard</i>	20
2. Desain <i>Front End</i>	21
3. Desain <i>Back End</i>	24
C. <i>Executing (Development, Testing, Review & Feedback)</i>	29
1. Profil Aplikasi	29
2. Fitur yang Dikembangkan	30
3. Implementasi Aplikasi	31
D. <i>Closure (Deployment & Release)</i>	39
BAB V : PENUTUP	40
A. Kesimpulan	40
B. Saran.....	40
DAFTAR PUSTAKA.....	41
LAMPIRAN	43

DAFTAR GAMBAR

Gambar 1. Ilustrasi arsitektur MVVM	10
Gambar 2. Perbandingan cara mengakses komponen pada <i>View Layer</i>	11
Gambar 3. Cara penggunaan <i>View Binding</i>	11
Gambar 4. Perbandingan struktur data Real Time Database dan Cloud Firestore.....	13
Gambar 5. Ilustrasi metode pengembangan <i>Hybrid</i>	15
Gambar 6. Struktur menu aplikasi.....	21
Gambar 7. <i>Mock up</i> aplikasi	22
Gambar 8. <i>Wireframe</i> aplikasi yang diterapkan pada literatur	24
Gambar 9. <i>Entity Relationship Diagram</i> (ERD) aplikasi.....	25
Gambar 10. Struktur database <i>document collection</i> dan relasinya	26
Gambar 11. Detail struktur database beserta atribut datanya	27
Gambar 12. <i>Activity Diagram</i> dari aplikasi	28
Gambar 13. Logo aplikasi Nydrobionics.....	30
Gambar 14. Hasil dari render untuk layout <i>CreateUserFragment</i> dan <i>EditProfileUserActivity</i>	37

DAFTAR TABEL

Tabel 1. Rencana jadwal pengembangan	17
--	----

DAFTAR KODE

Kode 1. Mengaktifkan <i>View Binding</i> pada build.gradle.....	11
Kode 2. <i>Syntax</i> dasar pada Firebase Authentication.....	12
Kode 3. Perbandingan <i>syntax</i> Real Time Database dan Cloud Firestore	13
Kode 4. <i>Syntax</i> dasar untuk upload foto profil pada Firebase Storage	14

Kode 5. Struktur data <i>UserModel</i> beserta konverternya	32
Kode 6. Enum class <i>Gender</i> dan <i>Role</i> untuk memudahkan pengelolaan data terkait	34
Kode 7. LoadingInterface yang berfungsi untuk mengatur aktivasi <i>view</i> saat pengiriman data	34
Kode 8. Penggunaan CreateProfileViewModel untuk membuat dan mengubah data Farm	36
Kode 9. Penggunaan ulang layout <i>CreateUserFragment</i> untuk <i>EditProfileUserActivity</i>	36
Kode 10. Pengiriman, penerimaan, dan pengaturan inisiali awal untuk <i>EditProfileUserActivity</i>	38
Kode 11. Penggunaan <i>IntentUtility</i> untuk membuka <i>maps</i>	39

BAB I

PENDAHULUAN

A. LATAR BELAKANG

Saat ini, bercocok tanam tidak hanya dilakukan dengan media tanam berupa tanah. Banyak media tanam lain yang dapat dijadikan alternatif lain jika tanah tidak memungkinkan untuk digunakan, contohnya adalah hidroponik. Hidroponik merupakan teknik budidaya tanaman yang memanfaatkan air (tanpa tanah) sebagai media tanamnya, dimana akar tanaman terendam dan tumbuh pada air dangkal yang bernutrisi tersirkulasi. Pada hidroponik yang terbuka, lebih rentan terkena hama dan penyakit seperti belalang (membuat daun berlubang), lalat buah (membuat daun memiliki garis meliuk tak beraturan), hama kutu (membuat daun keriting menggulung dan batang melemah), dll. sehingga dibutuhkan mekanisme tambahan penghalau hama penyakit. Sedangkan pada hidroponik tertutup seperti *greenhouse* di daerah perkotaan (biasanya menggunakan paronet), membuat suhu udara di dalamnya menjadi lebih panas dan kadar kelembaban menurun, sehingga membutuhkan mekanisme tambahan untuk menstabilkan keadaan tersebut.

Di beberapa kota di Indonesia juga banyak berkembang perkebunan hidroponik bahkan sudah memiliki cabang kebun di wilayah yang berbeda. Hal ini menimbulkan permasalahan tentang mekanisme pelaporan monitoring keadaan hidroponik, khususnya bagi petani pada skala industri, dikarenakan kebun yang harus dipantau semakin banyak dan beragam. Sistem pemantauan secara konvensional harus menunggu rekap dari petani, dimana petani harus menyelesaikan *jobdesk* terlebih dahulu yang menyebabkan kondisi yang dilaporkan tidak *real time*. Terutama pada situasi COVID-19 yang menyulitkan proses mobilisasi karyawan yang terlibat. Hal ini dapat mempengaruhi tingkat akurasi data keadaan kebun dan tingkat efisiensi kinerja karyawan.

B. PERMASALAHAN

Belum adanya mekanisme pelaporan keadaan kebun hidroponik dengan memanfaatkan teknologi digital menyebabkan kurang efektifnya pelaporan secara konvensional untuk tetap diterapkan pada pertanian hidroponik skala industri. Dengan

sistem ini, diharapkan dapat mempermudah *user* dalam memberikan laporan keadaan dan memantau kondisi lingkungan kebun hidroponik yang menunjang kualitas tanaman dan hasil panennya.

C. BATASAN MASALAH

Untuk memfokuskan permasalahan yang diangkat, maka batasan masalah tersebut diantaranya adalah :

- a) Aplikasi *mobile* yang dikembangkan untuk Android.
- b) Setiap akun hanya dapat memiliki 1 jenis role yang tidak dapat diubah, yaitu *Owner* (pemilik)
- c) Setiap akun hanya terintegrasi dengan 1 kebun. *Owner* hanya dapat memiliki 1 kebun.
- d) Setiap kebun dapat memiliki beberapa kit yang dapat dibuat oleh setiap anggota kebun.
- e) Data monitoring kebun dapat diinput manual melalui aplikasi yang dirancang.
- f) Pengelolaan tanaman yang ditanam hanya meliputi penambahan tanaman.
- g) Pengembangan fitur lainnya dapat dilakukan selama tidak melebihi waktu pengembangan yang disediakan.

D. TUJUAN DAN MANFAAT

Tujuan dari aplikasi yang dikembangkan ini adalah untuk meningkatkan kinerja petani hidroponik dengan memudahkan pelaporan hasil monitoring berbasis aplikasi sehingga data dapat diakses secara *real time* oleh petani maupun *stakeholder* kebun di berbagai lokasi serta dapat menunjang kualitas tanaman, Diharapkan aplikasi ini dapat memberikan manfaat berupa :

- a) Membantu petani hidroponik dalam memudahkan system database dan pelaporan hasil monitoring.
- b) Menjadi ilmu dan media informasi pembelajaran mengenai pengembangan aplikasi *mobile*.
- c) Memberikan referensi mengenai pengembangan fitur yang lebih baik untuk aplikasi sejenis.

BAB II

LANDASAN TEORI

A. ANDROID

Android adalah sebuah *operating system* untuk *mobile device* berbasis Linux. Android merupakan salah satu OS yang bersifat *open source*, sehingga banyak pihak yang bereksplorasi dengan OS ini. Sebagai contohnya yaitu *mobile devices* yang menggunakan Android sebagai OS nya, seperti *smartphones* dan *tablet*, dapat memiliki *graphical user interface* (GUI) yang berbeda meskipun menerapkan OS yang sama karena kostumisasi yang dilakukan setiap developernya. Tidak hanya GUI-nya, beberapa *mobile devices* juga memiliki *built-in application* dan juga mendukung *third-party programs* lainnya [1]. Hal ini dimanfaatkan sebagai ciri khas dari brand *mobile devices* tersebut.

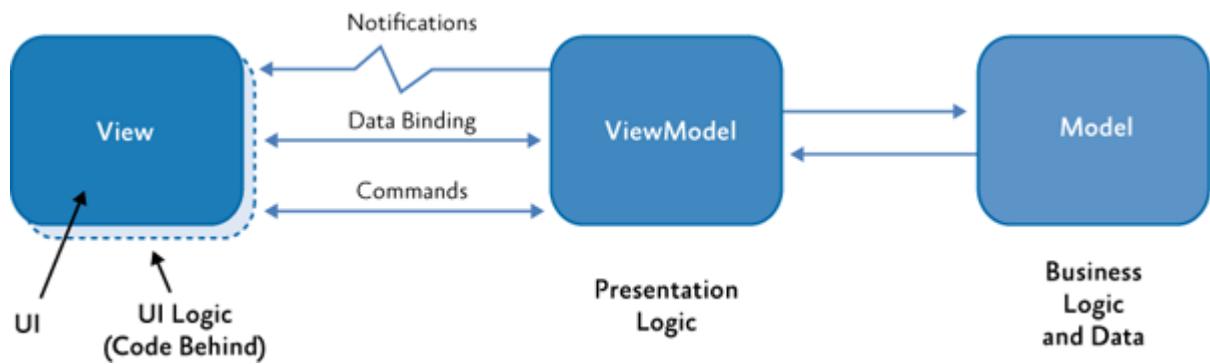
Dikenal sebagai produk yang dikembangkan oleh Google (GOOGL), namun pengembangan awalnya diinisiasi oleh Android Inc. pada 2003 yaitu sebuah perusahaan software yang berbasis di Silicon Valley sebelum diakuisisi oleh Google pada 2005 [2]. Android dirilis pertama kali ke publik pada Oktober 2008 dengan versi Android 1.0, dimana *codename* yang umum dikenal sekarang belum diterapkan pada versi 1.0 dan 1.1. Semenjak versi 1.5 hingga saat ini, setiap versi memiliki *codename* yang unik, untuk versi 1.5 hingga 9 memiliki *codename* dengan tema makanan sedangkan versi sejak 10 hingga saat ini, versi terbaru Android 12 yang dirilis pada 4 Oktober 2021, menerapkan urutan angka sebagai *codename*-nya [3].

Developer dapat membuat aplikasi untuk Android menggunakan Android *support developer kit* (SDK) yang berbasis Java, sehingga mendukung bahasa pemrograman turunan dari Java seperti Kotlin [2]. Biasanya SDK sudah terintegrasi dengan *integrated development environment* (IDE) yang mendukung SDK tersebut agar memudahkan developer dalam mengembangkan aplikasi tersebut. IDE untuk Android yang dikembangkan oleh Google adalah Android Studio yang juga mendukung *build system* yang lebih fleksibel, sehingga memungkinkan untuk membuat banyak variasi aplikasi untuk *mobile devices* yang berbeda dalam satu proyek. Dengan kata lain, Android SDK adalah “otak” untuk membuat aplikasi dan Android Studio adalah *tools* yang digunakan

untuk membuat aplikasi tersebut, sehingga sangat memungkinkan untuk membuat aplikasi dengan *tools* lainnya [4].

Dalam membangun aplikasi Android, menerapkan *architectural pattern* Android akan memudahkan dalam proses pengembangan karena *block codes* yang dibangun menjadi lebih terstruktur dan terorganisir. Beberapa arsitektur yang banyak digunakan adalah *Model-View-Controller* (MVC), *Model-View-Presenter* (MVP), dan *Model-View-ViewModel* (MVVM).

1. ARSITEKTUR MVVM



Gambar 1. Ilustrasi arsitektur MVVM

Sumber : <https://docs.microsoft.com/>

Arsitektur MVVM (*Model-View-ViewModel*) merupakan arsitektur yang direkomendasikan oleh Android, dimana memisahkan antara *business logic* dengan GUI. Arsitektur MVVM dibagi menjadi 3 layer, yaitu *Model Layer* berupa representasi dari data yang digunakan seperti *Plain Old Java Object* (POJO), *Data Class*, dan lainnya, *View Layer* berupa representasi dari *user interface* pada aplikasi seperti *Activity* dan *Fragment*, dan *ViewModel Layer* yang berinteraksi dengan Model dan menyalurkan data ke *View Layer* [5]. Dengan menerapkan arsitektur ini, data akan tetap konsisten meskipun terjadi konfigurasi pada GUI seperti perubahan orientasi *mobile device* saat terjadi *rotate* dari *portrait* ke *landscape* maupun sebaliknya.

Untuk berinteraksi dengan komponen yang ada pada Layout XML yang berada pada *View Layer*, banyak cara yang dapat dilakukan seperti `findViewById`, `DataBinding`, `Butterknife`, dan `Kotlin Syntethic`. Namun berdasarkan Gambar 2, masing-masing metode yang telah disebutkan di atas memiliki kekurangan yang dijawab melalui metode terakhir. Metode terakhir dan terbaru yang dikembangkan oleh Android ini bernama *View Binding* yang dirilis pada Februari 2020 [6].

How To Access Views?			
Method	Elegance	Compile Time Safety	Build Speed Impact
findViewById	✗	✗	✓
DataBinding	✓	✓	✗
Butterknife	✓	✗	✗
Kotlin Synthetic	✓	✗	✓
???	✓	✓	✓

Gambar 2. Perbandingan cara mengakses komponen pada *View Layer*

Sumber : <https://www.youtube.com/>

2. VIEW BINDING



```
// With ViewBinding
val binding = MyBinding.inflate(...)

binding.tvTitle.text = "Less boilerplate"
binding.tvDescription.text = "Make your code clean"
binding.tvAuthor.text = "Mael"
```

Gambar 3. Cara penggunaan *View Binding*

Sumber : <https://medium.com/>

View Binding adalah metode untuk mengakses komponen pada Layout XML dengan cara memanggil id dari komponen yang berkaitan dari *binding layout* yang digunakan seperti pada Gambar 3. Hal ini bisa dilakukan karena setiap kali ada perubahan pada Layout XML, *View Binding* akan otomatis menyimpan komponen tersebut dalam sebuah variable sesuai dengan id yang didefinisikan pada Layout XML dan langsung dapat digunakan pada layout yang bersesuaian. Untuk menggunakan *View Binding*, pastikan *minimum requirement* berupa Android Studio 3.6 dan Android Gradle Plugin 3.6 terpenuhi. Kemudian dapat mengonfigurasi pada build.gradle dengan menambahkan pada build.gradle [7].

```
android{
    ...
    buildFeatures{
        viewBinding true
    }
}
```

Kode 1. Mengaktifkan *View Binding* pada build.gradle

B. FIREBASE

Firebase adalah salah satu layanan dari Google yang menyediakan beragam fitur untuk dapat digunakan pada produk digital, seperti aplikasi atau *website*, dan dapat menerapkan banyak fitur dalam satu proyek sekaligus. Hal ini mempercepat pekerjaan developer dikarenakan mengurangi beban kerja terhadap bagian *back end* sehingga bisa focus pada pengembangan produk digital itu sendiri. Layanan pertama Firebase adalah Firebase Real Time Database pada 2011 (oleh Andrew Lee dan James Tamplin) sebelum diakusisi oleh Google pada 2014 dan berkembang sebagai layanan yang menyediakan beragam fitur. Firebase pun dapat digunakan secara gratis (*billing plan* Spark) maupun berbayar sesuai dengan pemakaiannya (*billing plan* Blaze) [8].

Karena Firebase dan Android merupakan produk yang sama-sama dikembangkan oleh Google, maka disediakan pula konfigurasi yang mudah untuk menghubungkan antar kedua layanan tersebut. Dari Android Studio, pengguna dapat langsung menggunakan Firebase melalui menu *Tools* dan memilih layanan Firebase yang akan digunakan. Dari sisi Firebase, pengguna perlu mendaftarkan aplikasi Android pada *Project Settings*. Kedua konfigurasi ini dapat dilakukan sekaligus memilih layanan Firebase, dikarenakan pengguna diharuskan untuk menghubungkan Firebase dan memasukkan *dependency library* yang bersangkutan hanya dengan menekan tombol.

1. FIREBASE AUTHENTICATION

Firebase Authentication adalah layanan Firebase untuk proses autentikasi ke produk digital yang dibuat. Autentikasi, yaitu proses validasi identitas saat memasuki suatu sistem, yang didukung oleh Firebase Authentication di antaranya yaitu dengan email, nomor telepon, secara anonymous, maupun provider lain seperti Google, Facebook, GitHub, Apple, Microsoft, Twitter, Yahoo, dan sebagainya. Firebase Authentication juga terintegrasi dengan layanan Firebase lainnya [8]. Berdasarkan dokumentasi Firebase Authentication untuk Android [9], terdapat beberapa *syntax* yang membangun fungsi-fungsi dasar Firebase Authentication seperti mengecek *auth* terkini (apakah ada yang aktif atau

```
fun firebaseAuth(email:String, password:String){
    val auth : FirebaseAuth = FirebaseAuth.auth

    /** Check current user auth */
    when(auth.currentUser){
        null -> "No currentUser data"
        else -> "currentUser is found"
    }

    /** Sign Up new user**/
    auth.createUserWithEmailAndPassword(email, password)

    /** Sign In with email password **/
    auth.signInWithEmailAndPassword(email, password)

    /** Sign Out current user **/
    auth.signOut()
}
```

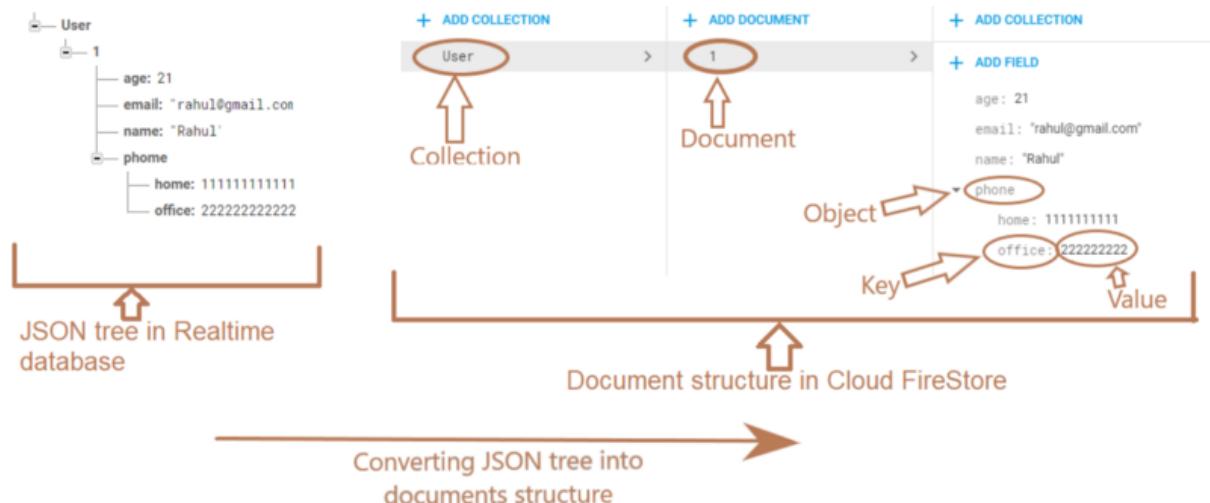
Kode 2. *Syntax* dasar pada Firebase

Authentication

tidak), melakukan *sign in* dan *sign out*, maupun sebagainya yang umum digunakan pada aplikasi *mobile* yang ditunjukkan melalui Kode 2.

2. FIREBASE CLOUD FIRESTORE

Firebase menyediakan 2 jenis layanan database, yaitu Firebase Real Time Database dan Firebase Cloud Firestore. Keduanya dapat langsung tersinkronisasi secara *real time* ke setiap pengguna. Hal ini sangat memudahkan developer karena saat membuat produk digital lintas platform, saat terjadi perubahan data mana semua pengguna akan mendapatkan *update* secara serentak dan otomatis [8].



Gambar 4. Perbandingan struktur data Real Time Database dan Cloud Firestore

Sumber : <https://stackoverflow.com/>

Namun yang membedakan keduanya adalah struktur datanya seperti pada Gambar 4. Firebase Real Time menyimpan data dalam bentuk JSON sedangkan Firebase Firestore disimpan dalam bentuk *collection* berupa kumpulan *document* yang berisi *key-value*. Firebase Cloud Firestore hadir untuk memudahkan pengembangan *mobile application*. Sebagai contoh, kedua database dapat melakukan sort atau filter. Namun, pada Firebase Cloud Firestore juga

```
fun firebaseRealTimeDatabase(id:Int, data:HashMap<String,Any>){
    val db : FirebaseDatabase = FirebaseDatabase.database

    /** Add Data */
    db.getReference( path: "users").child(id.toString()).setValue(data)

    /** */
    db.getReference( path: "users").child(id.toString()).removeValue()

    /** Order Data */
    db.getReference( path: "users").orderByChild( path: "age")

}

fun firebaseCloudFirestore(id:Int, data:HashMap<String,Any>){
    val db : FirebaseFirestore = FirebaseFirestore.getInstance()

    /** Add Data */
    db.collection( collectionPath: "users").document(id.toString()).set(data)

    /** Delete Data */
    db.collection( collectionPath: "users").document(id.toString()).delete()

    /** Order Data */
    db.collection( collectionPath: "users").orderBy( field: "age")

    /** Order with Query */
    db.collection( collectionPath: "users")
        .whereEqualTo( field: "name", value: "%fer%")
        .orderBy( field: "age")
}
}
```

Kode 3. Perbandingan *syntax* Real Time Database
dan Cloud Firestore

dapat menambahkan *query* [10]. Perbandingan *syntax* antara kedua layana Firebase Database dijabarkan melalui Kode 3.

3. FIREBASE STORAGE

Firebase Storage adalah layanan untuk menyediakan penyimpanan file seperti gambar, dokumen, dan video. Layanan ini terintegrasi dengan Firebase Authentication, sehingga dapat digunakan untuk media penyimpanan berkas *client-based*. Setelah mengunggah berkas, maka akan mendapatkan url berkas sehingga dilihat dan dimodifikasi oleh *client* (dibutuhkan mekanisme validasi antara pengunggah dan *client* yang mengakses berkas tersebut). Berdasarkan dokumentasi Firebase Storage untuk Android [11], terdapat beberapa *syntax* dasar yang membangun fungsi-fungsi dasar Firebase Storage seperti untuk *upload* dan *download* file. Sebagai contoh pada Kode 4 menunjukkan cara untuk mengunggah foto untuk *user profile*.

```
fun firebaseStorage(
    reference : String = "profile_images",
    fileUri:Uri){
    val storage : FirebaseStorage = Firebase.storage

    /** Define storage reference (location of file) */
    val storageReference : StorageReference =
        storage.getReference(reference)

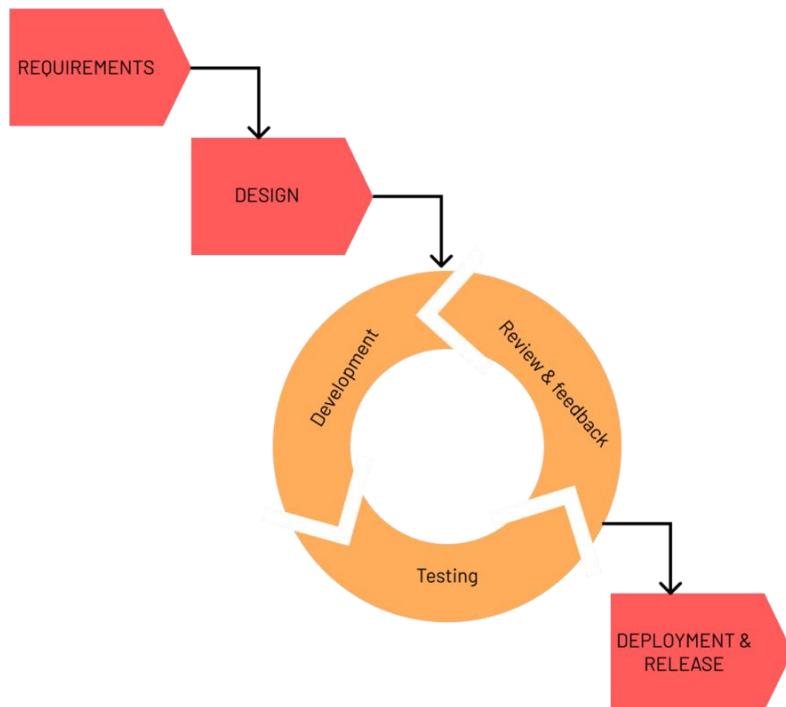
    /** Upload file */
    storageReference.putFile(fileUri)
        .continueWithTask { it: Task<UploadTask.TaskSnapshot!>
            storageReference.downloadUrl
        }
}
```

Kode 4. *Syntax* dasar untuk upload foto profil pada Firebase Storage

BAB III

METODE PENGEMBANGAN

A. METODE PENGEMBANGAN



Gambar 5. Ilustrasi metode pengembangan *Hybrid*

Sumber : <https://cleancommit.io/>

Dalam menjalankan kegiatan, pengembangan aplikasi ini menerapkan *Project Management Method* berupa *Hybrid* atau kombinasi metodologi *Waterfall* dan *Agile* dikarenakan proyek yang dilaksanakan sudah terstruktur namun membutuhkan fleksibilitas dalam pelaksanannya. Fitur atau layanan produk yang ingin dihasilkan dapat mengalami perubahan, dimana terbuka dengan eksperimen dan penambahan selama sesuai dengan tujuan yang dibutuhkan. Setiap tahapan memiliki fokus cakupan pekerjaan yang berbeda-beda sesuai dengan tujuan yang dicapai dari setiap tahapan.

1. INITIATION (REQUIREMENTS)

Tahapan ini menjadi fundamental pada pengembangan aplikasi yang mencakup :

- a) Menentukan latar belakang sehingga menjadi topik yang akan dikerjakan dan menyediakan solusi dari permasalahan yang diangkat.
- b) Membatasi kompleksitas proses pengembangan aplikasi sehingga dapat menentukan anggota kelompok dan pembagian jobdesk yang diinginkan.

2. PLANNING (DESIGN)

- a) Menentukan rancangan fitur, layanan, dan kebutuhan utama dari solusi yang ingin disediakan.
- b) Menentukan kebutuhan dan melakukan perancangan atau desain dari keseluruhan aplikasi mencakup :
 - (1) Tampilan *front end* dan data yang ditampilkan maupun diolah untuk *user*.
 - (2) Mekanisme *front end* agar aplikasi mudah dan nyaman untuk digunakan, seperti *loading progress* dan aktivasi *button*.
 - (3) Struktur database, apakah berbentuk JSON atau berupa *collection-document*.
 - (4) Jenis layanan Firebase yang akan digunakan, apakah Database Realtime, Firestore, dsb.
 - (5) Hal-hal lain yang berkaitan dengan input/output data dari *user* ke aplikasi maupun sebaliknya.

3. EXECUTING (DEVELOPMENT, TESTING, REVIEW & FEEDBACK)

Tahapan ini membutuhkan fleksibilitas karena memungkinkan adanya penyesuaian fitur yang ingin dibangun selama tetap searah dengan tujuan yang ingin dicapai. Pada bagian ini pula terjadi integrasi antara *front end* dan *back end* dapat membangun system yang dihendaki dan melakukan testing terhadap system tersebut. Untuk *front end*, tahapan ini berfokus pada apakah *user interface* dapat menampilkan dan menerima *input* sesuai dengan algoritma yang dibangun. Sedangkan *back end* berfokus pada konsistensi data, dimana data yang ditampilkan untuk *user* akan tetap sama sesuai dengan data yang diinputkan sebelumnya. Sehingga aplikasi dapat memberikan respon dari *user* sesuai dengan algoritma yang dibangun.

4. CLOSURE (DEPLOYMENT & RELEASE)

Tahapan ini menjadi akhir dari proses pengembangan yang dilaksanakan setelah aplikasi selesai dibangun dan berjalan sebagaimana mestinya. Finalisasi dan evaluasi perlu dilakukan agar memastikan layanan yang disediakan sesuai dengan perancangan awal. Kegiatan ini berakhir dengan dokumentasi akhir berupa pembuatan laporan dan proses presentasi.

B. RENCANA JADWAL PENGEMBANGAN

Tabel 1. Rencana jadwal pengembangan

No	Rincian Kegiatan	Hari ke-														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1.	Pencarian dan Pengembangan Topik															
2.	Menganalisa kebutuhan, spesifikasi, dan fitur															
3.	Mendesain struktur database															
4.	Mendesain <i>front end</i> (<i>wireframe</i>)															
5.	Menentukan blok diagram data dan <i>front end</i> yang terlibat															
6.	Melakukan pengkodean dan penyesuaian															
7.	Finalisasi dan Evaluasi Akhir															
8.	Pelaporan dan dokumentasi akhir															

C. SUMBER DAYA

1. PERLENGKAPAN PERANGKAT KERAS

a) Komputer Lab 28

Sebagai *device* untuk membuat aplikasi saat dikerjakan di BBPLK Bekasi dengan spesifikasi sebagai berikut :

- *Processor* : Intel Core i7-7700
- *Graphic Card* : AMD Radeon R7 450
- *Storage Unit* : 1TB HDD
- RAM : 16 GB
- *Operating System* : Windows 10 Enterprise 64-bit

b) Asus ROG Strix GL-553VD

Sebagai *device* untuk membuat aplikasi saat dikerjakan di luar jam aktif BBPLK Bekasi dengan spesifikasi sebagai berikut :

- *Processor* : Intel Core i7-7700HQ
- *Graphic Card* : NVIDIA GeForce GTX 1050
- *Storage Unit* : 1TB HDD + 128 GB SSD
- RAM : 16 GB
- *Operating System* : Windows 10 Education 64-bit

c) Xiaomi Redmi Note 8 Pro

Sebagai *device* untuk menjalankan aplikasi pada perangkat *mobile* eksternal dengan spesifikasi sebagai berikut :

- *CPU* : Octa-core
- *GPU* : Mali-G76 MC4
- *Android Version* : Android 11 (API 30)
- *Storage Unit* : 128 GB (Internal)
- RAM : 6 GB

2. PERLENGKAPAN PERANGKAT LUNAK

a) Android Studio Arctic Fox

Sebagai IDE dalam pembuatan aplikasi Android dengan spesifikasi sebagai berikut :

- *Version* : 2020.3.1.Patch 2
- *VM* : OpenJDK 64-bit Server VM

b) Firebase Console

Sebagai sistem penyedia layanan untuk database dalam aplikasi dengan spesifikasi sebagai berikut :

- *Billing Plan* : Spark (free \$0/month)
- *Firebase BoM Version* : 28.4.1 (September 13, 2021)

BAB IV

HASIL DAN PEMBAHASAN

A. INITIATION (REQUIREMENTS)

Pada tahap ini dilakukan pemilihan topik dan metode pengembangan melalui metode studi literatur, yaitu metode pengumpulan data berdasarkan referensi yang relevan dengan topik berkaitan. Literatur utama yang digunakan bersumber dari penulis pribadi berupa tugas mata kuliah dan portfolio saat uji kompetensi *IT Project Management* yang kemudian disesuaikan dengan materi yang didapatkan selama Pelatihan Berbasis Kompetensi (PBK) *Mobile Application*. Topik monitoring kebun hidroponik dipilih dengan pertimbangan bahwa latar belakang dan permasalahan yang dijabarkan sudah di alami oleh petani hidroponik di Surabaya. Sedangkan metode pengembangan *Hybrid* dipilih karena sudah ada kebutuhan awal untuk pengembangan aplikasi dan dapat mempercepat proses di tahap ini, namun dibutuhkan fleksibilitas agar dapat dilakukan penyesuaian dalam proses selanjutnya.

B. PLANNING (DESIGN)

Pada pengembangan kali ini, perancangan dilakukan berdasarkan bagian *front end* yang berkaitan dengan penyajian dan alur navigasi antar tampilan yang bersesuaian dan *back end* untuk mekanisme pengolahan data pada aplikasi. Agar kedua bagian ini dapat berfungsi sebagaimana mestinya, dimana aplikasi dapat merespon dan memberikan *output* sesuai *input* yang diberikan *user*, maka dibutuhkan perancangan *Storyboard* penggunaan sebagai referensi algoritma keseluruhan.

1. DESAIN STORYBOARD

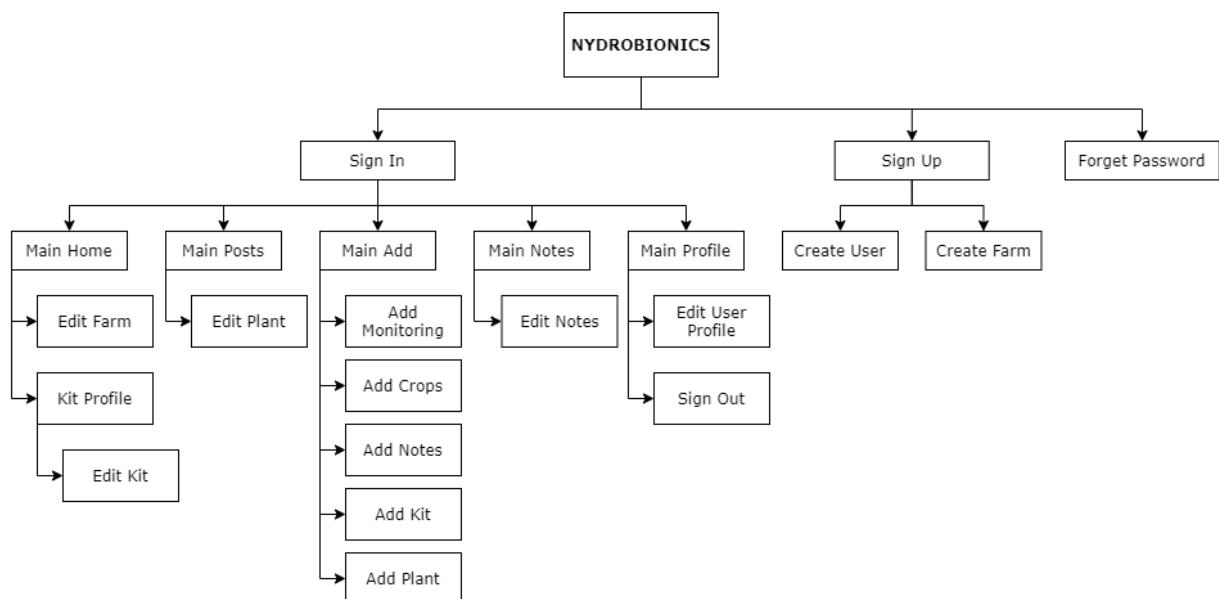
Alur penggunaan aplikasi ini dijelaskan sebagai berikut :

- a) Pengguna melakukan autentikasi menggunakan email dan *password* yang telah didaftarkan sebelumnya.
- b) Jika belum memiliki akun, maka pengguna melakukan *sign up* dengan email dan *password* kemudian membuat akun dengan memberikan data diri seperti nama, tanggal lahir, alamat, dsb. termasuk jenis akun yang akan dibuat, *owner* atau *staff*. Jika berjenis *owner*, pengguna diharuskan untuk membuat data kebun yang dimilikinya dan memberikan akses untuk *staff* yang sudah memiliki akun.

- c) Jika lupa dengan *password*, maka pengguna dapat melakukan *forget password* dengan email yang terdaftar lalu melalukan verifikasi dan membuat *password* baru melalui email sebelum dapat menggunakan Nydrobionics.
- d) Jika proses autentikasi berhasil dilakukan, maka aplikasi mengvalidasi apakah akun tersebut sudah melengkapi data pribadinya. Jika belum, maka diarahkan untuk mengurnya seperti pada proses *sign up* (tanpa mendaftarkan email). Jika sudah, maka pengguna dapat menggunakan fitur-fitur yang dikembangkan.

2. DESAIN FRONT END

a) Struktur Menu Aplikasi



Gambar 6. Struktur menu aplikasi

Sumber : Dokumentasi pribadi

Struktur menu aplikasi Nydrobioncs ditunjukkan melalui Gambar 6. Saat pengguna membuka aplikasi, maka ada 3 menu yang dapat dipilih, yaitu *Sign In*, *Sign Up*, dan *Forget Password*. *Sign In* berguna untuk memproses autentikasi. *Sign Up* berguna memproses pembuatan akun yang dilanjutkan untuk membuat profil (*Create User*) dan membuat kebun (*Create Farm*) jika akun berjenis *Owner*. Sedangkan *Forget Password* memproses permintaan pergantian *password* yang dikirimkan melalui email yang terdaftar.

Menu utama aplikasi tediri dari 5 menu yang ditampilkan jika proses autentikasi berhasil dilakukan. *Main Home* sebagai menu awal menampilkan profil kebun, data terakhir monitoring kit serta memungkinkan pengguna untuk

mengubah data kebun (*Edit Farm*), melihat detail setiap kit (*Kit Profile*), dan mengedit setiap kit (*Edit Kit*). *Main Social* menampilkan tanaman yang tersimpan pada database dan memungkinkan pengguna yang membuat untuk mengubahnya (*Edit Plant*). *Main Add* menampilkan pilihan untuk menambahkan data-data yang terlibat pada aplikasi, seperti *Add Monitoring*, *Add Crops*, *Add Notes*, *Add Kit*, dan *Add Plant*. *Main Notes* menampilkan jadwal dan catatan yang dibuat oleh pengguna terkait. Sedangkan *Main Profile* menampilkan detail profil dari akun yang sedang aktif dan memungkinkan untuk mengubahnya (*Edit Profile*) maupun memutus akses akun terhadap aplikasi (*Sign Out*).

b) Kerangka Tampilan Aplikasi

Berdasarkan struktur menu aplikasi yang dibuat, maka dapat dikembangkan kerangka tampilan aplikasi dari setiap menu yang telah ditentukan. Pada pengembangan kali ini, pembuatan kerangka tampilan aplikasi dilakukan secara manual gambar tangan sehingga hanya berupa *rough sketch* yang ditunjukkan pada Gambar 7. Seluruh proses desain kerangka tampilan hanya dilakukan dengan menggambar manual pada kertas dengan melakukan riset *front end* pada aplikasi-aplikasi yang memiliki fitur yang bersesuaian.



Gambar 7. *Mock up* aplikasi

Sumber : Dokumentasi pribadi

Selain itu digunakanlah *wireframe* (kerangka yang memiliki navigasi komponen) pada literatur terkait pada Gambar 8 sebagai pedoman yang disesuaikan terhadap fitur yang dibangun. *Wireframe* tersebut dibuat menggunakan Adobe XD memiliki 9 tampilan utama yang membangun fungsi

sign in (a), sign up (b), forget password (c), main home (d), main farm (e), main schedule (f), main profile (g), kit profile (h), dan add schedule (i). Desain front end final yang digunakan pada pengembangan kali ini langsung diimplementasikan melalui Layout XML yang bersangkutan.

a) Menu Sign In

The screen shows the Hydrobionics logo and a sign-in form. The email field contains "cahyonusuhendar@gmail.com" with a green checkmark. The password field contains "*****". A large teal "Sign In" button is at the bottom. Below it are links for "Forgot your password?" and "Create your account →".

b) Menu Sign Up

The screen shows a "Sign Up" form. It has fields for "Email" (containing "cahyonusuhendar@gmail.com" with a green checkmark), "Password" (containing "*****"), and "Confirm password" (containing "*****"). A large teal "Sign Up" button is at the bottom.

c) Menu Forget Password

The screen shows a "Forgot Password" form. It has a field for "Email" (containing "cahyonusuhendar@gmail.com" with a green checkmark). A large teal "Request Reset Link" button is at the bottom.

d) Menu Main Home

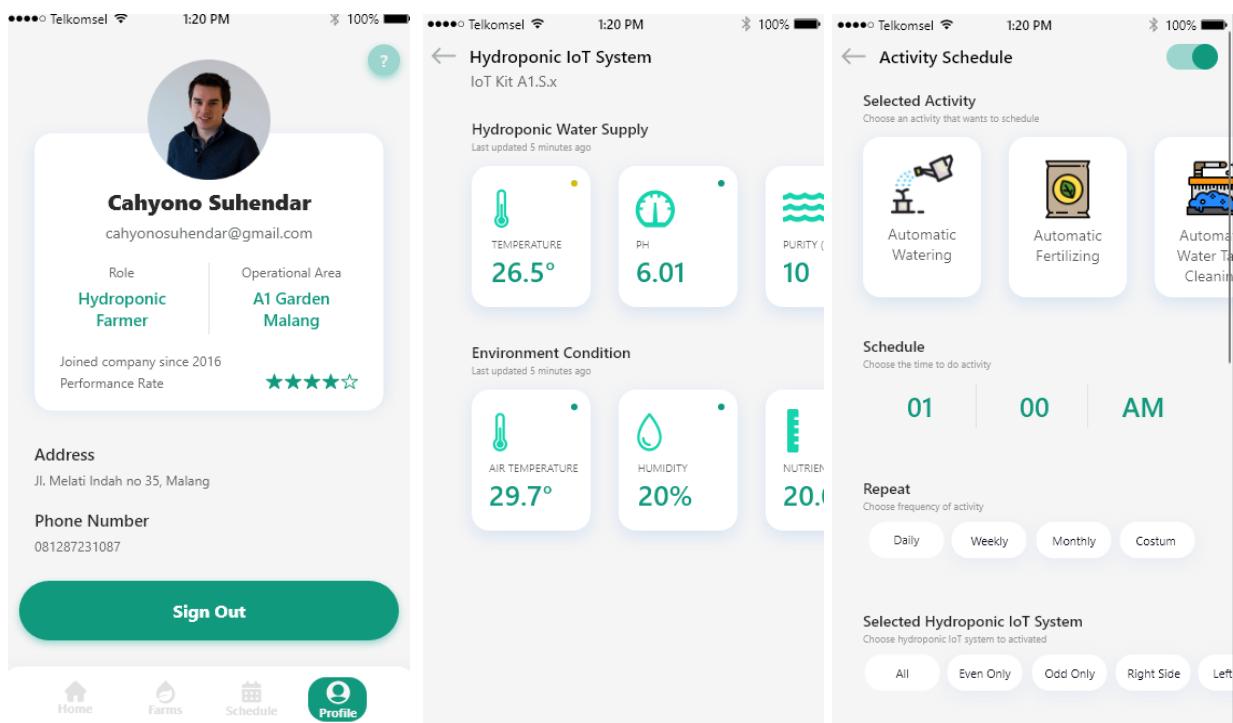
The screen shows a dashboard for "Cahyono Suhendar" (Hydroponic Farmer in Malang Farm). It includes a profile picture, a 5-star rating, and sections for "Next Activities" (Hydroponic Water Tank Cleaning at 02:00 PM, Automatic Watering at 04:00 PM, Turning on the Garden Main Light at 05:45 PM), "Hydroponic Water Supply" (last updated 5 minutes ago with values: Temperature 26.5°, PH 6.01, Purity (NTU) 10), and "Environment Condition" (last updated 5 minutes ago with values: Air Temperature 29.7°, Humidity 20%, Nutrient Level 20.0%). Navigation tabs at the bottom include Home, Farms, Schedule, and Profile.

e) Menu Main Farm

The screen shows a "Farm Profile" for "A1 Garden, Malang Farm". It displays "Hydroponic Water Supply" (Temperature 26.5°, PH 6.01, Purity (NTU) 10) and "Environment Condition" (Air Temperature 29.7°, Humidity 20%, Nutrient Level 20.0%). It also shows "Garden Layout" (last updated 5 minutes ago). Navigation tabs at the bottom include Home, Farms, Schedule, and Profile.

f) Menu Main Schedule

The screen shows an "Activities Schedule" section. It lists tasks with their times: Hydroponic Water Tank Cleaning (01:30 PM, 04:00 PM, 05:45 PM), Automatic Watering (00:00 AM, 05:00 AM, 08:00 AM), Turning on the Garden Main Light (00:00 AM, 05:00 AM, 08:00 AM), and Automatic Fertilizing (09:00 AM). It also shows "Automatic Watering" (08:00 AM, 04:00 PM, 00:00 AM), "Automatic Fertilizing" (09:00 AM), "Automatic Water Tank Cleaning" (08:00 AM), and "Automatic Data Retrieval" (08:00 AM). A green "+" button is at the bottom right. Navigation tabs at the bottom include Home, Farms, Schedule, and Profile.



Gambar 8. Wireframe aplikasi yang diterapkan pada literatur

Sumber : Dokumentasi pribadi

3. DESAIN BACK END

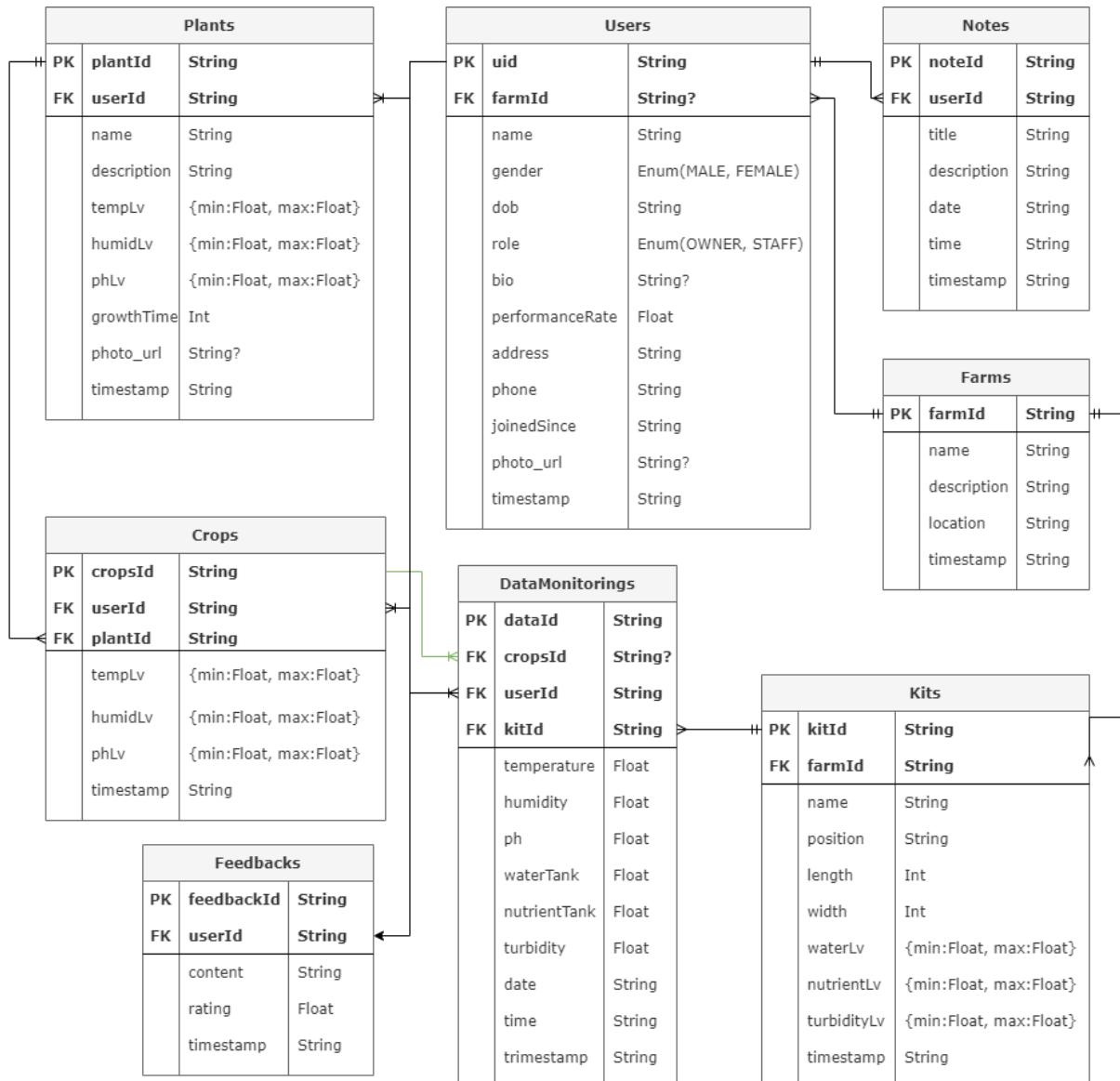
a) Entity Relationship Diagram (ERD)

Berdasarkan desain *front end* kemudian ditentukan kebutuhan data dan atribut yang dapat memberikan fungsionalitas alur data yang diharapkan. Model atau rancangan untuk memenuhi kebutuhan data tersebut ditunjukkan melalui *entity relationship diagram* (ERD) pada Gambar 9. Nydrobionics membutuhkan 8 tabel data (nama tabel berbentuk pruler dan data berbentuk singular, sebagai contoh tabel *users* dengan data *user*) dengan relasi antar tabel :

- (1) Setiap *user* dapat memiliki banyak *note* dan *feedback*, *dataMonitoring*, *crops*, dan *plant* namun hanya memiliki 1 *farm*.
- (2) Setiap *note* hanya memiliki 1 *user*.
- (3) Setiap *farm* dapat memiliki banyak *kit* dan *user*.
- (4) Setiap *kit* dapat memiliki banyak *dataMonitoring* namun hanya memiliki 1 *farm*.
- (5) Setiap *crops* dapat memiliki banyak *dataMonitoring* namun hanya memiliki 1 *user* dan 1 *plant*.

(6) Setiap *plant* dapat memiliki banyak *crop* dan *dataMonitoring* namun hanya memiliki 1 *user*.

(7) Setiap *feedback* hanya dimiliki oleh 1 *user*.



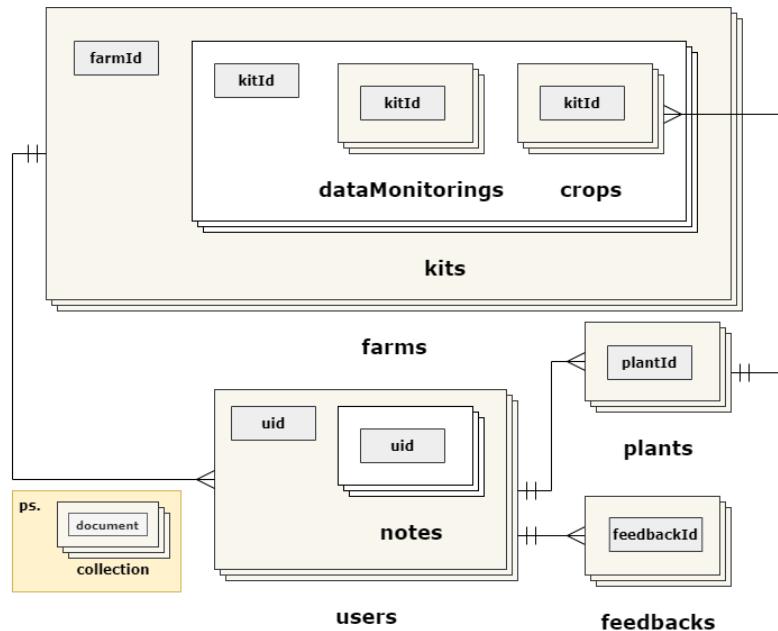
Gambar 9. *Entity Relationship Diagram (ERD)* aplikasi

Sumber : Dokumentasi pribadi

b) Struktur Database

Berdasarkan kebutuhan dan relasi data dari *Entity Relationship Diagram* (ERD), maka dapat ditentukan struktur database yang diterapkan sesuai dengan layanan penyedia penyimpanan yang digunakan. Nydrobionics menggunakan Firebase Cloud Firestore karena layanan tersebut menyediakan kemudahan untuk pengembangan *mobile application*. Karena Firebase Cloud Firestore menerapkan arsitektur *document collection*, maka atribut data yang ada pada

ERD disesuaikan ke dalam bentuk *document collection* dengan struktur database beserta relasi antar *collection* utamanya ditunjukkan pada Gambar 10.



Gambar 10. Struktur database *document collection* dan relasinya

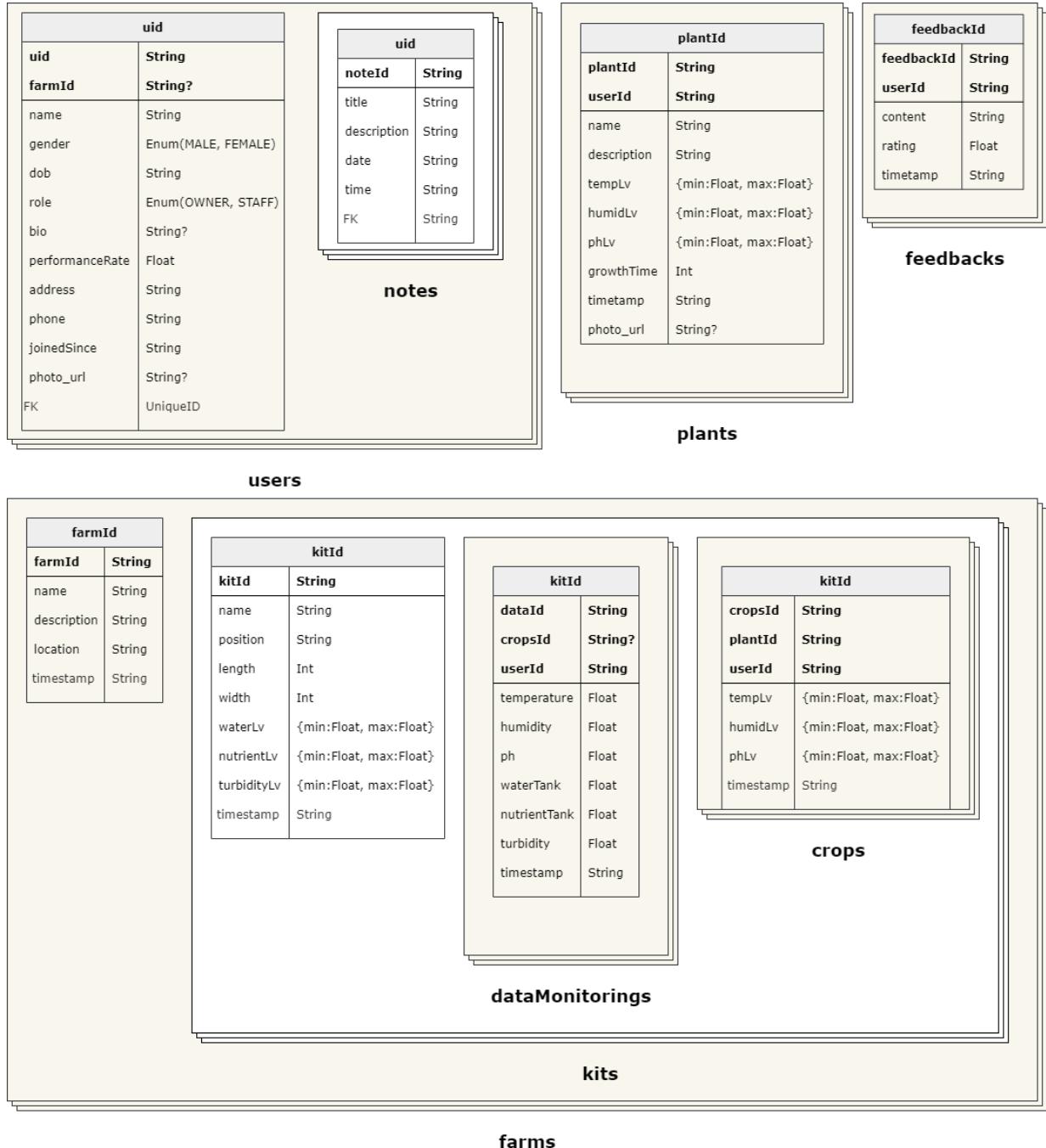
Sumber : Dokumentasi pribadi

Proses yang dilakukan yaitu dengan menentukan tabel manakah yang dapat dijadikan sebagai *subcollection* sehingga *foreign key* yang menghubungkan antar tabel dapat dihilangkan. Relasi “*one to many*” dijadikan pedoman dalam penentuan tersebut hingga menghasilkan 3 *collection* utama, yaitu *users*, *farms* dan *plants* (nama *collection* berbentuk prplural dan *document* berbentuk singular, sebagai contoh *document users* dengan data *user*) dengan sub-struktur yang berbeda seperti pada Gambar 11. Kegunaan dan relasi antar *collection-document* maupun *collection-collection* setara dengan relasi pada ERD, yaitu :

- (1) Setiap *user* dapat memiliki banyak *plant* dan *feedback* dan hanya terikat dengan 1 *farm*.
- (2) Setiap *farm* dapat memiliki banyak *user*.
- (3) Setiap *plant* dapat memiliki banyak *crops* (*subcollection farms*) dan hanya terikat dengan 1 *user*.
- (4) Setiap *feedback* hanya terikat dengan 1 *user*.

Penghapus *foreign key* dari setiap table pada ERD saat menjadi *subcollection* dari *collection* utama dilakukan untuk menghilangkan kemungkinan

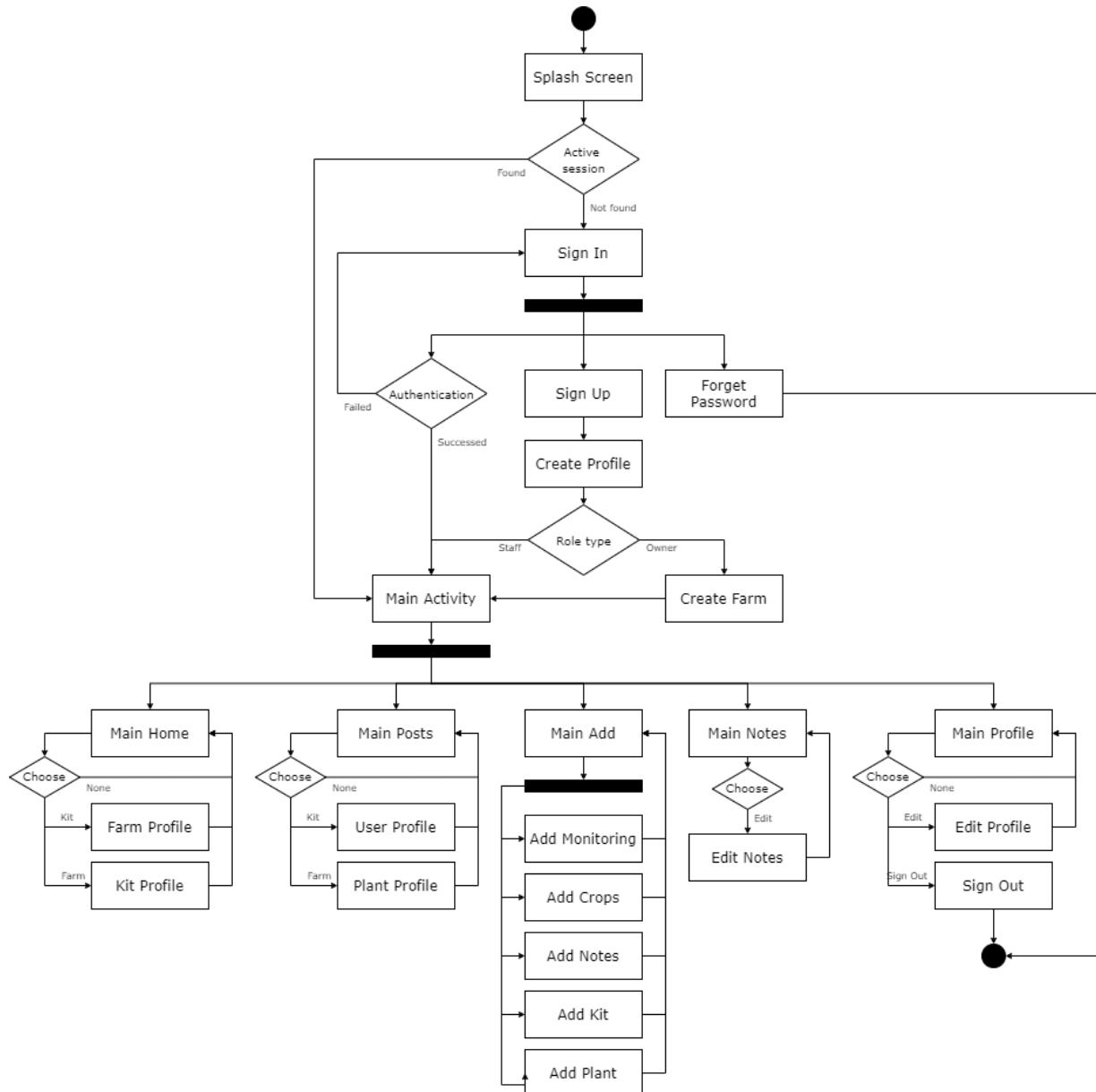
redudansi data. Sebagai contohnya untuk tabel *Notes* pada ERD (Gambar 9) memiliki *foreign key* *userId* yang terhubung dengan *uid* di tabel *Users*, disesuaikan sebagai *subcollection* dari *collection* utama *users* (Gambar 11) dengan *document id* yang digunakan adalah *uid* sehingga *foreign key* dapat dihilangkan.



Gambar 11. Detail struktur database beserta atribut datanya

Sumber : Dokumentasi pribadi

c) *Activity Diagram*



Gambar 12. *Activity Diagram* dari aplikasi

Sumber : Dokumentasi pribadi

Alur kerja dari aplikasi dijabarkan melalui *activity diagram* pada Gambar 12 yang dimulai pada *Splash Screen* dimana berfungsi untuk mengecek apakah ada akun yang sudah ter-sign in (*active session*) sebelumnya. Jika ada maka akan meminta *user profile* dari akun tersebut dan menuju ke *Main Activity*. Jika belum, maka menuju ke *Sign In*. Pada tahap ini, pengguna diberikan pilihan apakah ingin melakukan *Sign In* dengan email dan *password* (proses autentikasi), *Sign Up* yang dilanjut dengan *Create Profile*, atau *Forget Password* untuk meminta verifikasi *reset password* melalui email yang terdaftar. Pada *sign*

in dan *sign up* jika proses berhasil dilakukan maka diarahkan untuk ke *Main Activity*, sedangkan *forget password* mengharuskan pengguna untuk mengecek email sehingga mengakhiri alur kerja aplikasi.

Saat *Main Activity* ditampilkan, pengguna diberikan 5 pilihan *activity* yang bisa dilakukan dengan *Main Home* sebagai *default home*-nya. *Main Activity* bertindak sebagai pusat aplikasi yang memberikan pilihan untuk menampilkan seluruh data yang terlibat pada aplikasi. Pada *Main Home* melibatkan data tentang *farms* dan ringkasan data *kits* yang memungkinkan untuk menampilkan detail *kit*. *Main Social* memungkinkan pengguna untuk melihat profil *user* dan *plant* sesuai pilihannya. *Main Add* berfokus pada mekanisme penambahan data sesuai dengan menu untuk menjalankan *activity* yang berkaitan. *Main Notes* menampilkan seluruh *note* yang pernah dibuat oleh pengguna dan bisa memodifikasi data tersebut. Dan *Main Profile* memberikan pengguna pilihan untuk mengedit *user profile*-nya atau melakukan *Sign Out* untuk mengakhiri aplikasi.

C. EXECUTING (DEVELOPMENT, TESTING, REVIEW & FEEDBACK)

Setelah *blueprint* aplikasi yang menyelesaikan solusi atas permasalahan ditentukan, maka implementasi desain menjadi kodingan dilakukan. Pada tahap ini memungkinkan adanya perubahan dari desain yang dibuat dikarenakan menyesuaikan kendala dan hambatan yang dialami. Agar pekerjaan yang dilakukan dapat dilacak keterselesaiannya demi mencapai target selesai tepat waktu, maka dilakukan *task break down* menjadi penggeraan *View Layer* berupa Layout XML dan navigasi antar *activity*, penggeraan *ViewModel* untuk menambah dan mengedit data Firebase, penggeraan *ViewModel* untuk menampilkan data Firebase beserta adapter untuk data berbentuk list, dan *finishing* berupa menyesuaikan tema aplikasi dan supergrafis seperti ikon dan logo yang digunakan.

1. PROFIL APLIKASI

Nydrobionics adalah *mobile application* berbasis Android untuk monitoring kebun hidroponik agar memudahkan system pelaporan kondisi kebun dan meningkatkan akurasi data saat dicek dan dilaporkan. Aplikasi ini memungkinkan pengguna untuk memantau keadaan kebun hidroponik dengan tujuan dapat mengatur konsistensi variabel-variabel yang memengaruhi kualitas hasil panen. Terdiri dari 2 tipe akun, yaitu

Owner dan *Staff* yang memberikan batasan terhadap *jobdesk* setiap pengguna. Akun *Owner* sebagai pemilik kebun dapat mengatur anggota *staff* pada kebunnya dan mengalokasikan *staff* yang bertanggung jawab terhadap beban pekerjaannya. Dokumentasi dari keseluruhan project dapat diakses melalui [GitHub](#) maupun [Google Drive](#).



Gambar 13. Logo aplikasi Nydrobionics

Sumber : Dokumentasi pribadi

Nydrobionics dibangun dengan minimum SDK 26 dan target SDK 30, sehingga dapat berjalan optimum pada *mobile device* Android 11 dan seminimalnya *mobile device* dengan *operating system* Android Oreo (8.0). Aplikasi ini juga membutuhkan penggunanya untuk memperbolehkan beberapa *permission* seperti :

- a) Menggunakan jaringan internet (android.permission.INTERNET)
- b) Mengakses media files (android.permission.READ_EXTERNAL_STORAGE)
- c) Mengetahui kondisi jaringan (android.permission.ACCESS_NETWORK_STATE)
- d) Mengakses lokasi yang tepat (android.permission.ACCESS_FINE_LOCATION)
- e) Mengetahui perkiraan lokasi (android.permission.ACCESS_COARSE_LOCATION)
- f) Mengetahui kondisi Wi-Fi (android.permission.ACCESS_WIFI_STATE)
- g) Menggunakan kamera (android.permission.CAMERA)
- h) Memulai telepon (android.permission.CALL_PHONE)

2. FITUR YANG DIKEMBANGKAN

Berikut ini beberapa fitur yang dikembangkan pada Nydrobionics :

a) *Hydroponic Monitoring*

Pengguna dapat menyimpan data keadaan kebun seperti kondisi suhu ($^{\circ}\text{C}$), kelembaban udara (RH), tingkat keasaman air (pH), kejernihan air (NTU), kapasitas tanki air dan pupuk cair (L). Data yang telah tersimpan dapat dilihat dan dibandingkan dengan keseluruhan data sehingga dapat digunakan sebagai indicator untuk mengontrol kondisi tersebut sesuai dengan tanaman yang ditanam.

b) *Plant NydroDB*

Untuk mendukung monitoring keadaan kebun, Nydrobionics dilengkapi dengan system database tanaman, sehingga dapat memberikan informasi apakah kondisi kebun saat ini berada dalam batas ideal untuk tanaman yang ditanam. Pengguna dapat menyimpan data berupa profil tanaman seperti nama tanaman, nilai ideal (suhu, kelembaban udara, dan pH air), dan estimasi waktu yang dibutuhkan untuk siap panen. Selain itu juga memungkinkan untuk mengubah data tersebut jika diperlukan.

c) *Schedule and Notes*

Untuk mendukung pekerjaan petani kebun hidroponik, Nydrobionics dilengkapi dengan system yang memudahkan penggunanya untuk pengingat jadwal pekerjaan dan catatan.

3. IMPLEMENTASI APLIKASI

a) **Struktur Data**

Agar data yang terlibat dapat ditampilkan dan diolah dari aplikasi ke database maupun sebaliknya, maka diharuskan untuk menyiapkan struktur data pada aplikasi, baik data yang secara langsung ditransmisikan ke database maupun variabel-variabel lain yang digunakan dalam pengolahannya. Terdiri dari 7 *data class* utama (model) untuk mengolah data dari database yang berisi fungsi untuk mengkonversi struktur yang terlibat, seperti *Data Class* ke *Hash Map* (*Model.toHashMap()*) untuk pengiriman data dan *Document Snapshot* ke *Data Class* (*DocumentSnapshot.toObject()*) untuk penerimaan data. Pada Kode 5 menunjukkan *data class* untuk *UserModel* yang berisi deklarasi atribut dan fungsi konversi untuk memudahkan proses pengolahan data. Untuk model lain dapat dilihat melalui [lampiran](#).

```
@Parcelize
data class UserModel(
    var name : String? = null,
    var email: String? =null,
    var gender : String? = null,
    var dob : String? = null,
    var role : String? = null,
    var bio : String? = null,
    var uid : String? = null,
    var performanceRate : Float? = null,
    var address : String? = null,
    var phone : String? = null,
```

```

        var joinedSince : String? = null,
        var photo_url : String? = null,
        var farmId : String? = null,
        var timestamp : String? = null
    ) : Parcelable {
    companion object {
        fun DocumentSnapshot.toUserModel() : UserModel? {
            try{
                val name = getString("name")
                val email = getString("email")
                val gender = getString("gender")
                val dob = getString("dob")
                val role = getString("role")
                val bio = getString("bio")
                val uid = getString("uid")
                val performanceRate : Double? = get("performanceRate")
as Double?
                val address = getString("address")
                val phone = getString("phone")
                val joinedSince = getString("joinedSince")
                val photo_url = getString("photo_url")
                val farmId = getString("farmId")
                val timestamp = getString("timestamp")
                val output = UserModel(name, email, gender, dob,
                    role, bio, uid,
                    performanceRate?.toFloat(),
                    address, phone, joinedSince,
                    photo_url, farmId, timestamp)
                return output
            } catch (e: Exception) {
                Log.e(TAG, "Error converting $TAG", e)
                return null
            }
        }
    }

    fun UserModel.toHashMap():HashMap<String,Any?>{
        val output = hashMapOf<String,Any?>()
        output["name"] = name
        output["email"] = email
        output["gender"] = gender
        output["dob"] = dob
        output["role"] = role
        output["bio"] = bio
        output["uid"] = uid
        output["performanceRate"] = performanceRate
        output["address"] = address
        output["phone"] = phone
        output["joinedSince"] = joinedSince
        output["photo_url"] = photo_url
        output["farmId"] = farmId
        output["timestamp"] = ViewUtility().getCurrentTimestamp()
        return output
    }

    private const val TAG = "UserModel"
}
}

```

Kode 5. Struktur data *UserModel* beserta konverternya

Untuk memudahkan pengelolaan data enum pada aplikasi (*Gender* dan *Role*), maka dibuatlah *Enum Class* termasuk fungsi tambahan seperti *toString()* untuk mengubah *default* dari fungsi *toString()* yang ditunjukkan pada Kode 6. Pada *enum class Gender* (a) juga terdapat fungsi *getPosition()* karena *view component* yang digunakan untuk menampilkan data tersebut merupakan *SegmentedButton* yang menghasilkan *value* dari inputan pengguna berupa data posisi yang dipilih.

```
enum class Gender(var gender: String) {
    MALE("male") {
        override fun getPosition(): Int {
            return 0
        }
    },
    FEMALE("female") {
        override fun getPosition(): Int {
            return 1
        }
    };
}

override fun toString(): String {
    return gender
}

abstract fun getPosition(): Int

companion object{
    fun getType(type: String) : Gender?{
        return when(type){
            "male" -> MALE
            "female" -> FEMALE
            else -> null
        }
    }
}
}
```

a) *Enum Class Gender*

```
enum class Role(var role: String) {
    OWNER("owner"),
    STAFF("staff");

    override fun toString(): String {
        return role
    }

    companion object{
        fun getType(type: String) : Role? {
            return when(type){
                "owner" -> OWNER
                "female" -> STAFF
                else -> null
            }
        }
    }
}
```

```
}
```

b) *Enum Class Role*

Kode 6. Enum class *Gender* dan *Role* untuk memudahkan pengelolaan data terkait

b) Algoritma Input dan Output

```
interface LoadingInterface {
    var context: Context?
    var circularProgressBar: CircularProgressBar?
    var textInputEditTexts: ArrayList<TextInputEditText>?
    var viewsAsButton: ArrayList<View>?
    var numberPickers: ArrayList<ClickNumberPickerView>?
    var checkBoxes: ArrayList<CheckBox>?
    var actionBar: ActionBar?

    var initialLoadingState: Boolean
    var isLoading: Boolean
        get() = initialLoadingState
        set(value) {
            textInputEditTexts?.forEach {
                it.isCursorVisible = !value
                it.isFocusable = !value
                it.isFocusableInTouchMode = !value
            }
            viewsAsButton?.forEach {
                it.isEnabled = !value
            }
            numberPickers?.forEach {
                it.isEnabled = !value
            }
            circularProgressBar?.apply {
                this.isEnabled = true
                if (value) {
                    startAnimation()
                } else {
                    val handler = Handler(Looper.getMainLooper())
                    handler.postDelayed({
                        revertAnimation()
                    }, 3000)
                }
            }
            checkBoxes?.forEach {
                it.isClickable = !value
            }
            actionBar?.setDisplayShowHomeEnabled(value)
            onLoadingChangeListener { (initialLoadingState) }
            initialLoadingState = value
        }
    }

    fun onLoadingChangeListener(function: (isLoading: Boolean) -> Unit)
}
```

Kode 7. LoadingInterface yang berfungsi untuk mengatur aktivasi view saat pengiriman data
Terdapat 3 pengelompokkan menu berdasarkan fungsi yang dijalankan oleh pengguna, yaitu menu awal aplikasi berupa *Splash Screen*, *Sign In*, *Sign Up*,

dan *Forget Password*, menu *Create Profile* yang terdiri *Create User Profile* dan *Create Farm Profile*, serta menu *Main* berupa *Main Home*, *Main Posts*, *Main Add*, *Main Notes*, dan *Main Profile*. Masing-masing memiliki fungsi yang berbeda-beda, namun penerimaan input *user* dan pemberian responnya memiliki mekanisme yang mirip, yaitu saat input yang diberikan belum sesuai dengan syarat, maka tombol untuk melanjutkan ke proses selanjutnya tidak diaktifkan sehingga pengguna diharuskan untuk melengkapi input tersebut. Selain itu saat proses pengiriman data ke Firebase dilakukan, pengguna juga tidak dapat memberikan input, baik dengan *soft keyboard* maupun dengan menekan *button* atau *view*. Hal ini dikarenakan adanya penggunaan *LoadingInterface* yang diberikan nilainya saat pengiriman dan penerimaan data dimulai serta selesai dilakukan seperti pada Kode 7.

Setiap *activity* yang dibuat menerapkan arsitektur MVVM dan dikombinasikan dengan ViewBinding, dimana membutuhkan *class ViewModel* untuk setiap *activity* yang dibuat. Dalam pelaksanaan tugas akhir ini ada beberapa *activity* yang menggunakan *ViewModel* yang sama dikarenakan adanya kesamaan fungsi yang dibutuhkan, seperti *CreateProfileViewModel* yang digunakan untuk *CreateProfileActivity* (*CreateUserFragment* serta *CreateFarmFragment*), *EditProfileUserActivity*, dan *EditProfileFarmActivity*. Bagian yang membedakan penggunaan untuk *create* dan *edit* adalah pendefinisian *reference id* untuk pengiriman data ke Firestore seperti pada Kode 1 bagian *farmId*, dimana jika *farmId* bernilai *farmId* jika sudah memiliki nilai sebelumnya dan bernilai sesuai dengan *generate reference id* jika *farmId* masih bernilai null. Bagian inilah yang dimanfaatkan untuk mengubah fungsi dari *create farm* menjadi *edit farm*.

```
...
fun createFarmProfile(name:String, description:String,
location:String) {
    try {
        createProfileError.value = ""
        val db = firestore.collection("farms")
        val ref : DocumentReference = db.document()

        farmModel.value?.apply {
            this.farmId = farmId ?: ref.id
            this.name = name
            this.description = description
            this.location = location
        }
    }
}
```

```

db.document(farmModel.value!!.farmId!!).set(farmModel.value!!.toHashMap
()).addOnCompleteListener {
    if(it.isSuccessful) {
        userModel.value?.farmId = farmModel.value!!.farmId!!
        sendUserProfile()
        isFarmCreated.value = isUserCreated.value == true
    } else {
        createProfileError.value = it.exception.toString()
        isFarmCreated.value = false
    }
}
} catch (e:Exception) {
    Log.e(TAG, "Error submit farm", e)
}
...

```

Kode 8. Penggunaan CreateProfileViewModel untuk membuat dan mengubah data Farm

Selain penggunaan ViewModel untuk beberapa *activity*, penggunaan ulang *layout* juga dilakukan di beberapa bagian seperti untuk *EditProfileUserActivity* yang juga menggunakan layout untuk *CreateUserFragment* seperti pada Kode 9 dengan bentuk layout seperti pada Gambar 14 . Untuk mendefinisikan nilai awal data, maka dibutuhkan *observer* pada *activity* untuk mengetahui apakah ada nilai inisiasi awal yang diberikan dari *activity* sebelumnya (parsing data melalui intent seperti pada Kode 10). Selain itu, dilakukan beberapa penyesuaian seperti menghilangkan atau menambahkan beberapa komponen *view* yang dibutuhkan seperti pada b. Penerapan MVVM lebih lengkap lainnya dapat diakses pada **lampiran** maupun melalui dokumentasi project pada [GitHub](#).

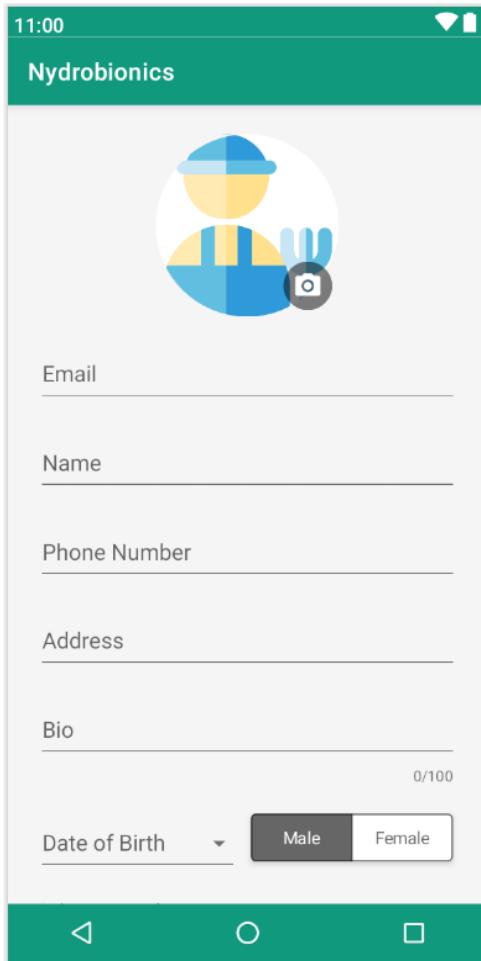
```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".activity.EditProfileUserActivity">

    <include
        android:id="@+id/editProfileFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        layout="@layout/fragment_create_user" />
</ScrollView>

```

Kode 9. Penggunaan ulang layout *CreateUserFragment* untuk *EditProfileUserActivity*



Gambar 14. Hasil dari render untuk layout *CreateUserFragment* dan *EditProfileUserActivity*
Sumber : Dokumentasi pribadi

```
...
R.id.drawer_edit -> {
    val intent = Intent(this, EditProfileUserActivity::class.java)
    intent.putExtra(BuildConfig.CURRENT_USER,
viewModel.getCurrentUser().value)
    startActivity(intent)
    drawerLayout.closeDrawer(GravityCompat.END)
    true
}
...
```

a) Pengiriman data User dari *MainActivity* ke *EditProfileUserActivity*

```
...
viewModel.setCurrentUser(intent.getParcelableExtra<UserModel>(BuildCon
fig.CURRENT_USER))
bindingFragment.apply {
    createUserRoleGroup.visibility = View.GONE
    createUserSubmit.text = getString(R.string.save)

    viewModel.getCurrentUserModel()?.let {
        createUserEmail.setText(Firebase.auth.currentUser?.email)
        createUserName.setText(it.name ?: "")
        createUserPhone.setText(it.phone ?: "")
    }
}
```

```

        createUserAddress.setText(it.address ?: "")
        createUserBio.setText(it.bio ?: "")
        createUserDOB.setText(it.dob ?: "")
        it.photo_url?.let {
            Glide.with(this@EditProfileUserActivity)
                .load(it)
                .centerCrop()
                .into(createUserPhoto)
        }

createUserGender.setPosition(Gender.getType(it.gender!!)!!.getPosition
(), false)

        strEdt[it.name ?: ""].text = createUserName
        strEdt[it.phone ?: ""].text = createUserPhone
        strEdt[it.address ?: ""].text = createUserAddress
        strEdt[it.bio ?: ""].text = createUserBio
        strEdt[it.dob ?: ""].text = createUserDOB
    }
}
...

```

b) Penerimaan data di *EditProfileUserActivity* dan menampilkan data ke *view* yang bersesuaian

Kode 10. Pengiriman, penerimaan, dan pengaturan inisiali awal untuk *EditProfileUserActivity*

c) Utility Class

Saat melakukan implementasi *back end*, sangat memungkinkan adanya penambahan *class* selain *class* utama untuk *activity* maupun struktur data dikarenakan adanya *utility class* ini dapat memudahkan proses pengolahan dan pengelolaan data dengan mengurangi potensi redundansi *block code* dan menghindari adanya ketidak konsistenan algoritma pengolahan data. Sebagai contoh pada *IntentUtility* yang digunakan untuk melakukan navigasi di luar aplikasi (*implicit intent*). Beberapa fitur yang memanfaatkan *class* ini di antaranya untuk membuka *maps* sesuai dengan *key* yang ditentukan seperti pada Kode 11 di *MainProfileFragment* dan *ProfileUserActiviy* yang penulisan *code*-nya menjadi lebih ringkas (a) dibandingkan harus menulis keseluruhan *block code* (b) setiap kali *request permission* dibutuhkan.

```

...
IntentUtility(this).openMaps(mainProfileAddress.toString())
...

```

a) Penggunaan *IntentUtility* untuk membuka maps

```

class IntentUtility(val context: Context) {
    fun openBrowser(url: String) {
        val intent = Intent()
        intent.addCategory(Intent.CATEGORY_BROWSABLE)
    }
}

```

```
        intent.action = Intent.ACTION_VIEW
        intent.data = Uri.parse(String.format(url))
        context.startActivity(intent)
    }

    fun openMaps(location:String){
        openBrowser(
            "https://www.google.com/maps?q=${location.replace(
                ' ', '+'
            )}"
        )
    }
}
```

b) Code dari membuka *maps* pada *IntentUtility*

Kode 11. Penggunaan *IntentUtility* untuk membuka *maps*

D. CLOSSURE (DEPLOYMENT & RELEASE)

Setelah aplikasi dapat menjalankan program sesuai dengan rancangan awal, maka dilakukan pembuatan laporan akhir yang mendokumentasikan keberlangsungan kegiatan pengembangan. Pengembangan aplikasi akan berakhir setelah melakukan presentasi untuk melaporkan hasil kegiatan yang telah dilaksanakan.

BAB V

PENUTUP

A. KESIMPULAN

Dari pelaksanaan pengembangan aplikasi yang sudah dilakukan, penulis dapat menemui beberapa kendala dan hambatan yang dapat ditarik beberapa kesimpulan sebagai berikut :

1. Beberapa *block code* yang dibutuhkan secara berulang dapat dijadikan satu *class* agar memudahkan modifikasi dan menjamin konsistensi pengkodean khususnya pada *class ViewUtility, IntentUtility, dan LoadingInterface*.
2. Penerapan *open source library* yang disediakan oleh developer lain melalui *github* untuk mencapai tujuan tertentu dapat mempermudah dan mempercepat proses pengembangan serta melatih diri dalam membaca *block code* dan memahami algoritma tersebut. Namun terdapat kekurangan jika membutuhkan sub-variable/resource namun di *private* maka membutuhkan waktu untuk explore lebih lanjut.

B. SARAN

Demi pengembangan lebih lanjut mengenai tugas akhir ini, disarankan beberapa langkah lanjutan sebagai berikut :

1. Jika aplikasi ini dikembangkan lebih lanjut, dapat memperbaiki dan menambah fitur-fitur lain seperti :
 - a) Menambah jenis tipe postingan yang dapat dibuat oleh pengguna dan melengkapinya dengan *attach document* seperti gambar, video, maupun dokumen lainnya.
 - b) Membedakan jenis *notes* untuk *reminder-schedule, notes*, maupun *to-do list* agar memberikan pengguna pilihan dalam memanfaatkan fitur yang sudah ada.
 - c) Menambahkan fitur *deeplink* yang langsung terkoneksi dengan Firebase saat melakukan *forget password* agar tidak memerlukan pihak ketiga dalam mencapai fungsi tersebut (dari email konfirmasi langsung ter-*redirect* ke aplikasi).
2. Dapat memperbaiki *codingan back end* agar lebih efektif dan efisien serta dapat dimengerti lebih mudah oleh para pembaca.

DAFTAR PUSTAKA

- [1] J. Chen, "Android Operating System," Dotdash, 03 Februari 2021. [Online]. Available: <https://www.investopedia.com/terms/a/android-operating-system.asp>. [Accessed 07 Oktober 2021].
- [2] P. Christensson, "Android Definition," TechTerms, 16 Mei 2016. [Online]. Available: <https://techterms.com/definition/android>. [Accessed 5 Oktober 2021].
- [3] Wikipedia, "Android Version History," 04 Oktober 2021. [Online]. Available: https://en.wikipedia.org/wiki/Android_version_history. [Accessed 2021 Oktober 07].
- [4] A. N. Anrini, "Apa Bedanya Android Studio dan Android SDK?," Definite, 31 Agustus 2020. [Online]. Available: <https://definite.co.id/blogs/apa-bedanya-android-studio-dan-android-sdk/>. [Accessed 07 Oktober 2021].
- [5] S. Okta, "Mengenal Arsitektur Model View ViewModel (MVVM) di Android — Bagian 1," 21 Desember 2017. [Online]. Available: <https://syafdia.medium.com/mengenal-arsitektur-model-view-viewmodel-mvvm-di-android-2e52ec98a74e>. [Accessed 07 Oktober 2021].
- [6] Android Developers, "What's New in Architecture Components (Google I/O'19)," 09 Mei 2019. [Online]. Available: https://www.youtube.com/watch?v=Qxj2eBmXLHg&t=446s&ab_channel=AndroidDevelopers. [Accessed 2021 Oktober 07].
- [7] M. Firdaus, "View Binding; Membuat Aplikasi Android menjadi Lebih Paten — Bukan Kaleng-kaleng!," 31 Maret 2020. [Online]. Available: <https://medium.com/gits-apps-insight/view-binding-membuat-aplikasi-android-menjadi-paten-bukan-kaleng-kaleng-7dca4b5f4739>. [Accessed 07 Oktober 2021].
- [8] Dicoding Intern, "Apa itu Firebase? Pengertian, Jenis-Jenis, dan Fungsi Kegunaannya," Dicoding, 25 November 2020. [Online]. Available: <https://www.dicoding.com/blog/apa-itu-firebase-pengertian-jenis-jenis-dan-fungsi-kegunaannya/>. [Accessed 2021 Oktober 2021].
- [9] Google Developers, "Get Started with Firebase Authentication on Android," [Online]. Available: <https://firebase.google.com/docs/auth/android/start>.

[Accessed 22 Oktober 2021].

- [10] R. Vyas, "Firebase Cloud Firestore v/s Firebase Realtime Database," 11 Oktober 2017. [Online]. Available: <https://medium.com/zero-equals-false/firebase-cloud-firestore-v-s-firebase-realtime-database-931d4265d4b0>. [Accessed 2021 Oktober 07].
- [11] Google Developers, "Create a Cloud Storage reference on Android," [Online]. Available: <https://firebase.google.com/docs/storage/android/create-reference>. [Accessed 22 Oktober 2021].

LAMPIRAN

Untuk memudahkan pembuatan lampiran, maka kontennya di-*generate* menjadi PDF menggunakan LaTeX (TexStudio sebagai IDE-nya) dengan memanfaatkan *package* pdffpages dan minted. LaTeX adalah sebuah bahasa pemrograman yang difokuskan untuk membuat sebuah dokumen dan banyak digunakan untuk membuat thesis maupun jurnal ilmiah. *Source code* dalam LaTeX untuk membuat lampiran ini ditunjukkan melalui Kode 1.

```
1  \documentclass{article}
2  \title{\vspace{-5ex} \bfseries {\sc LAMPIRAN}}
3  \date{\vspace{-10ex}}
4  \usepackage{geometry}
5  \geometry{
6      a4paper,
7      left=40mm,
8      top=30mm,
9      right=30mm,
10     bottom=30mm
11 }
12 \usepackage{pdffpages}
13 \usepackage[bookmarks=false]{hyperref}
14 \usepackage{minted}
15 \usepackage{caption}
16 \usepackage{setspace}
17 \usepackage{fancyhdr}
18 \pagestyle{fancy}
19 \fancyhf{}
20 \fancyfoot[R]{Laporan Tugas Akhir Aplikasi Monitoring Kebun Hidroponik Berbasis
→ \textit{Mobile} $\vert$ \thepage}
21 \fancypagestyle{plain}{
22     \renewcommand{\headrulewidth}{0pt}
23     \fancyhf{}%
24     \fancyfoot[R]{Laporan Tugas Akhir Aplikasi Monitoring Kebun Hidroponik Berbasis
→ \textit{Mobile} $\vert$ \thepage}
25 }
26
27 \setcounter{page}{43}
28 \setcounter{secnumdepth}{4}
29 \setminted{
30     linenos,
31     breaklines,
32     breakanywhere,
33     fontsize=\footnotesize,
34     frame=single,
35     framesep=10pt,
36     baselinestretch=1
37 }
38 \newenvironment{multipageListing}{
39     \setstretch{1}
40     \captionsetup{type=listing, justification=centering, skip=-5pt}
41 }{}
42
43 \renewcommand{\listingscaption}{Kode}
44 \renewcommand{\thesection}{\Alph{section}.}
45 \renewcommand{\thesubsection}{\arabic{subsection}.}
46 \renewcommand{\thesubsubsection}{\alph{subsubsection})}
47 \newcommand{\subsectionpostlude}{\vspace{1em}}
48
49 \begin{document}
50     \maketitle
51     Untuk memudahkan pembuatan lampiran, maka kontennya di-\textit{generate}
→ menjadi PDF menggunakan LaTeX (TexStudio sebagai IDE-nya) dengan
→ memanfaatkan \textit{package} pdffpages dan minted. LaTeX adalah sebuah
→ bahasa pemrograman yang difokuskan untuk membuat sebuah dokumen dan banyak
→ digunakan untuk membuat thesis maupun jurnal ilmiah. \textit{Source code}
→ dalam LaTeX untuk membuat lampiran ini ditunjukkan melalui Kode
→ \ref{code:latex}.
52     \begin{multipageListing}
```

```

53          \inputminted{latex}{generatePDF.tex}
54          \captionof{listing}{\textit{Source code} LaTeX untuk membuat
55          \hookrightarrow lampiran\label{code:latex}}
56      \end{multipageListing}

57      \includepdf[pages={1-3}, scale=0.75, nup=1x3, pagecommand=\section{PRODUCT
58      \hookrightarrow MANUAL}]{codes/product_manual.pdf}
59      \includepdf[pages={4-}, scale=0.75, nup=1x3]{codes/product_manual.pdf}

60      \includepdf[pages={11-12}, scale=0.8, nup=1x2, offset=0 0.9cm,
61      \hookrightarrow pagecommand=\section{FRONT END}]{codes/ppt.pdf}
62      \includepdf[pages={13-18}, scale=0.8, nup=1x2, offset=0 0.5cm]{codes/ppt.pdf}

63      \section{DATA MODEL}
64          \subsection{USER MODEL}
65          \begin{multipageListing}
66              \inputminted{kotlin}{codes/UserModel.kt}
67              \captionof{listing}{\textit{Data class} untuk \textit{User}}
68          \end{multipageListing}

69          \subsection{NOTE MODEL}
70          \begin{multipageListing}
71              \inputminted{kotlin}{codes/NoteModel.kt}
72              \captionof{listing}{\textit{Data class} untuk \textit{Note}}
73          \end{multipageListing}

74          \subsection{FARM MODEL}
75          \begin{multipageListing}
76              \inputminted{kotlin}{codes/FarmModel.kt}
77              \captionof{listing}{\textit{Data class} untuk \textit{Farm}}
78          \end{multipageListing}

79          \subsection{KIT MODEL}
80          \begin{multipageListing}
81              \inputminted{kotlin}{codes/KitModel.kt}
82              \captionof{listing}{\textit{Data class} untuk
83              \hookrightarrow \textit{Hydroponic Kit}}
84          \end{multipageListing}

85          \subsection{DATA MONITORING MODEL}
86          \begin{multipageListing}
87              \inputminted{kotlin}{codes/DataMonitoringModel.kt}
88              \captionof{listing}{\textit{Data class} untuk \textit{Data
89              \hookrightarrow Monitoring}}
90          \end{multipageListing}

91          \subsection{CROPS MODEL}
92          \begin{multipageListing}
93              \inputminted{kotlin}{codes/CropsModel.kt}
94              \captionof{listing}{\textit{Data class} untuk \textit{Crops}}
95          \end{multipageListing}

96          \subsection{PLANT MODEL}
97          \begin{multipageListing}
98              \inputminted{kotlin}{codes/PlantModel.kt}
99              \captionof{listing}{\textit{Data class} untuk \textit{Plant}}
100         \end{multipageListing}

101         \subsection{FEEDBACK MODEL}
102         \begin{multipageListing}
103             \inputminted{kotlin}{codes/FeedbackModel.kt}
104             \captionof{listing}{\textit{Data class} untuk \textit{Feedback}}
105         \end{multipageListing}

106         \subsection{SCORE LEVEL MODEL}
107         \begin{multipageListing}
108             \inputminted{kotlin}{codes/ScoreLevel.kt}
109             \captionof{listing}{\textit{Data class} untuk data yang
110             \hookrightarrow memiliki nilai minimum dan maksimum}
111         \end{multipageListing}

```

```

117
118         \subsection{URI FILE EXTENSIONS MODEL}
119         \begin{multipageListing}
120             \inputminted{kotlin}{codes/UriFileExtensions.kt}
121             \captionof{listing}{\textit{Data class} untuk yang digunakan
122             \rightarrow untuk mengelola \textit{attach} gambar}
123         \end{multipageListing}
124
125         \section{VIEW MODEL}
126         \subsection{SPLASH SCREEN VIEW MODEL}
127         \begin{multipageListing}
128             \inputminted{kotlin}{codes/SplashScreenViewModel.kt}
129             \captionof{listing}{\textit{View Model} untuk \textit{Splash
130             \rightarrow Screen Activity}}
131         \end{multipageListing}
132
133         \subsection{SIGN IN VIEW MODEL}
134         \begin{multipageListing}
135             \inputminted{kotlin}{codes/SignInViewModel.kt}
136             \captionof{listing}{\textit{View Model} untuk \textit{Sign In
137             \rightarrow Activity}}
138         \end{multipageListing}
139
140         \subsection{SIGN UP VIEW MODEL}
141         \begin{multipageListing}
142             \inputminted{kotlin}{codes/SignUpViewModel.kt}
143             \captionof{listing}{\textit{View Model} untuk \textit{Sign Up
144             \rightarrow Activity}}
145         \end{multipageListing}
146
147         \subsection{FORGET PASSWORD VIEW MODEL}
148         \begin{multipageListing}
149             \inputminted{kotlin}{codes/ForgetPasswordViewModel.kt}
150             \captionof{listing}{\textit{View Model} untuk \textit{Forget
151             \rightarrow Password Activity}}
152         \end{multipageListing}
153
154         \subsection{CREATE PROFILE VIEW MODEL}
155         \begin{multipageListing}
156             \inputminted{kotlin}{codes/CreateProfileViewModel.kt}
157             \captionof{listing}{\textit{View Model} untuk \textit{Create
158             \rightarrow Profile Activity} beserta \textit{Create User} dan
159             \textit{Create Farm} \textit{fragments}. Termasuk juga
160             \textit{Edit Profile User} dan \textit{Edit Profile
161             \rightarrow Farm} \textit{activities}}
162         \end{multipageListing}
163
164         \subsection{MAIN VIEW MODEL}
165         \begin{multipageListing}
166             \inputminted{kotlin}{codes/MainViewModel.kt}
167             \captionof{listing}{\textit{View Model} untuk \textit{Main
168             \rightarrow Activity} beserta \textit{Main Home}, \textit{Main Posts},
169             \textit{Main Add}, \textit{Main Notes}, dan \textit{Main
170             \rightarrow Profile} \textit{fragments})}
171         \end{multipageListing}
172
173         \subsection{ADD CROPS VIEW MODEL}
174         \begin{multipageListing}
175             \inputminted{kotlin}{codes/AddCropsViewModel.kt}
176             \captionof{listing}{\textit{View Model} untuk \textit{Add Crops
177             \rightarrow Activity}}
178         \end{multipageListing}
179
180         \subsection{ADD DATA MONITORING VIEW MODEL}
181         \begin{multipageListing}
182             \inputminted{kotlin}{codes/AddDataMonitoringViewModel.kt}
183             \captionof{listing}{\textit{View Model} untuk \textit{Add Data
184             \rightarrow Monitoring Activity}}
185         \end{multipageListing}

```

```

173   \subsection{ADD KIT VIEW MODEL}
174   \begin{multipageListing}
175     \inputminted{kotlin}{codes/AddKitViewModel.kt}
176     \captionof{listing}{\textit{View Model} untuk \textit{Add Kit}
177       \hookrightarrow Activity}}
178   \end{multipageListing}

179   \subsection{ADD NOTE VIEW MODEL}
180   \begin{multipageListing}
181     \inputminted{kotlin}{codes/AddNoteViewModel.kt}
182     \captionof{listing}{\textit{View Model} untuk \textit{Add Note}
183       \hookrightarrow Activity}}
184   \end{multipageListing}

185   \subsection{ADD PLANT VIEW MODEL}
186   \begin{multipageListing}
187     \inputminted{kotlin}{codes/AddPlantViewModel.kt}
188     \captionof{listing}{\textit{View Model} untuk \textit{Add Plant}
189       \hookrightarrow Activity}}
190   \end{multipageListing}

191   \subsection{PROFILE KIT VIEW MODEL}
192   \begin{multipageListing}
193     \inputminted{kotlin}{codes/ProfileKitViewModel.kt}
194     \captionof{listing}{\textit{View Model} untuk \textit{Profile
195       Kit Activity} beserta \textit{Kit Overview}, \textit{Kit
196       Monitoring}, dan \textit{Kit Crops} \textit{fragments}}}
197   \end{multipageListing}

198   \subsection{PROFILE USER VIEW MODEL}
199   \begin{multipageListing}
200     \inputminted{kotlin}{codes/ProfileUserViewModel.kt}
201     \captionof{listing}{\textit{View Model} untuk \textit{Profile
202       User Activity}}}
203   \end{multipageListing}

204   \subsection{SEARCH VIEW MODEL}
205   \begin{multipageListing}
206     \inputminted{kotlin}{codes/SearchViewModel.kt}
207     \captionof{listing}{\textit{View Model} untuk \textit{Search
208       Activity}}}
209   \end{multipageListing}

210   \subsection{FEEDBACK VIEW MODEL}
211   \begin{multipageListing}
212     \inputminted{kotlin}{codes/FeedbackViewModel.kt}
213     \captionof{listing}{\textit{View Model} untuk \textit{Feedback
214       Activity}}}
215   \end{multipageListing}

216   \subsection{SHOW RECYCLER VIEW MODEL}
217   \begin{multipageListing}
218     \inputminted{kotlin}{codes>ShowRecyclerViewModel.kt}
219     \captionof{listing}{\textit{View Model} untuk \textit{Show
220       Recycler Activity}}}
221   \end{multipageListing}

222   \section{VIEW LAYER (ACTIVITY AND FRAGMENT)}
223     \subsection{SPLASH SCREEN ACTIVITY}
224     \begin{multipageListing}
225       \inputminted{kotlin}{codes/SplashScreenActivity.kt}
226       \captionof{listing}{\textit{Source Code} untuk \textit{Splash
227         Screen Activity}}}
228     \end{multipageListing}

229     \subsection{SIGN IN ACTIVITY}
230     \begin{multipageListing}
231       \inputminted{kotlin}{codes/SignInActivity.kt}
232       \captionof{listing}{\textit{Source Code} untuk \textit{Sign In
233         Activity}}}

```

```

232     \end{multipageListing}

233
234     \subsection{SIGN UP ACTIVITY}
235     \begin{multipageListing}
236         \inputminted{kotlin}{codes/SignUpActivity.kt}
237         \captionof{listing}{\textit{Source Code} untuk \textit{Sign Up
238             Activity}}
239     \end{multipageListing}

240     \subsection{FORGET PASSWORD ACTIVITY}
241     \begin{multipageListing}
242         \inputminted{kotlin}{codes/ForgetPasswordActivity.kt}
243         \captionof{listing}{\textit{Source Code} untuk \textit{Forget
244             Password Activity}}
245     \end{multipageListing}

246     \subsection{CREATE PROFILE ACTIVITY}
247     \begin{multipageListing}
248         \inputminted{kotlin}{codes/CreateProfileActivity.kt}
249         \captionof{listing}{\textit{Source Code} untuk \textit{Create
250             Profile Activity} beserta \textit{fragments} terkait}
251     \end{multipageListing}

252         \subsubsection{CREATE USER FRAGMENT}
253         \begin{multipageListing}
254             \inputminted{kotlin}{codes/CreateUserFragment.kt}
255             \captionof{listing}{\textit{Source Code} untuk
256                 \textit{Create User Fragment}}
257         \end{multipageListing}

258         \subsubsection{CREATE FARM FRAGMENT}
259         \begin{multipageListing}
260             \inputminted{kotlin}{codes/CreateFarmFragment.kt}
261             \captionof{listing}{\textit{Source Code} untuk
262                 \textit{Create Farm Fragment}}
263         \end{multipageListing}

264         \subsection{MAIN ACTIVITY}
265         \begin{multipageListing}
266             \inputminted{kotlin}{codes/MainActivity.kt}
267             \captionof{listing}{\textit{Source Code} untuk \textit{Main
268                 Activity} beserta \textit{fragments} terkait}
269         \end{multipageListing}

270             \subsubsection{MAIN HOME FRAGMENT}
271             \begin{multipageListing}
272                 \inputminted{kotlin}{codes/MainHomeFragment.kt}
273                 \captionof{listing}{\textit{Source Code} untuk
274                     \textit{Main Home Fragment}}
275             \end{multipageListing}

276             \subsubsection{MAIN POSTS FRAGMENT}
277             \begin{multipageListing}
278                 \inputminted{kotlin}{codes/MainPostsFragment.kt}
279                 \captionof{listing}{\textit{Source Code} untuk
280                     \textit{Main Posts Fragment}}
281             \end{multipageListing}

282             \subsubsection{MAIN ADD FRAGMENT}
283             \begin{multipageListing}
284                 \inputminted{kotlin}{codes/MainAddFragment.kt}
285                 \captionof{listing}{\textit{Source Code} untuk
286                     \textit{Main Add Fragment}}
287             \end{multipageListing}

288             \subsubsection{MAIN NOTES FRAGMENT}
289             \begin{multipageListing}
290                 \inputminted{kotlin}{codes/MainNotesFragment.kt}
291                 \captionof{listing}{\textit{Source Code} untuk
292                     \textit{Main Notes Fragment}}

```

```

292     \end{multipageListing}

293
294     \subsubsection{MAIN PROFILE FRAGMENT}
295     \begin{multipageListing}
296         \inputminted{kotlin}{codes/MainProfileFragment.kt}
297         \captionof{listing}{\textit{Source Code} untuk
298             \textit{Main Profile Fragment}}
299     \end{multipageListing}

300     \subsection{ADD CROPS ACTIVITY}
301     \begin{multipageListing}
302         \inputminted{kotlin}{codes/AddCropsActivity.kt}
303         \captionof{listing}{\textit{Source Code} untuk \textit{Add
304             Crops Activity}}
305     \end{multipageListing}

306     \subsection{ADD DATA MONITORING ACTIVITY}
307     \begin{multipageListing}
308         \inputminted{kotlin}{codes/AddDataMonitoringActivity.kt}
309         \captionof{listing}{\textit{Source Code} untuk \textit{Add Data
310             Monitoring Activity}}
311     \end{multipageListing}

312     \subsection{ADD KIT ACTIVITY}
313     \begin{multipageListing}
314         \inputminted{kotlin}{codes/AddKitActivity.kt}
315         \captionof{listing}{\textit{Source Code} untuk \textit{Add Kit
316             Activity}}
317     \end{multipageListing}

318     \subsection{ADD NOTE ACTIVITY}
319     \begin{multipageListing}
320         \inputminted{kotlin}{codes/AddNoteActivity.kt}
321         \captionof{listing}{\textit{Source Code} untuk \textit{Add Note
322             Activity}}
323     \end{multipageListing}

324     \subsection{ADD PLANT ACTIVITY}
325     \begin{multipageListing}
326         \inputminted{kotlin}{codes/AddPlantActivity.kt}
327         \captionof{listing}{\textit{Source Code} untuk \textit{Add
328             Plant Activity}}
329     \end{multipageListing}

330     \subsection{PROFILE KIT ACTIVITY}
331     \begin{multipageListing}
332         \inputminted{kotlin}{codes/ProfileKitActivity.kt}
333         \captionof{listing}{\textit{Source Code} untuk \textit{Profile
334             Kit Activity} beserta \textit{fragments} terkait}
335     \end{multipageListing}

336         \subsubsection{KIT OVERVIEW FRAGMENT}
337         \begin{multipageListing}
338             \inputminted{kotlin}{codes/KitOverviewFragment.kt}
339             \captionof{listing}{\textit{Source Code} untuk
340                 \textit{KitOverview Fragment}}
341         \end{multipageListing}

342         \subsubsection{KIT MONITORING FRAGMENT}
343         \begin{multipageListing}
344             \inputminted{kotlin}{codes/KitMonitoringFragment.kt}
345             \captionof{listing}{\textit{Source Code} untuk
346                 \textit{KitMonitoring Fragment}}
347         \end{multipageListing}

348         \subsubsection{KIT CROPS FRAGMENT}
349         \begin{multipageListing}
350             \inputminted{kotlin}{codes/KitCropsFragment.kt}
351             \captionof{listing}{\textit{Source Code} untuk
352                 \textit{Kit Crops Fragment}}

```

```

352           \end{multipageListing}

353
354 \subsection{PROFILE USER ACTIVITY}
355 \begin{multipageListing}
356     \inputminted{kotlin}{codes/ProfileUserActivity.kt}
357     \captionof{listing}{\textit{Source Code} untuk \textit{Profile
358         \rightarrow User Activity}}
359 \end{multipageListing}

360 \subsection{EDIT PROFILE FARM ACTIVITY}
361 \begin{multipageListing}
362     \inputminted{kotlin}{codes/EditProfileFarmActivity.kt}
363     \captionof{listing}{\textit{Source Code} untuk \textit{Edit
364         \rightarrow Profile Farm Activity}}
365 \end{multipageListing}

366 \subsection{EDIT PROFILE USER ACTIVITY}
367 \begin{multipageListing}
368     \inputminted{kotlin}{codes/EditProfileUserActivity.kt}
369     \captionof{listing}{\textit{Source Code} untuk \textit{Edit
370         \rightarrow Profile User Activity}}
371 \end{multipageListing}

372 \subsection{SEARCH MODEL}
373 \begin{multipageListing}
374     \inputminted{kotlin}{codes/SearchActivity.kt}
375     \captionof{listing}{\textit{Source Code} untuk \textit{Search
376         \rightarrow Activity}}
377 \end{multipageListing}

378 \subsection{FEEDBACK ACTIVITY}
379 \begin{multipageListing}
380     \inputminted{kotlin}{codes/FeedbackActivity.kt}
381     \captionof{listing}{\textit{Source Code} untuk \textit{Feedback
382         \rightarrow Activity}}
383 \end{multipageListing}

384 \subsection{SHOW RECYCLER ACTIVITY}
385 \begin{multipageListing}
386     \inputminted{kotlin}{codes>ShowRecyclerActivity.kt}
387     \captionof{listing}{\textit{Source Code} untuk \textit{Show
388         \rightarrow Recycler Activity}}
389 \end{multipageListing}

390 \subsection{SHOW PICTURE ACTIVITY}
391 \begin{multipageListing}
392     \inputminted{kotlin}{codes>ShowPictureActivity.kt}
393     \captionof{listing}{\textit{Source Code} untuk \textit{Show
394         \rightarrow Picture Activity}}
395 \end{multipageListing}

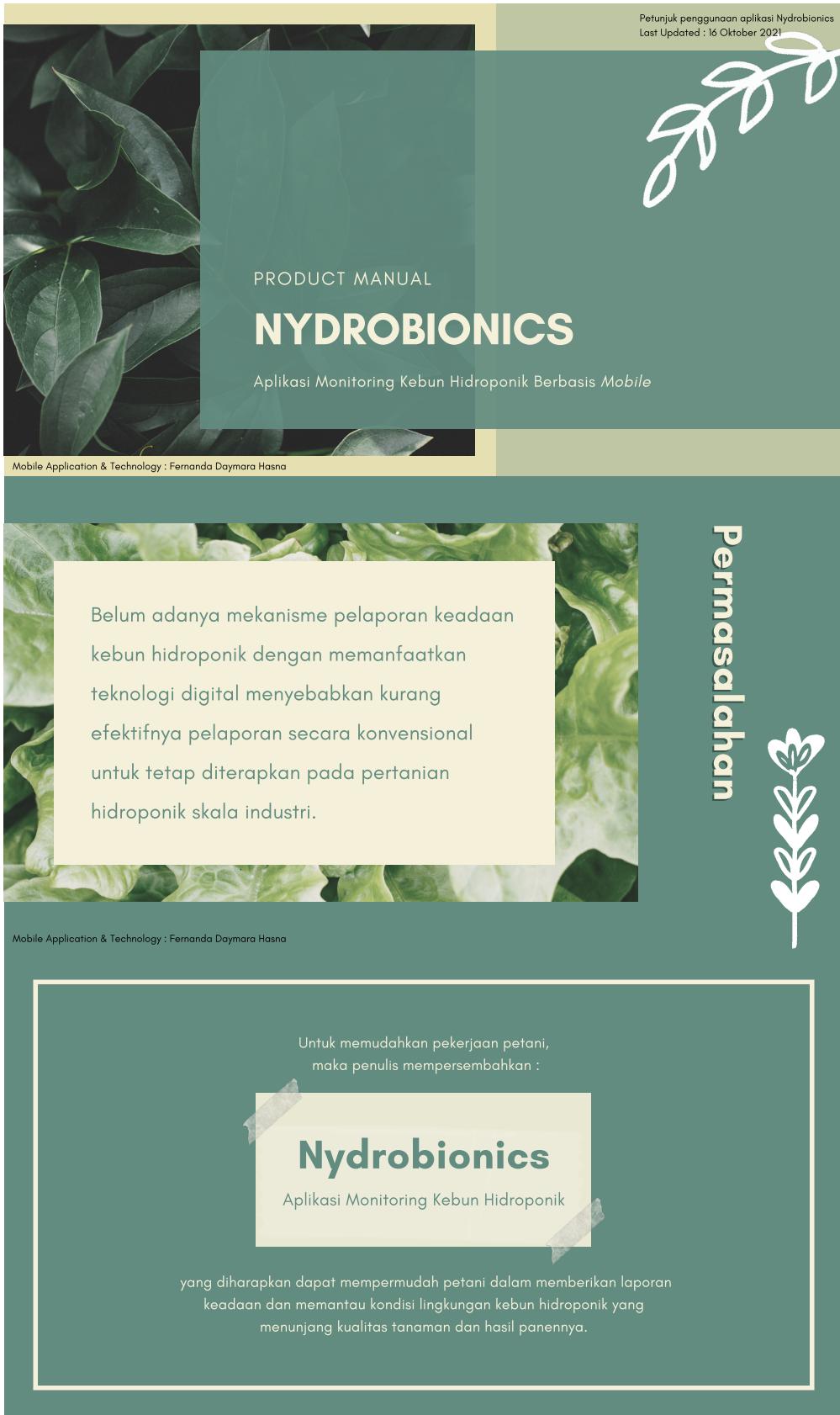
396 \subsection{ABOUT ME ACTIVITY}
397 \begin{multipageListing}
398     \inputminted{kotlin}{codes/AboutMeActivity.kt}
399     \captionof{listing}{\textit{Source Code} untuk \textit{About Me
400         \rightarrow Activity}}
401 \end{multipageListing}

401 \end{document}

```

Kode 1: *Source code* LaTeX untuk membuat lampiran

A. PRODUCT MANUAL



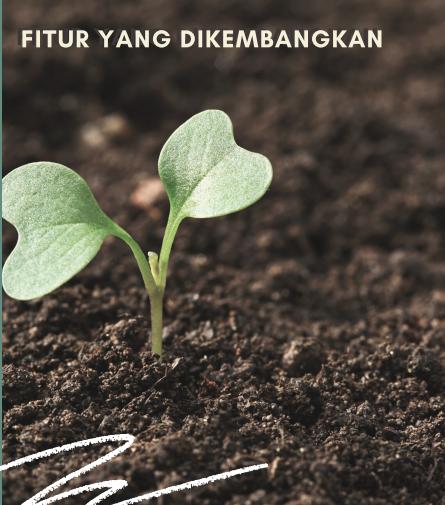


PENDAHULUAN

Hydrobionics adalah *mobile application* berbasis Android untuk monitoring kebun hidroponik agar memudahkan sistem pelaporan kondisi kebun dan meningkatkan akurasi data saat dicek dan dilaporkan.

Aplikasi ini memungkinkan pengguna untuk memantau keadaan kebun hidroponik dengan tujuan dapat mengatur konsistensi variabel-variabel yang memengaruhi kualitas hasil panen.

Mobile Application & Technology : Fernanda Daymara Hasna



FITUR YANG DIKEMBANGKAN

HYDROPONIC MONITORING
Pemantauan keadaan kebun seperti kondisi suhu, kelembaban udara , tingkat keasaman air, kejernihan air, kapasitas tanki air dan pupuk cair

PLANT NYDRODB
Database tanaman seperti nama tanaman, nilai ideal (suhu, kelembaban udara, dan pH air), dan estimasi waktu yang dibutuhkan untuk siap panen.

SCHEDULE AND NOTES
Database catatan dan jadwal untuk petani hidroponik

Mobile Application & Technology : Fernanda Daymara Hasna

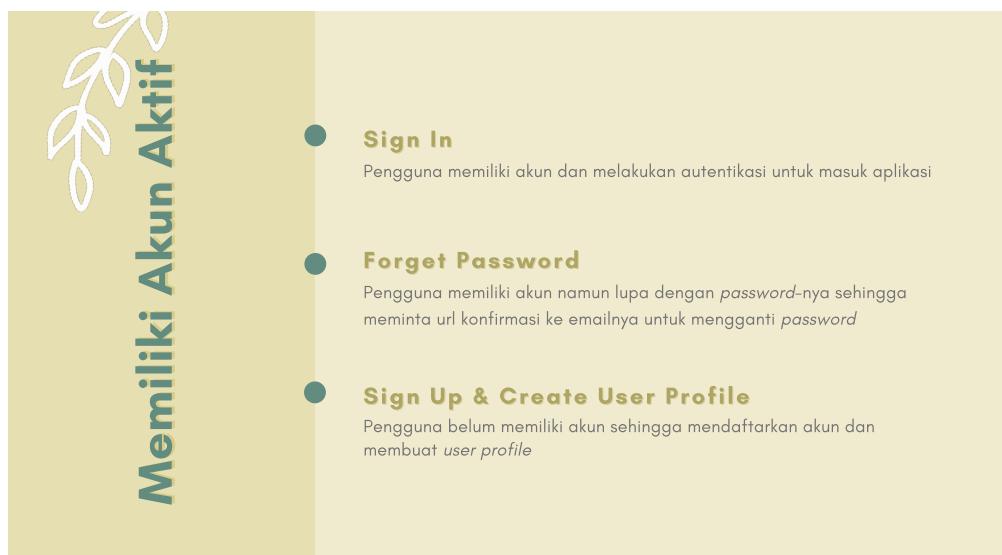


Fitur Utama App.

Untuk menggunakan Hydrobionics, pengguna diharuskan untuk :

- 1 Memiliki akun Aktif**
Terdafat sebagai salah satu pengguna Hydrobionics dan akun terhubung dengan aplikasi
- 2 Menjadi anggota kebun**
Terdafat sebagai Owner maupun Staff dari sebuah kebun yang terdaftar
- 3 Mendaftarkan kit hidroponik**
Terdapat seminimalnya 1 kit hidroponik untuk kebun yang terhubung dengan akun pengguna

Mobile Application & Technology : Fernanda Daymara Hasna



Mobile Application & Technology : Fernanda Daymara Hasna

SIGN IN

1. Diisi sesuai dengan email dan password yang terdaftar
2. Jika sudah, tap tombol "Sign In" untuk proses autentikasi

Sign In

Masuk ke aplikasi dengan autentikasi email dan password yang telah didaftarkan sebelumnya. Jika tidak sesuai, maka pengguna gagal masuk ke aplikasi.

FORGET PASSWORD

Provide your account's email for which you want to reset your password:

1. Diisi sesuai dengan email dari akun yang terdaftar
2. Jika sudah terisi, tap tombol "Next" untuk pengiriman email konfirmasi pada email tersebut

Memiliki Akun Aktif (1)

Sign Up

Membuat akun pengguna dengan memasukkan email dan password yang digunakan untuk proses autentikasi.

SIGN UP

1. Diisi sesuai email yang ingin didaftarkan
2. Diisi sesuai dengan password yang akan digunakan, namun harus memasukkannya 2x sebagai konfirmasi
3. Jika sudah terisi, tap tombol "SIGN UP" untuk proses pembuatan akun

Memiliki Akun Aktif (2)

Create User Profile

Setelah melakukan *Sign Up*, pengguna membuat *user profile* dengan memasukkan data sesuai dengan kolom yang tersedia.

Memiliki Akun Aktif (3)

1. Diperbolehkan untuk mengunggah gambar foto (tidak wajib diisi)
2. Diwajibkan untuk mengisi seluruh field sesuai dengan data yang bersesuaian
3. Role akan otomatis memilih "Owner"
4. Jika sudah terisi, tap tombol "Create Profile" untuk proses pembuatan akun profil

Menjadi Anggota Kebun

Create Farm Profile

Pengguna yang berperan sebagai Owner akan otomatis diarahkan untuk mendaftarkan *Farm Profile* sehingga sudah termasuk sebagai anggota kebun

Edit Farm Profile

Pengguna yang berperan sebagai Owner dapat mengubah data *Farm*

Mobile Application & Technology : Fernanda Daymara Hasna

Create Farm Profile

Setelah melakukan *Create User Profile*, pengguna yang berperan sebagai Owner membuat *Farm Profile*

Menjadi Anggota Kebun (1)

1. Dilis� sesuai dengan nama, deskripsi, dan lokasi dari kebun
2. Jika sudah terisi, tap tombol "Create Farm" untuk proses pembuatan akun profil kebun

Edit Farm Profile

Saat menggunakan aplikasi, pengguna yang berperan sebagai Owner dapat mengubah data kebun maupun memodifikasi staff-nya, baik menambah maupun mengurangi staff

Muncul ketika tombol "Farm Profile" dipilih untuk melihat detail dan mengubah profil kebun

Bentuk layout dan fungsionalitas tidak jauh berbeda dengan Create Farm Profile, hanya saja ada Farm Staff untuk menampilkan dan mengelola list anggota kebun (Coming Soon)

Dapat dipilih jika ingin keluar dari Farm Profile. Jika ada data yang terubah, maka akan menyimpan data tersebut

Dapat dipilih jika ingin menambahkan akun lain sebagai staff kebun (Coming Soon)

Tombol "Save" hanya bisa dipilih jika ada perubahan dalam data, termasuk foto profil

Menjadi Anggota Kebun (2)

Mendaftarkan Kit Hidroponik

- Add and Edit Kit**
Pengguna yang sudah terdaftar pada kebun dapat menambah maupun memodifikasi kit hidroponik
- Add Data Monitoring & Crops**
Kit yang sudah didaftarkan dapat ditambahkan *data monitoring* maupun *crops*-nya
- Show Kit Profile**
Pengguna yang berperan sebagai Owner dapat mengubah data *Farm* temasuk mengelola anggota *staff*-nya

Mobile Application & Technology : Fernanda Daymara Hasna

Add and Edit Kit

Pengguna dapat menambahkan kit dari kebun yang di-*assign* dengan memasukkan data-data yang bersesuaian. Jika dibutuhkan, pengguna juga dapat memodifikasi data tersebut nantinya.

Muncul ketika menu "Add Kit" dipilih atau saat menampilkan setting kit untuk dilihat dan bisa diubah

Seluruh kolom dan counter harus dilisi sesuai data yang berkaitan. Jika mengubah kit, maka data telah terisi sesuai kit yang dipilih

Terdiri dari 8 counter untuk memberikan nilai ukuran kit (panjang lebar) dan nilai min-max untuk kapasitas air, nutrisi, dan kejernihan air yang masing-masing ada 4 komponen

Tombol menampilkan info

Tombol mengurangi nilai

Menampilkan hasil dari klik tombol maupun slide

Tombol menambahkan nilai

Dapat dipilih jika ingin keluar dari Add/Edit Kit. Jika saat Edit Kit ada data yang terubah, maka akan menyimpan data tersebut

Tombol "Submit" hanya bisa dipilih jika semua data terisi dan memenuhi persyaratan ($\min < \max$)

Mendaftarkan Kit Hidroponik (1)

Add Data Monitoring

Pengguna dapat menambahkan data monitoring pada kit yang telah didaftarkan dengan memasukkan data-data yang bersesuaian

Mendaftarkan Kit Hidropotik (2)

Bentuk layout dan fungsionalitas tidak jauh berbeda dengan Create Farm Profile, hanya saja ada Farm Staff untuk menampilkan dan mengelola list anggota kebun (Coming Soon)

Dapat dipilih jika ingin keluar dari Farm Profile. Jika ada data yang terubah, maka akan menyimpan data tersebut

Dapat dipilih jika ingin menambahkan akun lain sebagai staff kebun (Coming Soon)

Tombol "Save" hanya bisa dipilih jika ada perubahan dalam data, termasuk foto profil

Add Crops

Jika kit tersebut belum terdapat tanaman yang ditanam, maka pengguna dapat menambahkan tanaman melalui tombol "Add Crops". Kemudian diberikan pilihan Select Plant dari database yang tersimpan dan Add New Plant

Mendaftarkan Kit Hidropotik (2)

Muncul ketika menu "Add Data Crops" dipilih, kemudian diberikan 2 pilihan :

1. Select Plant dari tanaman yang sudah ada
2. Add New Plant untuk diarahkan ke Add Plant

Ketika tanaman berhasil dipilih/dibuat, maka data tersebut dikembalikan ke menu ini untuk ditampilkan pada kolom yang bersesuaian

Diklik untuk membuka dropdown dan memilih kit yang ingin ditambah datanya

Tombol "Submit" hanya bisa dipilih jika kit dan tanaman sudah disi

Add and Edit Plant

Pengguna dapat menambahkan data tanaman ke Plant NydroDB dengan memasukkan data yang bersesuaian. Jika pengguna menambahkannya saat Add Data Monitoring atau Add Crops, maka data tanaman tersebut akan langsung ditambahkan sebagai tanaman yang ditanam.

Mendaftarkan Kit Hidropotik (3)

Muncul ketika menu "Add Plant" dipilih atau saat melihat atau mengubah postingan akun

Selain foto, seluruh kolom dan counter harus disesuaikan data yang berkaitan. Jika mengubah tanaman, maka data telah tersimpan tanaman yang dipilih

Terdiri dari 7 counter untuk memberikan nilai estimasi panen dan nilai min-max untuk suhu, kelembaban, dan pH yang masing-masing ada 4 komponen

Tombol menampilkan info

Tombol mengurangi nilai

Menampilkan hasil dari klik tombol maupun slide

Tombol menambahkan nilai

Dapat dipilih jika ingin keluar dari Add/Edit Plant. Jika saat Edit ada data yang terubah, maka akan menyimpan data tersebut

Tombol "Submit" hanya bisa dipilih jika semua data terisi dan memenuhi persyaratan (min < max)

Show Kit Profile

Saat masuk ke aplikasi, pengguna dapat melihat data terbaru dari setiap kit pada kebunnya. Jika tombol "Show Detail" dipilih, maka akan menampilkan data lebih detailnya.



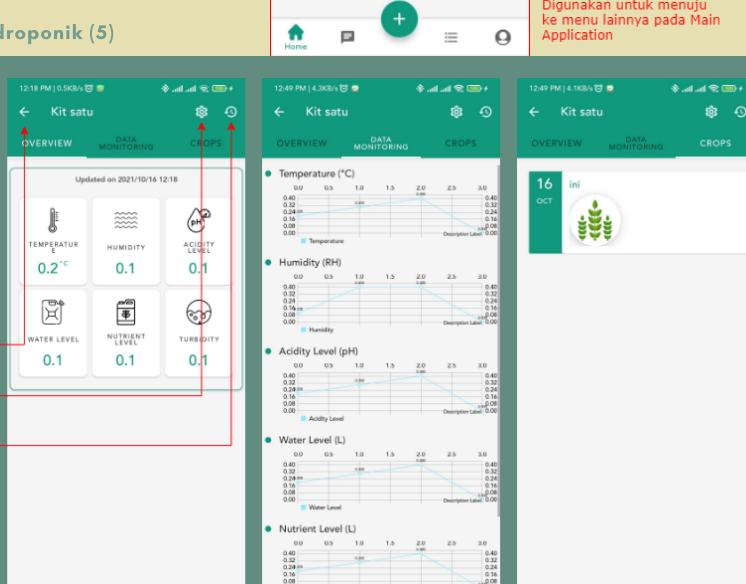
Muncul ketika autentikasi berhasil atau ketika "Home" dipilih
Menampilkan nama kebun yang terhubung dengan akun pengguna
Menampilkan data monitoring terakhir dari setiap kit yang dipilih :
1. Nama
2. Update terakhir
3. Nilai monitoring (suhu, kelembaban, pH, kapasitas air & pupuk, serta kejernihan air)
Dapat mengganti kit yang dilihat dengan tombol Next Previous
Dapat melihat profil kit lebih detail dengan tombol "Show More"
Digunakan untuk menuju ke menu lainnya pada Main Application

Mendaftarkan Kit Hidropotnik (5)

Muncul ketika memiliki "Show More" pada kit yang bersangkutan. Secara default menampilkan bagian "Overview" data monitoring terbaru

Terdapat 3 tab :
1. Overview yaitu data monitoring terakhir
2. Data Monitoring yaitu rekapitulasi monitoring kebun dalam bentuk grafik
3. Crops yaitu rekapitulasi tanaman yang pernah ditanam

Dapat dipilih jika ingin keluar dari Kit Profile
Dapat dipilih jika ingin melihat detail dan mengubah data kit
Dapat dipilih jika ingin menampilkan historis mengenai kit (Coming Soon)

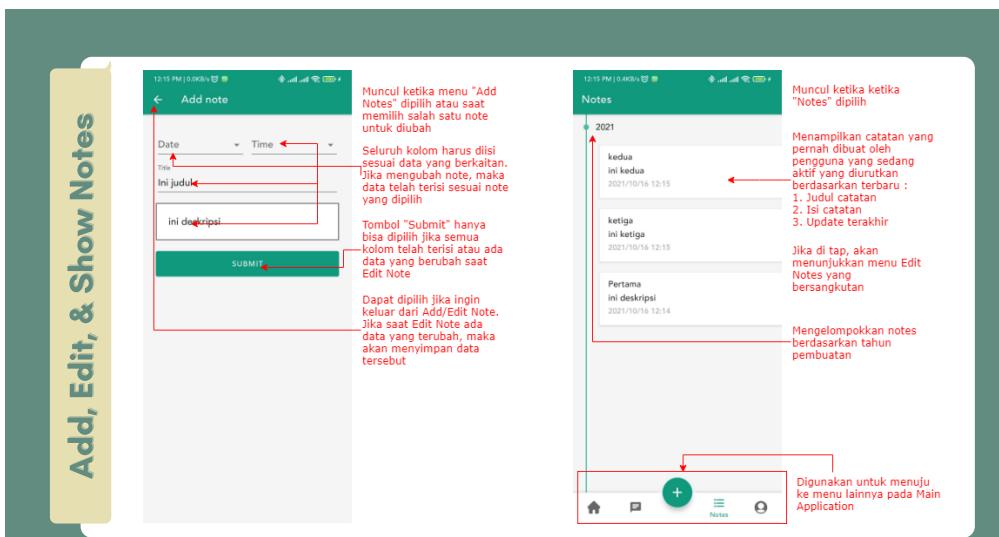


Untuk mendukung fitur utama pada Nydrobionics, disediakan fitur-fitur tambahan seperti :

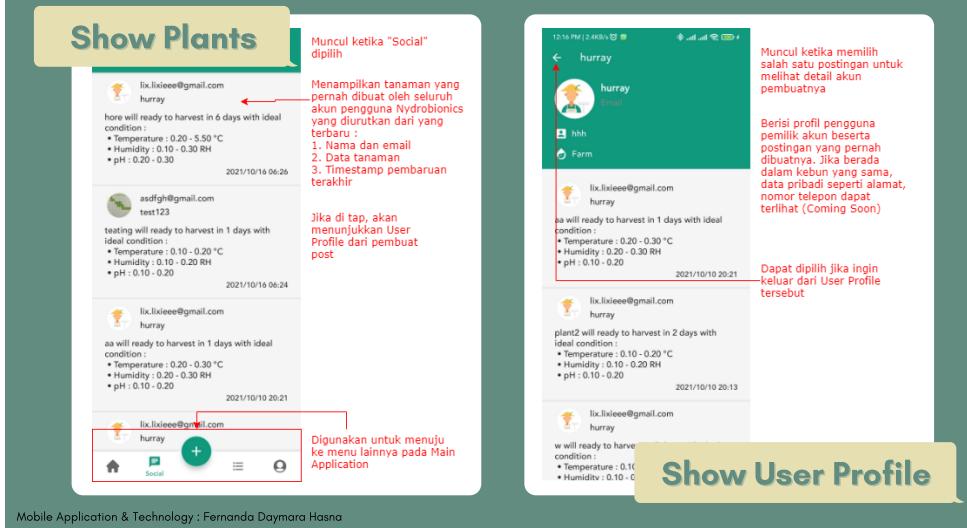
- 1 Add, Edit, and Show Notes**
Menambah, memodifikasi, dan menampilkan catatan pengguna yang ter-sign-in
- 2 Show Plants**
Menampilkan seluruh tanaman yang telah dibuat oleh pengguna lainnya
- 3 Show and Edit User Profile**
Menampilkan data pengguna lain sesuai yang dipilih dan mengedit data pengguna ter-sign-in

Fitur Pendukung App.

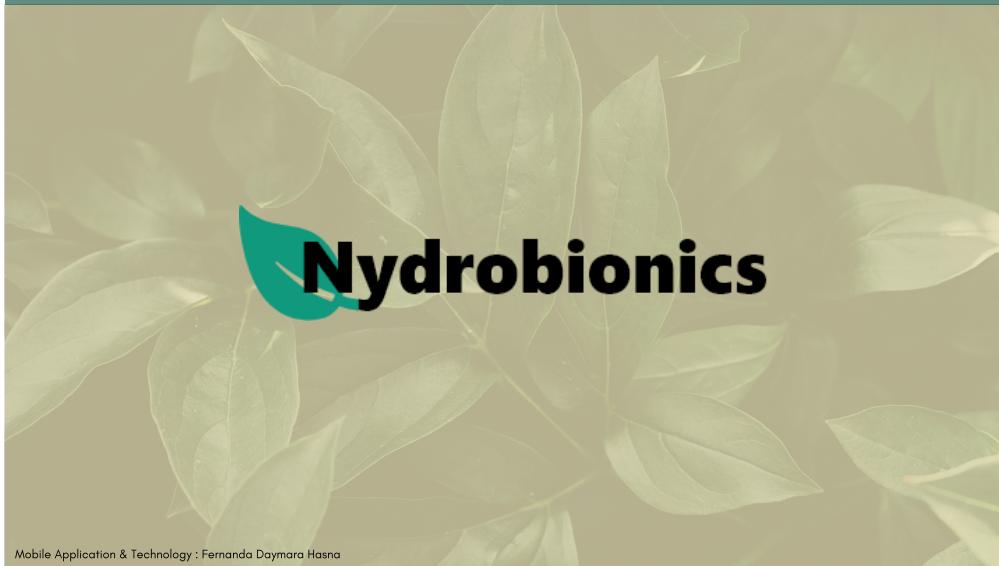




Mobile Application & Technology : Fernanda Daymara Hasna

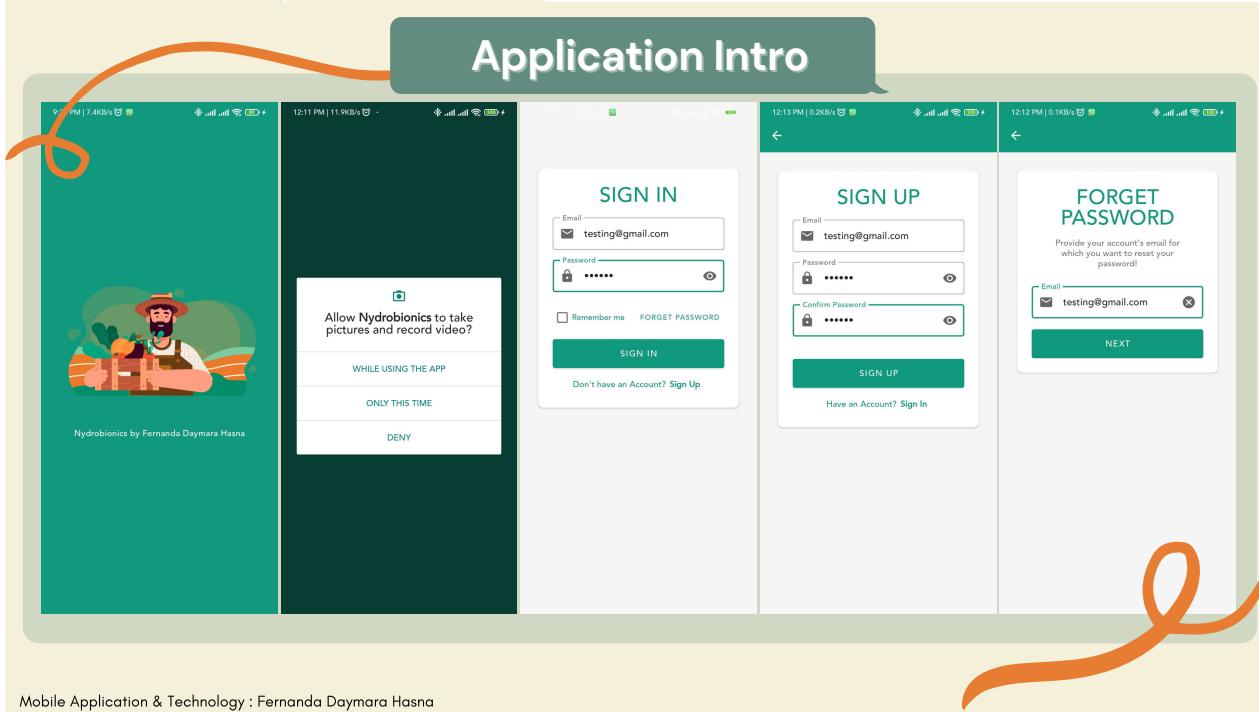
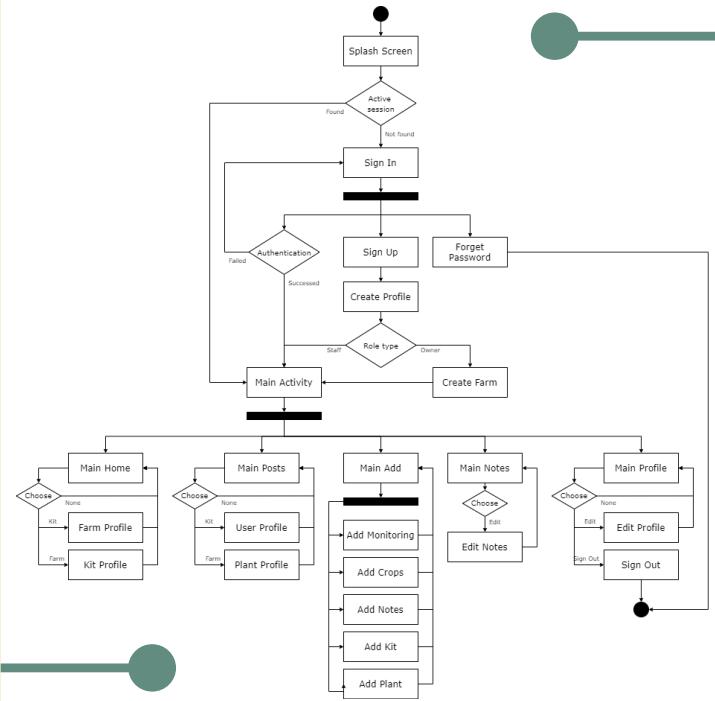


Mobile Application & Technology : Fernanda Daymara Hasna



Mobile Application & Technology : Fernanda Daymara Hasna

B. FRONT END



Edit Profile

Farm Name: Kebun Coba Aja
Farm Description: Ini buat coba coba
Farm Location: bekasi
Farm Staff: testing@gmail.com Testing Account

Create Profile

Create Farm

Farm Name: Kebun Coba
Farm Description: Ini buat coba coba
Farm Location: Bekasi

Profile

Email: testing@gmail.com
Name: Testing Account
Phone Number: 08123456789
Address: Bekasi
Bio: Akun cobacoba aja
Date of Birth: 05 October 202 - Male

Owner & Staff

Owner: Akun cobacoba
Staff: Akun cobacoba

CREATE PROFILE

Mobile Application & Technology : Fernanda Daymara Hasna

Main Application

Home

Kebun Coba Aja

Kit satu
Updated on 2021/10/16 12:18

TEMPERATURE 0.2 °C	HUMIDITY 0.1	ACIDITY LEVEL 0.1
WATER LEVEL 0.1	NUTRIENT LEVEL 0.1	TURBIDITY 0.1

Social

lix.lixeee@gmail.com huray
here will ready to harvest in 6 days with ideal condition :
• Temperature : 0.20 - 5.50 °C
• Humidity : 0.10 - 0.30 RH
• pH : 0.20 - 0.30

asdgh@gmail.com test123
teating will ready to harvest in 1 days with ideal condition :
• Temperature : 0.10 - 0.20 °C
• Humidity : 0.10 - 0.20 RH
• pH : 0.10 - 0.20

lix.lixeee@gmail.com huray
aa will ready to harvest in 1 days with ideal condition :
• Temperature : 0.20 - 0.30 °C
• Humidity : 0.20 - 0.30 RH
• pH : 0.10 - 0.20

lix.lixeee@gmail.com huray
aa will ready to harvest in 1 days with ideal condition :
• Temperature : 0.20 - 0.30 °C
• Humidity : 0.20 - 0.30 RH
• pH : 0.10 - 0.20

Notes

2021
kedua
ini kedua
2021/10/16 12:15

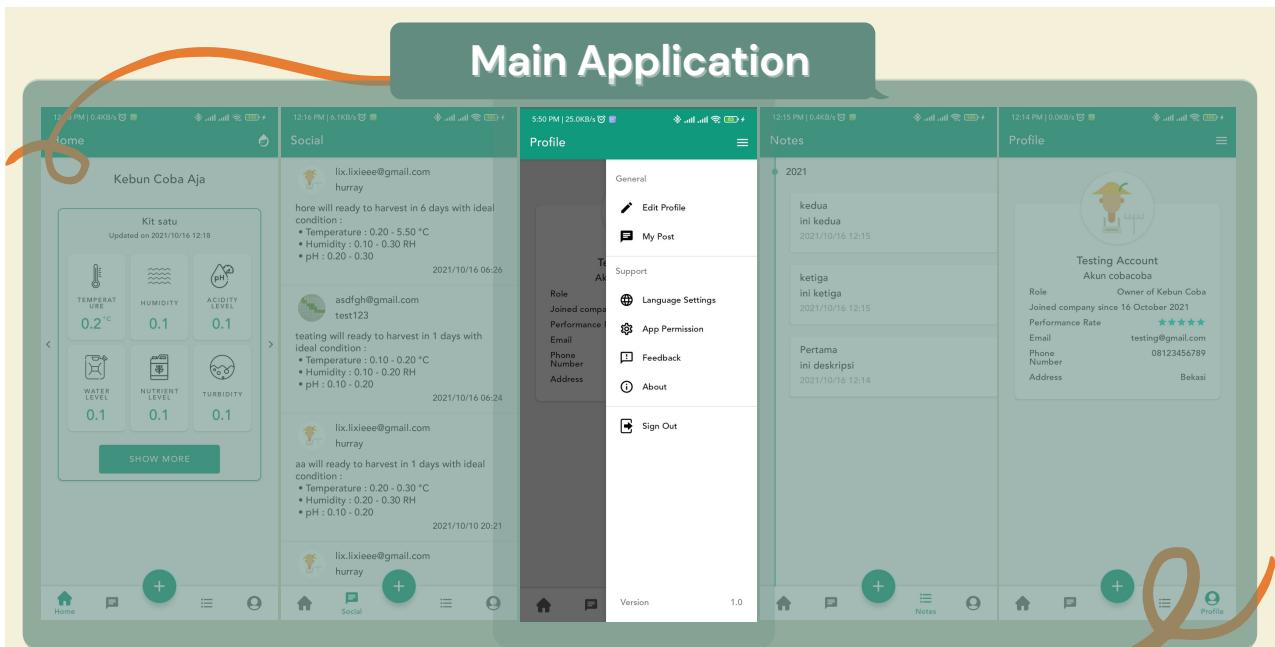
ketiga
ini ketiga
2021/10/16 12:15

Pertama
ini deskripsi
2021/10/16 12:14

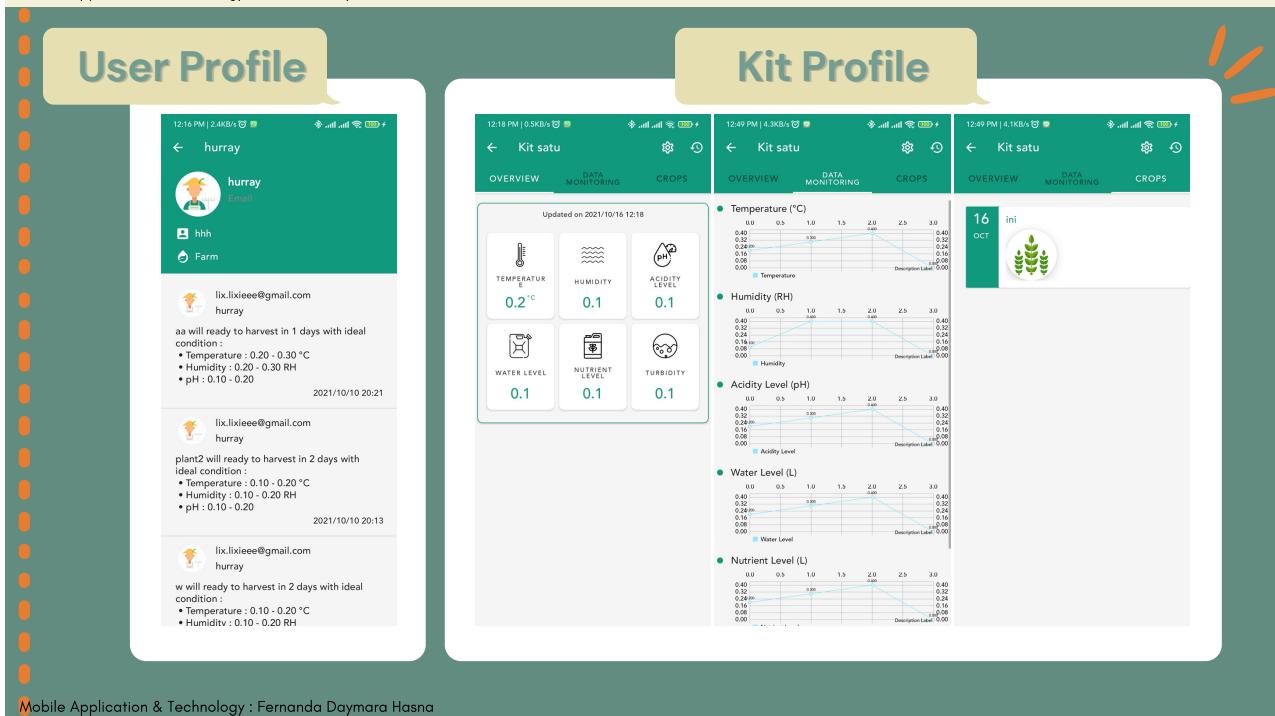
Profile

Testing Account
Akun cobacoba
Role: Owner of Kebun Coba
Joined company since 16 October 2021
Performance Rate: ★★★★
Email: testing@gmail.com
Phone Number: 08123456789
Address: Bekasi

Mobile Application & Technology : Fernanda Daymara Hasna



Mobile Application & Technology : Fernanda Daymara Hasna



Add and Edit Data

Add recent data monitoring

Kebun Coba Aja

Hydroponic Kit

Kit satu

Temperature (°C)

- 0.2 +

Humidity (RH)

- 0.1 +

Acidity Level (pH)

- 0.1 +

Water Level (L)

- 0.1 +

Nutrient Level (L)

- 0.1 +

Turbidity (NTU)

- 0.1 +

ADD NEW CROPS

SUBMIT

Add New Crops

Kebun Coba Aja

Hydroponic Kit

Kit satu

Plant Name

ini

Plant Description

itu

Growth Time (days)

3 days

Estimated Harvest

19 October 2021

Temperature (°C)

Minimum 0.20 °C Maximum 0.30 °C

Humidity (RH)

Minimum 0.30% RH Maximum 0.40% RH

Acidity Level (pH)

Minimum 0.2 Maximum 0.3

SUBMIT

Add note

Kebun Coba Aja

Date

Time

Title

Ini judul

ini deskripsi

SUBMIT

Create New Kit

Kebun Coba Aja

Hydroponic Kit Name

kit satu

Kit Position Description

ini satu

Kit Size

Width - 1 + Length - 2 +

Water Level (L)

Minimum 0.1 Maximum 0.3

Nutrient Level (L)

Minimum 0.1 Maximum 0.3

Turbidity (NTU)

Minimum 1 Maximum 2

SUBMIT

Add New Plant

Kebun Coba Aja

Plant Name

Ini tanaman

Plant Description

ini deskripsinya

Growth Time (days)

- 2 +

Temperature (°C)

Minimum 0.1 Maximum 0.2

Humidity (RH)

Minimum 0.1 Maximum 0.2

Acidity Level (pH)

Minimum 0.1 Maximum 0.2

SUBMIT

Mobile Application & Technology : Fernanda Daymara Hasna

Others

About Me

Fernanda Daymara Hasna
fernanda.daymara.hasna@gmail.com

Biography

As known as Nanda, I was born in Magelang on August 21, 1998. I am the first of three children who have completed education at Computer Engineering FTUI Institut Teknologi Sepuluh Nopember Surabaya. I am interested in the development of mobile applications, especially in the fields of IT Project Management, User-Oriented Product Development, and Design System Analysis. Currently, I am exploring Mobile Programming with Android Studio.

Find Me

- [Email](#)
- [WhatsApp](#)
- [GitHub](#)
- [Project Source](#)

Credit

Splash Screen by Mysaffa

Feedback

Did you like Nydrobonics?

Let us know what you think

★★★★★ Excellent

good!

SUBMIT

Testing Account



Search Plant

Search

asdgh@gmail.com	teating
aa	aa
lix.lixehee@gmail.com	plant2
testing@gmail.com	ini
w	w
lix.lixehee@gmail.com	hore
lix.lixehee@gmail.com	1
lix.lixehee@gmail.com	123
testing@gmail.com	Ini tanaman

History

testing@gmail.com Testing Account

ini will ready to harvest in 3 days with ideal condition :

- Temperature : 0.20 - 0.30 °C
- Humidity : 0.30 - 0.40 RH
- pH : 0.20 - 0.30

2021/10/16 12:19

testing@gmail.com Testing Account

Ini tanaman will ready to harvest in 2 days with ideal condition :

- Temperature : 0.10 - 0.20 °C
- Humidity : 0.10 - 0.20 RH
- pH : 0.10 - 0.20

2021/10/16 12:17

Mobile Application & Technology : Fernanda Daymara Hasna

C. DATA MODEL

1. USER MODEL

```
1  @Parcelize
2  data class UserModel(
3      var name : String? = null,
4      var email: String? =null,
5      var gender : String? = null,
6      var dob : String? = null,
7      var role : String? = null,
8      var bio : String? = null,
9      var uid : String? = null,
10     var performanceRate : Float? = null,
11     var address : String? = null,
12     var phone : String? = null,
13     var joinedSince : String? = null,
14     var photo_url : String? = null,
15     var farmId : String? = null,
16     var timestamp : String? = null
17 ) : Parcelable
```

Kode 2: *Data class* untuk *User*

2. NOTE MODEL

```
1  @Parcelize
2  data class NoteModel(
3      var noteId : String? = null,
4      var title: String? = null,
5      var description : String? = null,
6      var timestamp : String?=null,
7      var date : String? = null,
8      var time : String? = null
9  ) : Parcelable
```

Kode 3: *Data class* untuk *Note*

3. FARM MODEL

```
1  @Parcelize
2  data class FarmModel(
3      var farmId : String? = null,
4      var name : String? = null,
5      var description : String? = null,
6      var location : String? = null,
7      var gps : String? = null,
8      var timestamp : String? = null,
9      var kitModels : ArrayList<KitModel>? = null
10 ) : Parcelable
```

Kode 4: *Data class* untuk *Farm*

4. KIT MODEL

```
1  @Parcelize
2  data class KitModel(
3      var kitId : String? = null,
4      var name : String? = null,
5      var position : String? = null,
6      var length : Int? = null,
7      var width : Int? = null,
8      var waterLv : @RawValue ScoreLevel? = null,
```

```

9     var nutrientLv : @RawValue ScoreLevel? = null,
10    var turbidityLv: @RawValue ScoreLevel? = null,
11    var isPlanted : Boolean? = false,
12    var timestamp : String? = null,
13    var lastMonitoring : DataMonitoringModel? = null,
14    var lastCrops : CropsModel? = null
15 ) : Parcelable

```

Kode 5: *Data class* untuk *Hydroponic Kit*

5. DATA MONITORING MODEL

```

1  @Parcelize
2  data class DataMonitoringModel(
3      var dataId : String? = null,
4      var temperature : Float? = 0f,
5      var humidity : Float? = 0f,
6      var waterTank : Float? = 0f,
7      var nutrientTank : Float? = 0f,
8      var turbidity : Float? = 0f,
9      var ph : Float? = 0f,
10     var userId : String? = null,
11     var cropsId : String? = null,
12     var timestamp : String? = null
13 ) : Parcelable

```

Kode 6: *Data class* untuk *Data Monitoring*

6. CROPS MODEL

```

1  @Parcelize
2  data class CropsModel(
3      var cropsId : String? = null,
4      var plantId : String? = null,
5      var userId : String? = null,
6      var tempLv : @RawValue ScoreLevel? = null,
7      var humidLv : @RawValue ScoreLevel? = null,
8      var phLv : @RawValue ScoreLevel? = null,
9      var timestamp : String? = null,
10     var plantModel: PlantModel? = null
11 ) : Parcelable

```

Kode 7: *Data class* untuk *Crops*

7. PLANT MODEL

```

1  @Parcelize
2  data class PlantModel(
3      var plantId : String? = null,
4      var name : String? = null,
5      var description : String? = null,
6      var photo_url : String? = null,
7      var tempLv : @RawValue ScoreLevel? = null,
8      var humidLv : @RawValue ScoreLevel? = null,
9      var phLv : @RawValue ScoreLevel? = null,
10     var userId : String? = null,
11     var growthTime : Int? = 0,
12     var timestamp : String? = null
13 ) : Parcelable

```

Kode 8: *Data class* untuk *Plant*

8. FEEDBACK MODEL

```
1  @Parcelize
2  data class FeedbackModel(
3      var feedbackId : String? = null,
4      var userId : String? = null,
5      var content : String? = null,
6      var rating : Float? = null,
7      var timestamp: String? = null
8  ) : Parcelable
```

Kode 9: *Data class* untuk *Feedback*

9. SCORE LEVEL MODEL

```
1  @Parcelize
2  data class ScoreLevel(
3      var min : Float? = 0f,
4      var max : Float? = 0f,
5      var score : Float? = 0f
6  ) : Parcelable {
7      companion object {
8          fun getLevelModel(string: String?) : ScoreLevel? {
9              return try {
10                  val output = Gson().fromJson<ScoreLevel>(string, ScoreLevel::class.java)
11                  ScoreLevel(output.min, output.max)
12              } catch (e:Exception){
13                  Log.e(TAG, "Error converting $TAG", e)
14                  null
15              }
16          }
17      }
18      private const val TAG = "ScoreLevel"
19  }
```

Kode 10: *Data class* untuk data yang memiliki nilai minimum dan maksimum

10. URI FILE EXTENSIONS MODEL

```
1  data class UriFileExtensions(
2      var uri : Uri,
3      var fileExtensions : String
4  )
```

Kode 11: *Data class* untuk yang digunakan untuk mengelola *attach* gambar

D. VIEW MODEL

1. SPLASH SCREEN VIEW MODEL

```
1  class SplashScreenViewModel : ViewModel() {
2      private var auth : FirebaseAuth = Firebase.auth
3      private var firestore : FirebaseFirestore = Firebase.firestore
4      var currentUserModel : MutableLiveData<UserModel?> = MutableLiveData(null)
5      var currentFarmModel : MutableLiveData<FarmModel?> = MutableLiveData(null)
6      var isCurrentUserExist : MutableLiveData<Boolean?> = MutableLiveData(null)
7      var isCurrentFarmExist : MutableLiveData<Boolean?> = MutableLiveData(null)
8
9      companion object {
10          const val TAG = "splashScreen"
11      }
12  }
```

```

13     fun getCurrentData(){
14         try{
15             auth.uid?.let {
16                 firestore.collection("users").document(auth.uid!!)
17                     .get().addOnSuccessListener {
18                         if(it.exists()){
19                             currentUserModel.value = it.toUserModel()
20                             isCurrentUserExist.value = true
21                             currentUserModel.value?.farmId?.let {
22                                 firestore.collection("farms").document(it)
23                                     .get().addOnSuccessListener {
24                                         if(it.exists()){
25                                             currentFarmModel.value = it.toFarmModel()
26                                             Log.i(
27                                                 TAG,
28                                                 "getCurrentData:
29                                     → ${currentUserModel.value}\n" +
30                                     "${currentFarmModel.value}"
31                                         )
32                                         isCurrentFarmExist.value = true
33                                     } else {
34                                         isCurrentFarmExist.value = false
35                                         }
36                                     }
37                                 isCurrentFarmExist.value = false
38                             }
39                         } else {
40                             isCurrentUserExist.value = false
41                         }
42                     }
43                 }
44             } ?: kotlin.run {
45                 isCurrentUserExist.value = false
46             }
47         } catch (e:Exception){
48             Log.e(TAG, "getCurrentData ${auth.uid}", e)
49         }
50     }
51 }

```

Kode 12: *View Model* untuk *Splash Screen Activity*

2. SIGN IN VIEW MODEL

```

1  class SignInViewModel : ViewModel() {
2     private var auth = Firebase.auth
3     private var isRememberChecked : MutableLiveData<Boolean> = MutableLiveData(false)
4     private var isNotEmpty : MutableLiveData<Boolean> = MutableLiveData(false)
5     private var localDataModel : MutableLiveData<DatabaseModel> =
6         MutableLiveData(DatabaseModel())
7     var isUserSignIn : MutableLiveData<Boolean> = MutableLiveData(false)
8     var signInError : MutableLiveData<String?> = MutableLiveData(null)
9
10    fun checkNotEmpty(boolean: Boolean) : MutableLiveData<Boolean> {
11        isNotEmpty.value = boolean
12        return isNotEmpty
13    }
14
15    fun getDataModel() : DatabaseModel = localDataModel.value!!
16
17    fun setRememberChecked(boolean: Boolean){
18        isRememberChecked.value = boolean
19    }
20
21    fun signIn(email:String, password:String){
22        auth.signInWithEmailAndPassword(email, password).addOnCompleteListener {
23            if(it.isSuccessful){

```

```

24         localDataModel.value = DatabaseModel(
25             uid = auth.uid,
26             email = email,
27             remember = isRememberChecked.value!!
28         )
29         isUserSignIn.value = true
30     } else {
31         signInError.value = it.exception.toString()
32         isUserSignIn.value = false
33     }
34 }
35 }
36

```

Kode 13: *View Model* untuk *Sign In Activity*

3. SIGN UP VIEW MODEL

```

1 class SignUpViewModel : ViewModel() {
2     private var auth = Firebase.auth
3     private var isNotEmpty : MutableLiveData<Boolean> = MutableLiveData(false)
4     var isUserSignUp : MutableLiveData<Boolean> = MutableLiveData(false)
5     var signUpError : MutableLiveData<String?> = MutableLiveData(null)
6
7     fun checkNotEmpty(boolean: Boolean) : MutableLiveData<Boolean> {
8         isNotEmpty.value = boolean
9         return isNotEmpty
10    }
11
12    fun signUp(email:String, password:String){
13        signUpError.value = null
14        auth.createUserWithEmailAndPassword(email, password).addOnCompleteListener {
15            if(it.isSuccessful){
16                isUserSignUp.value = true
17            } else {
18                signUpError.value = it.exception.toString()
19                isUserSignUp.value = false
20            }
21        }
22    }
23

```

Kode 14: *View Model* untuk *Sign Up Activity*

4. FORGET PASSWORD VIEW MODEL

```

1 class ForgetPasswordViewModel : ViewModel() {
2     private var auth = Firebase.auth
3     private var isEmailNotEmpty : MutableLiveData<Boolean> = MutableLiveData(false)
4     var isUserForgotPassword : MutableLiveData<Boolean> = MutableLiveData(null)
5     var forgetPasswordError : MutableLiveData<String> = MutableLiveData("")
6
7     fun checkNotEmpty(boolean: Boolean) : MutableLiveData<Boolean> {
8         isEmailNotEmpty.value = boolean
9         return isEmailNotEmpty
10    }
11
12    fun sendEmailReset(email:String) {
13        forgetPasswordError.value = ""
14        auth.sendPasswordResetEmail(email).addOnCompleteListener {
15            if(it.isSuccessful){
16                isUserForgotPassword.value = true
17            } else {
18                forgetPasswordError.value = it.exception.toString()
19                isUserForgotPassword.value = false
20            }
21        }
22    }
23

```

21 | }
22 }
23 }

Kode 15: *View Model* untuk *Forget Password Activity*

5. CREATE PROFILE VIEW MODEL

```
1  class CreateProfileViewModel : ViewModel() {
2      private val today = ViewUtility().getCurrentDate()
3      private var auth : FirebaseAuth = Firebase.auth
4      private var storage : FirebaseStorage = Firebase.storage
5      private var firestore : FirebaseFirestore = Firebase.firestore
6      private var userImageUri : MutableLiveData<UriFileExtensions> =
7          MutableLiveData(null)
8
8      private var userModel : MutableLiveData<UserModel> = MutableLiveData(
9          UserModel(gender = Gender.MALE.toString(),
10             dob = today)
11     )
12
13     private var farmModel : MutableLiveData<FarmModel> = MutableLiveData(FarmModel())
14     private var updateStaff : MutableLiveData<ArrayList<UserModel>> =
15         MutableLiveData(arrayListOf())
16     private var currentStaff : MutableLiveData<ArrayList<UserModel>> =
17         MutableLiveData(null)
18
18
19     var isNotEmpty : MutableLiveData<Boolean> = MutableLiveData(false)
20     var isUserCreated : MutableLiveData<Boolean> = MutableLiveData(false)
21     var isFarmCreated : MutableLiveData<Boolean> = MutableLiveData(false)
22     var isStaffAdded : MutableLiveData<Boolean> = MutableLiveData(false)
23     var createProfileError : MutableLiveData<String> = MutableLiveData("")
24
25     private var updateUser : MutableLiveData<UserModel> = MutableLiveData(null)
26
27     companion object{
28         const val TAG = "createProfileViewModel"
29     }
30
31
32     fun checkNotEmpty(boolean: Boolean) : MutableLiveData<Boolean>{
33         isNotEmpty.value = boolean
34         return isNotEmpty
35     }
36
37
38     /** SET MODEL */
39     fun setCurrentUser(userModel: UserModel?){
40         userModel?.let {
41             this.userModel.value = userModel
42             isUserCreated.value = true
43             updateUserGender.value =
44                 Gender.getType(userModel.gender!!)?.getPosition()!!
45         }
46     }
47
48
49     fun getCurrentUserModel() : UserModel? = userModel.value
50
51     fun getCurrentFarmModel() : FarmModel? = farmModel.value
52
53
54     /** CREATE USER FRAGMENT */
55     fun setPhotoProfile(uri : Uri?, fileExtension: String?=null){
56         userImageUri.value = if(uri != null){
57             UriFileExtensions(uri, fileExtension!!)
58         }
```

```

58         } else {
59             null
60         }
61     Log.i(TAG, "setPhotoProfile: $uri")
62 }
63
64 fun getPhotoProfile() : MutableLiveData<UriFileExtensions>{
65     return userImageUri
66 }
67
68 fun setRole(role:Role){
69     userModel.value?.role = role.toString()
70 }
71
72 fun setGender(gender: Gender){
73     userModel.value?.gender = gender.toString()
74 }
75
76 fun setDOB(date:Long?) : String? {
77     val temp : UserModel? = userModel.value
78     temp?.dob = ViewUtility().formatDateToString(date)
79     userModel.value = temp
80     Log.i(TAG, "setDOB: $date")
81     return userModel.value?.dob
82 }
83
84 fun getDOB() : Long{
85     if(userModel.value?.dob == null){
86         val temp : UserModel? = userModel.value
87         temp?.dob = ViewUtility().getCurrentDate()
88         userModel.value = temp
89     }
90     return ViewUtility().formatStringToDate(userModel.value?.dob)
91 }
92
93 fun createUserProfile(name:String, phone:String, address:String, bio:String) {
94     try {
95         userModel.value?.apply {
96             this.name = name
97             this.email = auth.currentUser?.email
98             this.phone = phone
99             this.address = address
100            this.bio = bio
101            this.uid = auth.uid
102            this.joinedSince = today
103            this.performanceRate = 5.0f
104        }
105
106        if(userImageUri.value != null){
107            val storageReference : StorageReference =
108                storage.getReference("profile_images")
109                .child(System.currentTimeMillis().toString() +
110                    ".${userImageUri.value?.fileExtensions!!}")
111            val uploadTask = storageReference.putFile(userImageUri.value?.uri!!)
112            uploadTask.continueWithTask {
113                if(!it.isSuccessful){
114                    throw it.exception!!.cause!!
115                }
116                storageReference.downloadUrl
117            }.addOnCompleteListener {
118                if (it.isSuccessful) {
119                    it.result.let {
120                        userModel.value?.photo_url = it.toString()
121                        sendUserProfile()
122                    }
123                }
124            }
125        } else {
126            sendUserProfile()
127        }
128    }
129 }

```

```

126         } catch (e:Exception){
127             Log.e(TAG, "Error submit user", e)
128         }
129     }
130
131     private fun sendUserProfile(){
132         createProfileError.value = ""
133         try {
134             firestore.collection("users").document(auth.uid!!).set(userModel.value!!.to
135             ↪ HashMap())
136             .addOnCompleteListener {
137                 if (it.isSuccessful) {
138                     isUserCreated.value = true
139                     isNotEmpty.value = false
140                 } else {
141                     createProfileError.value = it.exception.toString()
142                     isUserCreated.value = false
143                 }
144             } catch (e:Exception){
145                 Log.e(TAG, "sendUserProfile failed", e)
146             }
147         }
148
149
150     /** CREATE FARM FRAGMENT */
151     fun createFarmProfile(name:String, description:String, location:String){
152         try {
153             createProfileError.value = ""
154             val db = firestore.collection("farms")
155             val ref : DocumentReference = db.document()
156
157             farmModel.value?.apply {
158                 this.farmId = farmId ?: ref.id
159                 this.name = name
160                 this.description = description
161                 this.location = location
162             }
163
164             db.document(farmModel.value!!.farmId!!).set(farmModel.value!!.toHashMap()).]
165             ↪ addOnCompleteListener
166             ↪ {
167                 if(it.isSuccessful){
168                     userModel.value?.farmId = farmModel.value!!.farmId!!
169                     sendUserProfile()
170                     isFarmCreated.value = isUserCreated.value == true
171                 } else {
172                     createProfileError.value = it.exception.toString()
173                     isFarmCreated.value = false
174                 }
175             } catch (e:Exception){
176                 Log.e(TAG, "Error submit farm", e)
177             }
178
179
180     /** EDIT FARM */
181     fun getCurrentStaff() : MutableLiveData<ArrayList<UserModel>> = currentStaff
182
183     fun getStaff() : MutableLiveData<ArrayList<UserModel>> = updateStaff
184
185     fun addStaff(userModel : UserModel){
186         updateStaff.value?.let {
187             val array : ArrayList<UserModel> = it
188             array.add(userModel)
189             updateStaff.value = array
190             Log.i(TAG, "addStaff: ${updateStaff.value}")
191         }
192     }

```

```

193
194     fun removeStaff(position : Int){
195         updateStaff.value?.let {
196             val array : ArrayList<UserModel> = it
197             array.removeAt(position)
198             updateStaff.value = array
199             Log.i(TAG, "removeStaff: ${updateStaff.value}")
200         }
201     }
202
203     fun createStaff(){
204         try {
205             farmModel.value?.let {
206                 firestore.collection("farms").document(it.farmId!!)
207                     .set(it.toHashMap())
208             }
209
210             currentStaff.value?.let {
211                 for (staff in it) {
212                     firestore.collection("users").document(staff.uid!!)
213                         .update("farmId", null)
214                 }
215             }
216
217             updateStaff.value?.let {
218                 for (staff in it) {
219                     firestore.collection("users").document(staff.uid!!)
220                         .update("farmId", farmModel.value?.farmId)
221                 }
222                 isStaffAdded.value = true
223             }
224         } catch (e:Exception){
225             Log.e(TAG, "createStaff: ", e)
226         }
227     }
228
229     fun updateStaff() {
230         try {
231             firestore.collection("users")
232                 .get().addOnCompleteListener {
233                     if(it.isSuccessful) {
234                         val staffs: ArrayList<UserModel> = arrayListOf()
235                         for (staffDocument in it.result.documents) {
236                             staffDocument?.let { staff ->
237                                 staff.toUserModel()?.let { staffModel ->
238                                     staffModel.farmId?.let {
239                                         if (it == farmModel.value?.farmId) {
240                                             staffs.add(staffModel)
241                                         }
242                                     }
243                                 }
244                             }
245                         }
246                         currentStaff.value = staffs
247                         updateStaff.value = staffs
248                     }
249                 }
250         } catch (e:Exception){
251             Log.e(TAG, "updateCurrentStaff for farm ${farmModel.value?.farmId}: ", e)
252         }
253     }
254
255     /** EDIT USER */
256     private var updateUserGender : MutableLiveData<Int> = MutableLiveData(0)
257     fun getUpdateGender() : Int? = updateUserGender.value
258 }

```

Kode 16: View Model untuk Create Profile Activity beserta Create User dan Create Farm frgments. Termasuk juga untuk Edit Profile User dan Edit Profile Farm activities

6. MAIN VIEW MODEL

```
1 class MainViewModel : ViewModel() {
2     private var auth : FirebaseAuth = FirebaseAuth.auth
3     private var firestore : FirebaseFirestore = FirebaseFirestore.firebaseio
4     private var currentUserModel : MutableLiveData<UserModel?> = MutableLiveData(null)
5     private var currentFarmModel : MutableLiveData<FarmModel?> = MutableLiveData(null)
6     private var currentKitModels : MutableLiveData<ArrayList<KitModel>?> =
7         MutableLiveData(null)
8     private var currentNoteModels : MutableLiveData<ArrayList<NoteModel>?> =
9         MutableLiveData(null)
10    private var allUserModels : MutableLiveData<ArrayList<UserModel>?> =
11        MutableLiveData(null)
12    private var allPlantModels : MutableLiveData<ArrayList<PlantModel>?> =
13        MutableLiveData(null)
14
15    companion object{
16        const val TAG = "mainViewModel"
17    }
18
19    fun setCurrentData(userModel: UserModel?, farmModel: FarmModel?){
20        getUsersUpdate()
21        getPlantsUpdate()
22        userModel?.let {
23            currentUserModel.value = it
24            farmModel?.let {
25                currentFarmModel.value = it
26            } ?: kotlin.run {
27                getFarmsUpdate(currentFarmModel.value?.farmId)
28            }
29        } ?: kotlin.run {
30            getUsersUpdate()
31        }
32    }
33
34    /** GET DATA */
35    fun getCurrentUser() : MutableLiveData<UserModel?> = currentUserModel
36    fun getCurrentFarm() : MutableLiveData<FarmModel?> = currentFarmModel
37    fun getCurrentKits() : MutableLiveData<ArrayList<KitModel>?> = currentKitModels
38    fun getCurrentNotes() : MutableLiveData<ArrayList<NoteModel>?> = currentNoteModels
39    fun getAllUsers() : MutableLiveData<ArrayList<UserModel>?> = allUserModels
40    fun getAllPosts() : MutableLiveData<ArrayList<PlantModel>?> = allPlantModels
41
42    /** UPDATE DATABASE LISTENER */
43    private lateinit var usersListener : ListenerRegistration
44    private lateinit var farmsListener : ListenerRegistration
45    private lateinit var kitsListener : ListenerRegistration
46    private var monitoringsListeners : ArrayList<ListenerRegistration> = arrayListOf()
47    private var cropsListeners : ArrayList<ListenerRegistration> = arrayListOf()
48    private lateinit var plantsListener : ListenerRegistration
49
50    private fun getUsersUpdate(){
51        val db = firestore.collection("users")
52        try {
53            usersListener =
54                db.orderBy("timestamp", Query.Direction.DESCENDING)
55                    .addSnapshotListener { usersSnapshot, error ->
56                        usersSnapshot?.let {
57                            val users : ArrayList<UserModel> = arrayListOf()
58                            for(userDocument in usersSnapshot.documents){
59                                userDocument?.let { user ->
60                                    user.toUserModel()?.let { userModel ->
61                                        users.add(userModel)
62                                        val userId = userDocument.id
63                                        if(auth.uid == userId) {
64                                            currentUserModel.value = userModel
65                                            getFarmsUpdate(userModel.farmId)
66
67                                            val userRef = db.document(userId)
68                                            userRef.collection("notes")
69                                        }
70                                    }
71                                }
72                            }
73                        }
74                    }
75                }
76            }
77        }
78    }
79}
```

```

65         .orderBy("timestamp",
66             Query.Direction.DESCENDING)
67         .addSnapshotListener { notesSnapshot,
68             error ->
69             notesSnapshot?.let {
70                 val notes: ArrayList<NoteModel>
71                     = arrayListOf()
72                 for (noteDocument in
73                     notesSnapshot.documents) {
74                     noteDocument?.let { note ->
75                         note.toNoteModel()?.let
76                             { noteModel ->
77                                 notes.add(noteModel)
78                             }
79                         }
80                     }
81                 currentNoteModels.value = notes
82                 Log.i(TAG, "getUsersUpdate ${currentUserModel.value?.uid}")
83                 + +
84                     "\tnotes:${currentNoteModels.value?.size}\n"
85                     "\n"
86                     +"${currentUserModel.value?.uid}\n"
87                     )
88                 }
89             }
90             allUserModels.value = users
91             Log.i(TAG, "getUsersUpdate: ${allUserModels.value?.size}\n")
92         }
93
94             error?.let {
95                 Log.e(TAG, "Listen users failed", it)
96             }
97         }
98     } catch (e:Exception){
99         Log.e(TAG, "getUsersUpdate() error", e)
100    }
101}
102
103 private fun getFarmsUpdate(farmId: String?){
104     if(farmId == null){
105         currentFarmModel.value = null
106     } else {
107         try{
108             val db = firestore.collection("farms").document(farmId)
109             farmsListener = db.addSnapshotListener { farmSnapshot, error ->
110                 farmSnapshot?.let { farm ->
111                     farm.toFarmModel()?.let { farmModel ->
112                         currentFarmModel.value = farmModel
113
114                         val kitsRef = db.collection("kits")
115                         kitsListener = kitsRef.addSnapshotListener{ kitSnapshot,
116                             error ->
117                             kitSnapshot?.let { kitDocuments ->
118                                 val kits : ArrayList<KitModel> = arrayListOf()
119                                 kitDocuments.documents.forEachIndexed { index,
120                                     kitDocument ->
121                                     kitDocument?.toKitModel()?.let { kit ->
122                                         kit?.let { kitModel ->

```

```

121     val kitId = kitDocument.id
122     val kitRef = kitsRef.document(kitId)
123     kits.add(kitModel)
124
125     val monitorListener =
126         kitRef.collection("dataMonitorings")
127             .orderBy("timestamp",
128                     Query.Direction.DESCENDING)
129             .addSnapshotListener {
130                 monitoringSnapshot, error ->
131                 monitoringSnapshot?.let {
132                     monitoringDocuments ->
133                         if(monitoringDocuments.documents.size > 0){
134                             val lastMonitoring : DocumentSnapshot? = monitoringDocuments.documents[0]
135                             monitoringModel = lastMonitoring.toObject(DataMonitoringModel::class.java)!!
136                             updateKitModel(index,
137                                 monitoringModel =
138                                     lastMonitoring)
139                         }
140
141                     error?.let {
142                         Log.e(TAG, "Listen
143                             dataMonitoring from kit
144                             $kitId failed", it)
145                     }
146
147                     cropsListener =
148                         kitRef.collection("crops")
149                             .orderBy("timestamp",
150                                     Query.Direction.DESCENDING)
151                             .addSnapshotListener {
152                                 cropsSnapshot, error ->
153                                     cropsSnapshot?.let {
154                                         cropsDocuments ->
155                                             if(cropsDocuments.documents.size > 0){
156                                                 val lastCropsModel : CropsModel? = cropsDocuments.documents[0].toCropsModel()
157                                                 updateKitModel(index,
158                                     cropsModel =
159                                         lastCropsModel)
160
161                     }
162
163                     monitoringsListeners.add(monitorListener)
164                     cropsListeners.add(cropsListener)
165
166                 }
167
168             }
169             currentKitModels.value = kits
170             updateKitModel(0)
171         }

```

```

163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182     private fun getPlantsUpdate(){
183         val db = firestore.collection("plants").orderBy("timestamp",
184             Query.Direction.DESCENDING)
185         try {
186             plantsListener = db.addSnapshotListener { plantsSnapshot, error ->
187                 plantsSnapshot?.let {
188                     val plants : ArrayList<PlantModel> = arrayListOf()
189                     for(plantDocument in plantsSnapshot.documents){
190                         plantDocument?.let { plant ->
191                             plant.toPlantModel()?.let { plantModel ->
192                                 plants.add(plantModel)
193                             }
194                         }
195                     allPlantModels.value = plants
196                     Log.i(TAG, "getPlantsUpdate: ${allPlantModels.value?.size}")
197                 }
198
199                 error?.let {
200                     Log.e(TAG, "Listen plants failed", it)
201                 }
202             }
203         } catch (e:Exception){
204             Log.e(TAG, "real time plant error", e)
205         }
206     }
207
208     private fun updateKitModel(index:Int, cropsModel: CropsModel?=null,
209     &gt; monitoringModel: DataMonitoringModel? = null){
210         try {
211             currentKitModels.value?.let {
212                 val kitModels = it
213                 kitModels.forEachIndexed { kitIndex, kitModel ->
214                     if(kitIndex == index){
215                         cropsModel?.let {
216                             kitModel.lastCrops = it
217                         }
218                         monitoringModel?.let {
219                             kitModel.lastMonitoring = it
220                         }
221                     }
222                 currentKitModels.value = kitModels
223             }
224
225             currentFarmModel.value?.let {
226                 val farmModel = it
227                 farmModel.kitModels = currentKitModels.value
228                 currentFarmModel.value = farmModel
229             }

```

```

230     }catch (e:Exception) {
231         Log.e(TAG, "updateKitModel ${currentKitModels.value!![index]}\n", e)
232     }
233
234     Log.i(TAG, "getFarmsUpdate ${currentFarmModel.value?.farmId}" +
235             "\ntkits:${currentKitModels.value?.size}\n" +
236             "${currentFarmModel.value}\n")
237 }
238
239 /**
240 var isNoteDeleted : MutableLiveData<Boolean?> = MutableLiveData(null)
241 var deleteNoteError : MutableLiveData<String> = MutableLiveData(null)
242
243 fun getNote(position:Int) : NoteModel?{
244     return currentNoteModels.value?.get(position)
245 }
246
247 fun deleteNote(position:Int){
248     isNoteDeleted.value = null
249     val noteId = getNote(position)?.noteId
250     try {
251         noteId?.let{
252             firestore.collection("users").document(auth.uid!!)
253                 .collection("notes").document(noteId)
254                 .delete().addOnCompleteListener {
255                     if (it.isSuccessful) {
256                         isNoteDeleted.value = true
257                     } else {
258                         isNoteDeleted.value = false
259                         deleteNoteError.value = it.exception.toString()
260                         Log.e(TAG, "deleteNote $noteId\n ${it.exception}")
261                     }
262                 }
263         }
264     } catch (e:Exception){
265         Log.e(TAG, "deleteNote $noteId failed", e)
266         isNoteDeleted.value = false
267         deleteNoteError.value = e.toString()
268     }
269 }
270
271 fun refreshNotes(){
272     usersListener.remove()
273     currentNoteModels.value?.clear()
274     getUsersUpdate()
275 }
276
277 /**
278 var isPostDeleted : MutableLiveData<Boolean?> = MutableLiveData(null)
279 var deletePostError : MutableLiveData<String> = MutableLiveData(null)
280
281 fun getPostUser(position: Int) : UserModel? {
282     var output : UserModel? = null
283     val post = getPost(position)?.userId
284     post?.let { postId -
285         allUserModels.value?.forEach {
286             if (it.uid == postId){
287                 output = it
288             }
289         }
290     }
291     return output
292 }
293
294 fun getPost(position: Int) : PlantModel?{
295     return allPlantModels.value?.get(position)
296 }
297
298 fun deletePost(position: Int){
299     isPostDeleted.value = null

```

```

300     val plantModel =getPost(position)
301     val plantId = plantModel?.plantId
302     try {
303         if(plantModel?.userId == auth.uid) {
304             firestore.collection("plants").document(plantId!!)
305                 .delete().addOnCompleteListener {
306                     if (it.isSuccessful) {
307                         isPostDeleted.value = true
308                     } else {
309                         isPostDeleted.value = false
310                         deletePostError.value = it.exception.toString()
311                         Log.e(TAG, "deletePost $plantId\n ${it.exception}")
312                     }
313                 }
314             }
315         } catch (e:Exception){
316             Log.e(TAG, "deletePost $plantId failed", e)
317         }
318     }
319
320     fun refreshPosts(){
321         plantsListener.remove()
322         allPlantModels.value?.clear()
323         getPlantsUpdate()
324     }
325
326     /** FARM DASHBOARD */
327     fun getKit(position: Int) : KitModel? {
328         return currentKitModels.value?.get(position)
329     }
330
331     fun refreshFarm(){
332         farmsListener.remove()
333         kitsListener.remove()
334         monitoringsListeners.forEach { it.remove() }
335         cropsListeners.forEach { it.remove() }
336
337         currentFarmModel.value?.let {
338             val farmModel = it
339             farmModel.kitModels?.clear()
340             currentFarmModel.value = farmModel
341         }
342         currentKitModels.value?.clear()
343         getFarmsUpdate(currentUserModel.value?.farmId)
344     }
345
346     /** SIGN OUT */
347     var isUserSignOut : MutableLiveData<Boolean?> = MutableLiveData(null)
348     var signOutError : MutableLiveData<String?> = MutableLiveData(null)
349
350     fun signOut(){
351         try {
352             auth.signOut()
353             isUserSignOut.value = true
354         } catch (e:Exception){
355             isUserSignOut.value = false
356             signOutError.value = e.toString()
357         }
358     }
359 }

```

Kode 17: *View Model* untuk *Main Activity* beserta *Main Home*, *Main Posts*, *Main Add*, *Main Notes*, dan *Main Profile* fragments

7. ADD CROPS VIEW MODEL

```

1 class AddCropsViewModel : ViewModel() {
2     var isPlantExist: MutableLiveData<Boolean> = MutableLiveData(false)

```

```
3     private var plantModel : MutableLiveData<PlantModel> = MutableLiveData(PlantModel())
4
5     fun setPlantModel(plantModel: PlantModel) {
6         this.plantModel.value = plantModel
7         isPlantExist.value = true
8     }
9
10    fun getPlantModel() : MutableLiveData<PlantModel> = plantModel
11 }
```

Kode 18: *View Model* untuk *Add Crops Activity*

8. ADD DATA MONITORING VIEW MODEL

```
1 class AddDataMonitoringViewModel : ViewModel() {
2     private var auth : FirebaseAuth = Firebase.auth
3     private var firestore : FirebaseFirestore = Firebase.firestore
4     var currentUserModel : MutableLiveData<UserModel> = MutableLiveData(UserModel())
5     var currentFarmModel : MutableLiveData<FarmModel> = MutableLiveData(FarmModel())
6     private var currentKitModel : MutableLiveData<KitModel> = MutableLiveData(null)
7     private var currentPlantModel : MutableLiveData<PlantModel> = MutableLiveData(null)
8     private var currentCropsModel : MutableLiveData<CropsModel> = MutableLiveData(null)
9     private var dataMonitoringModel : MutableLiveData<DataMonitoringModel> =
10        MutableLiveData(DataMonitoringModel())
11     private var kitSelectors : MutableLiveData<MutableList<String>> =
12        MutableLiveData(mutableListOf<String>())
13     private var availableKits : MutableLiveData<ArrayList<KitModel>> =
14        MutableLiveData(arrayListOf())
15
16     private var water : MutableLiveData<Float> = MutableLiveData(0f)
17     private var nutrient : MutableLiveData<Float> = MutableLiveData(0f)
18     private var turb : MutableLiveData<Float> = MutableLiveData(0f)
19     private var temp : MutableLiveData<Float> = MutableLiveData(0f)
20     private var humid : MutableLiveData<Float> = MutableLiveData(0f)
21     private var acid : MutableLiveData<Float> = MutableLiveData(0f)
22     private var isNotEmpty : MutableLiveData<Boolean> = MutableLiveData(false)
23     var isDataMonitoringAdd: MutableLiveData<Boolean> = MutableLiveData(false)
24     var addDataMonitoringError : MutableLiveData<String> = MutableLiveData("")
25     var isCropsAdd: MutableLiveData<Boolean> = MutableLiveData(false)
26     var addCropsError : MutableLiveData<String> = MutableLiveData("")
27     private var isMenuAddCrops : MutableLiveData<Boolean> = MutableLiveData(false)
28
29     companion object {
30         const val TAG = "addDataMonitoringViewModel"
31     }
32
33     fun checkNotEmpty(boolean: Boolean) : MutableLiveData<Boolean> {
34         isNotEmpty.value = boolean
35         return isNotEmpty
36     }
37
38     fun setCurrentData(userModel: UserModel?, farmModel: FarmModel?,
39                        isAddCrops:Boolean){
40         isMenuAddCrops.value = isAddCrops
41         userModel?.let {
42             this.currentUserModel.value = it
43         }
44         farmModel?.let {
45             this.currentFarmModel.value = it
46             val kitSelectors : MutableList<String> = mutableListOf()
47             currentFarmModel.value?.kitModels?.let {
48                 val plantTrue : ArrayList<Int> = arrayListOf()
49                 it.forEachIndexed { index, kit ->
50                     if((isAddCrops && kit.isPlanted != true)) {
51                         kitSelectors.add(kit.name.toString())
52                         availableKits.value?.add(kit)
53                     }
54                     if(!isAddCrops){
55                         kitSelectors.add(kit.name.toString())
56                     }
57                 }
58             }
59         }
60     }
61 }
```

```

52             }
53         }
54         this.kitSelectors.value = kitSelectors
55     }
56     Log.i(TAG, "$farmModel")
57 }
58 }
59
60 fun setCurrentKit(position:Int){
61     currentKitModel.value = if(isMenuAddCrops.value == true){
62         availableKits.value?.get(position)
63     } else {
64         currentFarmModel.value?.kitModels?.get(position)
65     }
66     val cropsModel = currentFarmModel.value?.kitModels?.get(position)?.lastCrops
67     currentCropsModel.value = CropsModel()
68     Log.i(TAG, "setCurrentKit: ${currentKitModel.value}")
69 }
70
71 fun getKitSelector() : MutableLiveData<MutableList<String>> = kitSelectors
72
73 fun getCurrentKit() : MutableLiveData<KitModel> = currentKitModel
74
75 fun getCurrentCrops() : MutableLiveData<CropsModel> = currentCropsModel
76
77 fun setCurrentPlant(plantModel: PlantModel?) {
78     currentPlantModel.value = plantModel
79 }
80
81 fun getCurrentPlant() : MutableLiveData<PlantModel> = currentPlantModel
82
83 fun setNumberPickerValue(currentValue : Float, type: NumberPickerType?){
84     when(type){
85         NumberPickerType.WATER_TANK -> water.value = currentValue
86         NumberPickerType.NUTRIENT_TANK -> nutrient.value = currentValue
87         NumberPickerType.TURBIDITY -> turb.value = currentValue
88         NumberPickerType.TEMPERATURE -> temp.value = currentValue
89         NumberPickerType.HUMIDITY -> humid.value = currentValue
90         NumberPickerType.ACIDITY -> acid.value = currentValue
91         else -> { }
92     }
93 }
94
95 fun getNumberPickerValue(type: NumberPickerType?) : MutableLiveData<Float>?{
96     return when(type){
97         NumberPickerType.WATER_TANK -> water
98         NumberPickerType.NUTRIENT_TANK -> nutrient
99         NumberPickerType.TURBIDITY -> turb
100        NumberPickerType.TEMPERATURE -> temp
101        NumberPickerType.HUMIDITY -> humid
102        NumberPickerType.ACIDITY -> acid
103        else -> null
104    }
105 }
106
107 /**
108 fun addDataMonitoring(){
109     val db =
110         firestore.collection("farms").document(currentFarmModel.value?.farmId!!)
111         .collection("kits").document(currentKitModel.value?.kitId!!)
112         .collection("dataMonitorings")
113     val ref : DocumentReference = db.document()
114
115     try {
116         dataMonitoringModel.value?.apply {
117             this.dataId = ref.id
118             this.userId = auth.uid
119             this.timestamp = ViewUtility().getCurrentTimestamp()
120             this.temperature = temp.value
121             this.humidity = humid.value

```

```

121         this.turbidity = turb.value
122         this.waterTank = water.value
123         this.nutrientTank = nutrient.value
124         this.ph = acid.value
125     }
126
127     db.document(ref.id).set(dataMonitoringModel.value!!.toHashMap()).addOnCompl_]
128     ↪ eteListener
129     ↪ {
130         if(it.isSuccessful){
131             isDataMonitoringAdd.value = true
132         } else {
133             addDataMonitoringError.value = it.exception.toString()
134             isDataMonitoringAdd.value = false
135         }
136     } catch (e:Exception){
137         Log.e(TAG, "Error submit data monitoring", e)
138     }
139 }
140
141 /**
142 fun addCrops(){
143     val db =
144         ↪ firestore.collection("farms").document(currentFarmModel.value?.farmId!!)
145         .collection("kits").document(currentKitModel.value?.kitId!!)
146         .collection("crops")
147     val ref : DocumentReference = db.document()
148
149     try{
150         currentCropsModel.value.apply {
151             currentCropsModel.value = CropsModel(
152                 cropsId = ref.id,
153                 userId = auth.uid,
154                 plantId = currentPlantModel.value?.plantId,
155                 tempLv = currentPlantModel.value?.tempLv,
156                 humidLv = currentPlantModel.value?.humidLv,
157                 phLv = currentPlantModel.value?.phLv
158             )
159
160             db.document(ref.id).set(currentCropsModel.value!!.toHashMap()).addOnCom_]
161             ↪ pleteListener
162             ↪ {
163                 if(it.isSuccessful){
164                     updateKit(ref.id)
165                 } else {
166                     addCropsError.value = it.exception.toString()
167                     isCropsAdd.value = false
168                 }
169             }
170             Log.e(TAG, "Error add crops to ${currentKitModel.value?.kitId}", e)
171         }
172     }
173
174     private fun updateKit(cropsId : String){
175         val db =
176             ↪ firestore.collection("farms").document(currentFarmModel.value?.farmId!!)
177             .collection("kits").document(currentKitModel.value?.kitId!!)
178         try {
179             currentKitModel.value?.isPlanted = true
180             db.set(currentKitModel.value!!.toHashMap()).addOnCompleteListener {
181                 if(it.isSuccessful){
182                     isCropsAdd.value = true
183                     if(isMenuAddCrops.value == false) {
184                         dataMonitoringModel.value?.cropsId = cropsId
185                         addDataMonitoring()

```

```

185             }
186         } else {
187             addCropsError.value = it.exception.toString()
188             isCropsAdd.value = false
189         }
190     }
191 } catch (e:Exception){
192     Log.e(TAG, "Error update kit ${currentKitModel.value?.kitId}", e)
193 }
194 }
195 }
```

Kode 19: *View Model* untuk *Add Data Monitoring Activity*

9. ADD KIT VIEW MODEL

```

1 class AddKitViewModel : ViewModel() {
2     private var firestore : FirebaseFirestore = Firebase.firebaseio
3     var isNotEmpty : MutableLiveData<Boolean> = MutableLiveData(false)
4     var isKitAdd : MutableLiveData<Boolean> = MutableLiveData(false)
5     var addKitError : MutableLiveData<String> = MutableLiveData("")
6
7     private var currentUserModel : MutableLiveData<UserModel> = MutableLiveData(null)
8     private var currentFarmModel : MutableLiveData<FarmModel> = MutableLiveData(null)
9     private var currentKitModel : MutableLiveData<KitModel> =
10        MutableLiveData(KitModel())
11
12     private var kitWidth : MutableLiveData<Float> = MutableLiveData(0f)
13     private var kitLength : MutableLiveData<Float> = MutableLiveData(0f)
14     private var waterMin : MutableLiveData<Float> = MutableLiveData(0f)
15     private var waterMax : MutableLiveData<Float> = MutableLiveData(0f)
16     private var nutrientMin : MutableLiveData<Float> = MutableLiveData(0f)
17     private var nutrientMax : MutableLiveData<Float> = MutableLiveData(0f)
18     private var turbidityMin : MutableLiveData<Float> = MutableLiveData(0f)
19     private var turbidityMax : MutableLiveData<Float> = MutableLiveData(0f)
20
21     companion object {
22         const val TAG = "addKitViewModel"
23     }
24
25     fun checkNotEmpty(boolean: Boolean) : MutableLiveData<Boolean> {
26         isNotEmpty.value = boolean
27         return isNotEmpty
28     }
29
30     fun setCurrentData(userModel: UserModel?, farmModel: FarmModel?, kitModel:
31        KitModel?) {
32         currentUserModel.value = userModel
33         currentFarmModel.value = farmModel
34         kitModel?.let{
35             this.currentKitModel.value = it
36         }
37     }
38
39     fun getCurrentFarm():MutableLiveData<FarmModel> = currentFarmModel
40     fun getCurrentKit() : MutableLiveData<KitModel> = currentKitModel
41
42     fun setNumberPickerValue(currentValue : Float, type: NumberPickerType?){
43         when(type){
44             NumberPickerType.KIT_WIDTH -> kitWidth.value = currentValue
45             NumberPickerType.KIT_LENGTH -> kitLength.value = currentValue
46             NumberPickerType.WATER_MIN -> waterMin.value = currentValue
47             NumberPickerType.WATER_MAX -> waterMax.value = currentValue
48             NumberPickerType.NUTRIENT_MIN -> nutrientMin.value = currentValue
49             NumberPickerType.NUTRIENT_MAX -> nutrientMax.value = currentValue
50             NumberPickerType.TURBIDITY_MIN -> turbidityMin.value = currentValue
51             NumberPickerType.TURBIDITY_MAX -> turbidityMax.value = currentValue
52         }
53     }
54 }
```

```

52         else -> { }
53     }
54 }
55
56 fun getNumberPickerValue(type: NumberPickerType?) : MutableLiveData<Float>?{
57     return when(type){
58         NumberPickerType.KIT_WIDTH -> kitWidth
59         NumberPickerType.KIT_LENGTH -> kitLength
60         NumberPickerType.WATER_MIN -> waterMin
61         NumberPickerType.WATER_MAX -> waterMax
62         NumberPickerType.NUTRIENT_MIN -> nutrientMin
63         NumberPickerType.NUTRIENT_MAX -> nutrientMax
64         NumberPickerType.TURBIDITY_MIN -> turbidityMin
65         NumberPickerType.TURBIDITY_MAX -> turbidityMax
66         else -> null
67     }
68 }
69
70 fun createKit(name:String, position:String){
71     val db = firestore.collection("farms").document(currentUserModel.value?.farmId!]
72     ↪ !).collection("kits")
73     val ref : DocumentReference = db.document()
74
75     try {
76         currentKitModel.value?.apply {
77             this.kitId = kitId ?: ref.id
78             this.isPlanted = isPlanted ?: false
79             this.name = name
80             this.position = position
81             this.length = kitLength.value?.toInt()
82             this.width = kitWidth.value?.toInt()
83             this.waterLv = ScoreLevel(waterMin.value, waterMax.value)
84             this.nutrientLv = ScoreLevel(nutrientMin.value, nutrientMax.value)
85             this.turbidityLv = ScoreLevel(turbidityMin.value, turbidityMax.value)
86             this.timestamp = ViewUtility().getCurrentTimestamp()
87         }
88
89         val kitRef = db.document(currentKitModel.value!!..kitId!!)
90         kitRef.set(currentKitModel.value!!.toHashMap()).addOnCompleteListener {
91             if(it.isSuccessful){
92                 if(currentKitModel.value!!..kitId == ref.id){
93                     val dataRef = kitRef.collection("dataMonitorings")
94                     val dataId = dataRef.document().id
95                     val initData = DataMonitoringModel(userId =
96                         ↪ FirebaseAuth.uid, dataId = dataId)
97                     dataRef.document(dataId).set(initData.toHashMap()).addOnComp
98                     ↪ leteListener
99                     ↪ {
100                         if(it.isSuccessful){
101                             isKitAdd.value = true
102                         }
103                     } else {
104                         addKitError.value = it.exception.toString()
105                         isKitAdd.value = false
106                     }
107                 }
108             } catch (e:Exception){
109                 Log.e(TAG, "Error submit kit", e)
110             }
111         }
112     }

```

Kode 20: *View Model* untuk *Add Kit Activity*

10. ADD NOTE VIEW MODEL

```
1  class AddNoteViewModel : ViewModel(){
2      private var auth : FirebaseAuth = Firebase.auth
3      private var firestore : FirebaseFirestore = Firebase.firebaseio
4      private var noteModel : MutableLiveData<NoteModel> = MutableLiveData(NoteModel())
5      private var timeHour : MutableLiveData<Int> = MutableLiveData(0)
6      private var timeMinute : MutableLiveData<Int> = MutableLiveData(0)
7      var isNotEmpty : MutableLiveData<Boolean> = MutableLiveData(false)
8      var isNoteAdd: MutableLiveData<Boolean> = MutableLiveData(false)
9      var addNoteError : MutableLiveData<String> = MutableLiveData("")
10
11     companion object {
12         const val TAG = "addNote"
13     }
14
15     fun setCurrentData(noteModel: NoteModel?){
16         noteModel?.let{
17             this.noteModel.value = it
18         }
19     }
20
21     fun getCurrentNoteModel() : MutableLiveData<NoteModel> = noteModel
22
23     fun checkNotEmpty(boolean: Boolean) : MutableLiveData<Boolean> {
24         isNotEmpty.value = boolean
25         return isNotEmpty
26     }
27
28     fun getDate(): Long {
29         if(noteModel.value?.date == null){
30             noteModel.value?.date = ViewUtility().getCurrentDate()
31         }
32         return ViewUtility().formatStringToDate(noteModel.value?.date)
33     }
34
35     fun setDate(date: Long?): String? {
36         noteModel.value?.date = ViewUtility().formatDateToString(date)
37         return noteModel.value?.date
38     }
39
40     fun setTime(hour: Int, minute: Int) : String?{
41         noteModel.value?.time = ViewUtility().formatTimeToString(hour, minute)
42         return noteModel.value?.time
43     }
44
45     fun getTime(){
46         if(noteModel.value?.time == null){
47             noteModel.value?.time = ViewUtility().getCurrentTime()
48         }
49         val (hour, minute) = ViewUtility().formatStringToTime(noteModel.value?.time)
50         timeHour.value = hour
51         timeMinute.value = minute
52     }
53
54     fun getTimeHour() : Int {
55         timeHour.value?.let {
56             return it
57         }
58         return 0
59     }
60
61     fun getTimeMinute() : Int {
62         timeMinute.value?.let {
63             return it
64         }
65         return 0
66     }
67
68     fun createNote(title:String, description:String){
```

```

69     val db = firestore.collection("users").document(auth.uid!!).collection("notes")
70     val ref : DocumentReference = db.document()
71
72     try {
73         noteModel.value?.apply{
74             this.noteId = noteId ?: ref.id
75             this.title = title
76             this.description = description
77         }
78
79         db.document(noteModel.value!!.noteId!!).set(noteModel.value!!.toHashMap()).]
80             ← addOnCompleteListener
81             ← {
82                 if(it.isSuccessful){
83                     isNoteAdd.value = true
84                 } else {
85                     addNoteError.value = it.exception.toString()
86                     isNoteAdd.value = false
87                 }
88             }
89         } catch (e:Exception){
90             Log.e(TAG, "Error submit note", e)
91         }
92     }

```

Kode 21: *View Model* untuk *Add Note Activity*

11. ADD PLANT VIEW MODEL

```

1  class AddPlantViewModel : ViewModel() {
2      private var auth : FirebaseAuth = Firebase.auth
3      private var firestore : FirebaseFirestore = Firebase.firebaseio
4      private var storage : FirebaseStorage = Firebase.storage
5      private var growth : MutableLiveData<Float> = MutableLiveData(0f)
6      private var tempMin : MutableLiveData<Float> = MutableLiveData(0f)
7      private var tempMax : MutableLiveData<Float> = MutableLiveData(0f)
8      private var humidMin : MutableLiveData<Float> = MutableLiveData(0f)
9      private var humidMax : MutableLiveData<Float> = MutableLiveData(0f)
10     private var acidMin : MutableLiveData<Float> = MutableLiveData(0f)
11     private var acidMax : MutableLiveData<Float> = MutableLiveData(0f)
12     private var imageUri : MutableLiveData<UriFileExtensions> = MutableLiveData(null)
13
14     var plantModel : MutableLiveData<PlantModel> = MutableLiveData(PlantModel())
15
16     var isNotEmpty : MutableLiveData<Boolean> = MutableLiveData(false)
17     var isPlantAdd : MutableLiveData<Boolean> = MutableLiveData(false)
18     var addPlantError : MutableLiveData<String> = MutableLiveData("")
19
20     companion object {
21         const val TAG = "addPlantViewModel"
22     }
23
24     fun checkNotEmpty(boolean: Boolean) : MutableLiveData<Boolean> {
25         isNotEmpty.value = boolean
26         return isNotEmpty
27     }
28
29     fun setCurrentData(plantModel: PlantModel?){
30         plantModel?.let{
31             this.plantModel.value = it
32         }
33     }
34
35     fun setNumberPickerValue(currentValue : Float, type: NumberPickerType?){
36         when(type){
37             NumberPickerType.GROWTH_TIME -> growth.value = currentValue
38             NumberPickerType.TEMP_MIN -> tempMin.value = currentValue

```

```

39         NumberPickerType.TEMP_MAX -> tempMax.value = currentValue
40         NumberPickerType.HUMID_MIN -> humidMin.value = currentValue
41         NumberPickerType.HUMID_MAX -> humidMax.value = currentValue
42         NumberPickerType.ACID_MIN -> acidMin.value = currentValue
43         NumberPickerType.ACID_MAX -> acidMax.value = currentValue
44     else -> { }
45   }
46 }
47
48 fun getNumberPickerValue(type: NumberPickerType?) : MutableLiveData<Float>?{
49   return when(type){
50     NumberPickerType.GROWTH_TIME -> growth
51     NumberPickerType.TEMP_MIN -> tempMin
52     NumberPickerType.TEMP_MAX -> tempMax
53     NumberPickerType.HUMID_MIN -> humidMin
54     NumberPickerType.HUMID_MAX -> humidMax
55     NumberPickerType.ACID_MIN -> acidMin
56     NumberPickerType.ACID_MAX -> acidMax
57     else -> null
58   }
59 }
60
61 fun createPlant(name:String, description:String){
62   try{
63     plantModel.value?.apply {
64       this.name = name
65       this.description = description
66       this.growthTime = growth.value?.toInt()
67       this.tempLv = ScoreLevel(tempMin.value, tempMax.value)
68       this.humidLv = ScoreLevel(humidMin.value, humidMax.value)
69       this.phLv = ScoreLevel(acidMin.value, acidMax.value)
70       this.userId = auth.uid!!
71     }
72
73     if(imageUri.value != null){
74       val storageReference : StorageReference =
75         storage.getReference("profile_images")
76         .child(System.currentTimeMillis().toString() +
77           ".$imageUri.value?.fileExtensions!!")
78       val uploadTask = storageReference.putFile(imageUri.value?.uri!!)
79       uploadTask.continueWithTask {
80         if(!it.isSuccessful){
81           throw it.exception!!.cause!!
82         }
83         storageReference.downloadUrl
84       }.addOnCompleteListener {
85         if (it.isSuccessful) {
86           it.result.let {
87             plantModel.value?.photo_url = it.toString()
88             sendPlantProfile()
89           }
90         }
91       } else {
92         sendPlantProfile()
93       }
94     } catch (e:Exception){
95       Log.e(TAG, "Error submit plant ", e)
96     }
97
98   private fun sendPlantProfile(){
99     val db = firestore.collection("plants")
100    val ref : DocumentReference = db.document()
101
102    plantModel.value?.apply {
103      this.plantId = plantId ?: ref.id
104    }
105  }

```

```

106         db.document(plantModel.value!!.plantId!!).set(plantModel.value!!.toHashMap()).addOnCompleteListener {
107             if(it.isSuccessful){
108                 isPlantAdd.value = true
109             } else {
110                 addPlantError.value = it.exception.toString()
111                 isPlantAdd.value = false
112             }
113         }
114     }
115
116     fun setPhotoPlant(uri : Uri?, fileExtension: String?=null){
117         imageUri.value = if(uri != null){
118             UriFileExtensions(uri, fileExtension!!)
119         } else {
120             null
121         }
122         Log.i(TAG, "setPhotoPlant: $uri")
123     }
124
125     fun getPhotoProfile() : MutableLiveData<UriFileExtensions>{
126         return imageUri
127     }
128
129     fun getCurrentPlant() : MutableLiveData<PlantModel>{
130         return plantModel
131     }
132 }
```

Kode 22: *View Model* untuk *Add Plant Activity*

12. PROFILE KIT VIEW MODEL

```

1 class ProfileKitViewModel : ViewModel() {
2     private var firestore : FirebaseFirestore = FirebaseFirestore.getInstance()
3     private var currentUserModel : MutableLiveData<UserModel> = MutableLiveData(null)
4     private var currentKitModel : MutableLiveData<KitModel> = MutableLiveData(null)
5     private var currentFarmModel : MutableLiveData<FarmModel> = MutableLiveData(null)
6
7     private var monitoringArray : MutableLiveData<ArrayList<DataMonitoringModel>> =
8         MutableLiveData(null)
9     private var cropsArray : MutableLiveData<ArrayList<CropsModel>> =
10        MutableLiveData(null)
11     private lateinit var kitListener : ListenerRegistration
12     private lateinit var monitoringListener : ListenerRegistration
13     private lateinit var cropsListener : ListenerRegistration
14
15     companion object {
16         const val TAG = "profileKitViewModel"
17     }
18
19     fun setCurrentKit(userModel: UserModel?, farmModel: FarmModel?, kitModel:
20         KitModel?) {
21         userModel?.let { user ->
22             currentUserModel.value = user
23             farmModel?.let {
24                 currentFarmModel.value = it
25             }
26             kitModel?.let { kit ->
27                 currentKitModel.value = kit
28                 getKitUpdate()
29             }
30         }
31         Log.i(TAG, "setCurrentKit: $userModel $kitModel")
32     }
33
34     private fun getKitUpdate() {
35         try{
```

```

33     val db =
34         firestore.collection("farms").document(currentUserModel.value?.farmId!!)
35             .collection("kits").document(currentKitModel.value?.kitId!!)
36
37     kitListener = db.addSnapshotListener { kitSnapshot, error -
38         kitSnapshot?.let {
39             currentKitModel.value = it.toKitModel()
40         }
41
42         error?.let {
43             Log.e(TAG, "Listen kits failed", it)
44         }
45     }
46
47     monitoringListener = db.collection("dataMonitorings")
48         .orderBy("timestamp", Query.Direction.DESCENDING)
49         .addSnapshotListener { monitoringSnapshot, error -
50             monitoringSnapshot?.let {
51                 val monitorings : ArrayList<DataMonitoringModel> = arrayListOf()
52                 for(each in it.documents){
53                     each.toDataMonitoringModel()?.let {
54                         monitorings.add(it)
55                     }
56                 }
57                 monitoringArray.value = monitorings
58             }
59             error?.let {
60                 Log.e(TAG, "Listen monitoring failed", it)
61             }
62         }
63
64     cropsListener = db.collection("crops")
65         .orderBy("timestamp", Query.Direction.DESCENDING)
66         .addSnapshotListener { cropsSnapshot, error -
67             cropsSnapshot?.let {
68                 val crops : ArrayList<CropsModel> = arrayListOf()
69                 for(each in it.documents){
70                     each.toCropsModel()?.let { cropsModel ->
71                         cropsModel.plantId?.let{ plantId ->
72                             firestore.collection("plants").document(plantId)
73                                 .get().addOnSuccessListener {
74                                     cropsModel.plantModel = it.toPlantModel()
75                                 }
76                         crops.add(cropsModel)
77                     }
78                 }
79                 cropsArray.value = crops
80             }
81
82             error?.let {
83                 Log.e(TAG, "Listen crops failed", it)
84             }
85         }
86     } catch (e:Exception){
87         Log.e(TAG, "getKitUpdate: ",e )
88     }
89 }
90
91 fun getCurrentKit() = currentKitModel
92 fun getCurrentMonitoring() = monitoringArray
93 fun getCurrentCrops() = cropsArray
94 fun getCurrentUser() = currentUserModel
95 fun getCurrentFarm() = currentFarmModel
96 }
```

Kode 23: *View Model* untuk *Profile Kit Activity* beserta *Kit Overview*, *Kit Monitoring*, dan *Kit Crops* frgments

13. PROFILE USER VIEW MODEL

```
1  class ProfileUserViewModel : ViewModel(){
2      private var firestore : FirebaseFirestore = Firebase.firebaseio
3      private var userModel : MutableLiveData<UserModel> = MutableLiveData(null)
4      private var plantModels : MutableLiveData<ArrayList<PlantModel>> =
5          MutableLiveData(null)
6      private var allUsers : MutableLiveData<ArrayList<UserModel>> = MutableLiveData(null)
7
8      companion object {
9          const val TAG = "profileUserViewModel"
10     }
11
12     fun setUserModel(userModel: UserModel?){
13         this.userModel.value = userModel
14         try {
15             this.userModel.value?.let { user ->
16                 firestore.collection("plants")
17                     .whereEqualTo("userId", user.uid)
18                     .addSnapshotListener { plantSnapshot, error ->
19                         plantSnapshot?.let {
20                             val plants : ArrayList<PlantModel> = arrayListOf()
21                             for(plantDoc in it.documents){
22                                 plantDoc.toPlantModel()?.let {
23                                     plants.add(it)
24                                 }
25                             }
26                             plantModels.value = plants
27                         }
28                     }
29             } catch (e:Exception){
30                 Log.e(TAG, "setUserModel: ", )
31             }
32         }
33
34         fun getUserModel() : MutableLiveData<UserModel> = userModel
35         fun getUserPosts() : MutableLiveData<ArrayList<PlantModel>> = plantModels
36     }
}
```

Kode 24: *View Model* untuk *Profile User Activity*

14. SEARCH VIEW MODEL

```
1  class SearchViewModel : ViewModel(){
2      private var firestore : FirebaseFirestore = Firebase.firebaseio
3      private var userModels : MutableLiveData<ArrayList<UserModel>> = MutableLiveData()
4      private var plantModels : MutableLiveData<ArrayList<PlantModel>> = MutableLiveData()
5      private var searchUsers : MutableLiveData<ArrayList<UserModel>> = MutableLiveData()
6      private var searchPlants : MutableLiveData<ArrayList<PlantModel>> =
7          MutableLiveData()
8      private var exceptUsers : MutableLiveData<ArrayList<UserModel>> =
9          MutableLiveData(null)
10
11     var isSearchSuccess : MutableLiveData<Boolean> = MutableLiveData(false)
12     var searchError : MutableLiveData<String> = MutableLiveData("")
13
14     companion object {
15         const val TAG = "searchViewModel"
16     }
17     init {
18         getAllUsers(null)
19     }
20
21     fun setExceptUsers(userModels : ArrayList<UserModel>?){
22         exceptUsers.value = userModels
23         Log.i(TAG, "setExceptUsers: ${exceptUsers.value}")
24     }
}
```

```

23
24     fun getAllUsers() : MutableLiveData<ArrayList<UserModel>> = userModels
25
26     fun getAllUsers(lastKey : String? = null){
27         val db = firestore.collection("users")
28         try {
29             db.addSnapshotListener { snapshot, error ->
30                 snapshot?.let {
31                     val users : ArrayList<UserModel> = arrayListOf()
32                     for(plant in it.documents){
33                         plant.toUserModel()?.let { it1 -> users.add(it1) }
34                     }
35                     userModels.value = users
36                     searchUsers(lastKey)
37                 }
38
39                 error?.let {
40                     Log.e(TAG, "Listen data users failed", it)
41                     return@addSnapshotListener
42                 }
43             }
44         } catch (e:Exception){
45             Log.e(TAG, "Error get plants", e)
46         }
47     }
48     fun getAllPlants(lastKey : String? = null){
49         val db = firestore.collection("plants")
50         try {
51             db.addSnapshotListener { snapshot, error ->
52                 snapshot?.let {
53                     val plants : ArrayList<PlantModel> = arrayListOf()
54                     for(plant in it.documents){
55                         plant.toPlantModel()?.let { it1 -> plants.add(it1) }
56                     }
57                     plantModels.value = plants
58                     searchPlants(lastKey)
59                 }
60
61                 error?.let {
62                     Log.e(TAG, "Listen data plants failed", it)
63                     return@addSnapshotListener
64                 }
65             }
66         } catch (e:Exception){
67             Log.e(TAG, "Error get plants", e)
68         }
69     }
70
71     fun getUsers() : MutableLiveData<ArrayList<UserModel>> = searchUsers
72     fun getPlants() : MutableLiveData<ArrayList<PlantModel>> = searchPlants
73
74     fun getUser(position: Int) : UserModel? {
75         return searchUsers.value?.get(position)
76     }
77     fun getPlant(position:Int) : PlantModel?{
78         return searchPlants.value?.get(position)
79     }
80
81     fun searchUsers(key:String?) {
82         if(key != null){
83             val results : ArrayList<UserModel> = arrayListOf()
84             userModels.value?.let {
85                 for (user in it) {
86                     if (user.name?.contains(key) == true || user.email?.contains(key)
87                         == true) {
88                         exceptUsers.value?.let {
89                             for (exUser in it) {
90                                 if (exUser.uid != user.uid){ /// user.role !=
91                                     Role.OWNER.toString()) {
92                                     results.add(user)

```

```

91             }
92         }
93     } ?: kotlin.run {
94         results.add(user)
95     }
96 }
97 searchUsers.value = results
98 }
99 }
100 }
101 else {
102     userModels.value?.let { allUsers ->
103         exceptUsers.value?.let {
104             val results : ArrayList<UserModel> = arrayListOf()
105             for(user in allUsers) {
106                 for (exUser in it) {
107                     if (exUser.uid != user.uid) {
108                         results.add(user)
109                     }
110                 }
111             }
112             searchUsers.value = results
113         } ?: kotlin.run {
114             searchUsers.value = userModels.value
115         }
116     }
117 }
118 }
119 fun searchPlants(key:String?) {
120     if(key != null && plantModels.value != null){
121         val results : ArrayList<PlantModel> = arrayListOf()
122         plantModels.value?.let {
123             for (user in it) {
124                 if (user.name?.contains(key) == true) {
125                     results.add(user)
126                 }
127             }
128             searchPlants.value = results
129         }
130     }
131     else {
132         searchPlants.value = plantModels.value
133     }
134 }
135 }
136 }

```

Kode 25: *View Model* untuk *Search Activity*

15. FEEDBACK VIEW MODEL

```

1 class FeedbackViewModel : ViewModel(){
2     private var auth : FirebaseAuth = FirebaseAuth.getInstance()
3     private val db = FirebaseFirestore.getInstance().collection("feedbacks")
4     private var ref : DocumentReference = db.document()
5     private var feedbackModel : MutableLiveData<FeedbackModel> =
6         MutableLiveData(FeedbackModel())
7     private var isNotEmpty : MutableLiveData<Boolean> = MutableLiveData(false)
8     var isFeedbackSubmit : MutableLiveData<Boolean> = MutableLiveData(false)
9     var feedbackSubmitError : MutableLiveData<String> = MutableLiveData("")
10
11    companion object {
12        const val TAG = "feedback"
13    }
14
15    fun checkNotEmpty(boolean: Boolean) : MutableLiveData<Boolean> {
16        isNotEmpty.value = boolean
17        return isNotEmpty
18    }

```

```
18     fun setRating(rating : Float) : MutableLiveData<FeedbackModel>? {
19         feedbackModel.value?.rating = rating
20         return feedbackModel
21     }
22
23
24     fun getRating() = feedbackModel.value?.rating
25
26     fun submitFeedback(content:String){
27         try{
28             feedbackModel.value?.let {
29                 it.feedbackId = ref.id
30                 it.content = content
31                 it.userId = auth.uid
32             }
33
34             db.document(ref.id).set(feedbackModel.value!!.toHashMap()).addOnCompleteListener {
35                 it.tener
36                 if(it.isSuccessful){
37                     isFeedbackSubmit.value = true
38                 } else {
39                     feedbackSubmitError.value = it.exception.toString()
40                     isFeedbackSubmit.value = false
41                 }
42             }
43         } catch (e:Exception){
44             Log.e(TAG, "Error submit feedback", e)
45         }
46     }
47 }
```

Kode 26: *View Model* untuk *Feedback Activity*

16. SHOW RECYCLER VIEW MODEL

```
1 class ShowRecyclerViewModel : ViewModel() {
2     private var auth : FirebaseAuth = Firebase.auth
3     private var firestore : FirebaseFirestore = Firebase.firestore
4     private var plantModels : MutableLiveData<ArrayList<PlantModel>> =
5         MutableLiveData(null)
6     private var userModel : MutableLiveData<UserModel> = MutableLiveData(null)
7
8     fun setCurrentPosts(plantModel: ArrayList<PlantModel>?, userModel : UserModel?) {
9         this.userModel.value = userModel
10        plantModel?.let {
11            val plants : ArrayList<PlantModel> = arrayListOf()
12            it.forEach {
13                if(it.userId == auth.uid){
14                    plants.add(it)
15                }
16            }
17            plantModels.value = plants
18            getPlantsUpdate()
19        } ?: kotlin.run {
20            plantModels.value = arrayListOf()
21            getPlantsUpdate()
22        }
23    }
24
25    fun getCurrentPosts() : MutableLiveData<ArrayList<PlantModel>> = plantModels
26    fun getCurrentUser() : MutableLiveData<UserModel> = userModel
27
28    var isPostDeleted : MutableLiveData<Boolean?> = MutableLiveData(null)
29    var deletePostError : MutableLiveData<String> = MutableLiveData(null)
30
31    fun getPost(position: Int) : PlantModel?{
32        return plantModels.value?.get(position)
33    }
34
```

```

33
34     fun deletePost(position: Int){
35         isPostDeleted.value = null
36         val plantModel = getPost(position)
37         val plantId = plantModel?.plantId
38         try {
39             if(plantModel?.userId == auth.uid) {
40                 firestore.collection("plants").document(plantId!!)
41                     .delete().addOnCompleteListener {
42                         if (it.isSuccessful) {
43                             isPostDeleted.value = true
44                         } else {
45                             isPostDeleted.value = false
46                             deletePostError.value = it.exception.toString()
47                             Log.e(MainViewModel.TAG, "deletePost $plantId\n"
48                                 + it.exception)
49                         }
50                     }
51             } catch (e:Exception){
52                 Log.e(MainViewModel.TAG, "deletePost $plantId failed", e)
53             }
54         }
55
56         private fun getPlantsUpdate(){
57             val db = firestore.collection("plants")
58                 .whereEqualTo("userId", auth.uid)
59                 .orderBy("timestamp", Query.Direction.DESCENDING)
60             try {
61                 db.addSnapshotListener { plantsSnapshot, error ->
62                     plantsSnapshot?.let {
63                         val plants : ArrayList<PlantModel> = arrayListOf()
64                         for(plantDocument in plantsSnapshot.documents){
65                             plantDocument?.let { plant ->
66                                 plant.toPlantModel()?.let { plantModel ->
67                                     plants.add(plantModel)
68                                 }
69                             }
70                         }
71                         plantModels.value = plants
72                         Log.i(MainViewModel.TAG, "getPlantsUpdate:
73                             + plantModels.value?.size")
74                     }
75                     error?.let {
76                         Log.e(MainViewModel.TAG, "Listen plants failed", it)
77                     }
78                 }
79             } catch (e:Exception){
80                 Log.e(MainViewModel.TAG, "real time plant error", e)
81             }
82         }
83     }

```

Kode 27: View Model untuk Show Recycler Activity

E. VIEW LAYER (ACTIVITY AND FRAGMENT)

1. SPLASH SCREEN ACTIVITY

```

1  class SplashScreenActivity : AppCompatActivity() {
2      private lateinit var binding : ActivitySplashScreenBinding
3      private lateinit var viewModel : SplashScreenViewModel
4      private var auth : FirebaseAuth = FirebaseAuth.auth
5
6      companion object{
7          const val TAG = "splashScreenActivity"
8      }

```

```

9
10    override fun onCreate(savedInstanceState: Bundle?) {
11        super.onCreate(savedInstanceState)
12        binding = ActivitySplashScreenBinding.inflate(layoutInflater)
13        setContentView(binding.root)
14        viewModel = ViewModelProvider(this).get(SplashScreenViewModel::class.java)
15        RequestPermission().requestAllPermissions(this)
16    }
17
18    fun splashIsDone(){
19        if(auth.currentUser != null){
20            viewModel.getCurrentData()
21            viewModel.isCurrentUserExist.observe(this,{

22                when(it){
23                    true -> {
24                        Log.i(TAG, "user ${viewModel.currentUserModel.value}")
25                        viewModel.isCurrentFarmExist.observe(this, {
26                            when(it){
27                                true -> {
28                                    Log.i(TAG, "farm
29                                     ${viewModel.currentFarmModel.value}")
30                                    val intent = Intent(this, MainActivity::class.java)
31                                    intent.putExtra(BuildConfig.CURRENT_USER,
32                                     viewModel.currentUserModel.value)
33                                    intent.putExtra(BuildConfig.CURRENT_FARM,
34                                     viewModel.currentFarmModel.value)
35                                    startActivity(intent)
36                                    finish()
37                                }
38                                false -> {
39                                    Log.i(TAG, "currentFarm not exist")
40                                    if(Role.getType(viewModel.currentUserModel.value?.r
41                                     ole!!) ==
42                                     Role.OWNER){
43                                         val intent = Intent(this,
44                                         CreateProfileActivity::class.java)
45                                         intent.putExtra(BuildConfig.SELECTED_USER,
46                                         viewModel.currentUserModel.value)
47                                         startActivity(intent)
48                                         finish()
49                                     } else {
50                                         Toast.makeText(this, "wait for owner add you",
51                                         Toast.LENGTH_SHORT).show()
52                                         Log.i(TAG, "wait for owner add you")
53                                     }
54                                 }
55                                 null -> {
56                                     Log.i(TAG, "currentFarm loading")
57                                 }
58                             }
59                         })
60                     }
61                 }
62             }
63         } else {
64             val handler = Handler(Looper.getMainLooper())
65             handler.postDelayed({
66                 startActivity(Intent(this, SignInActivity::class.java))
67             }, 3000)
68         }
69     }
70 }
```

```
71 }
```

Kode 28: Source Code untuk Splash Screen Activity

2. SIGN IN ACTIVITY

```
1  class SignInActivity : AppCompatActivity(), View.OnClickListener, TextWatcher,
2      → CompoundButton.OnCheckedChangeListener {
3          private lateinit var binding: ActivitySignInBinding
4          private lateinit var viewModel: SignInViewModel
5          private lateinit var utility: ViewUtility
6          private lateinit var editTexts: ArrayList<TextInputEditText>
7          private lateinit var viewsAsButton: ArrayList<View>
8
9          companion object{
10              const val TAG = "signIn"
11          }
12
13          override fun onCreate(savedInstanceState: Bundle?) {
14              super.onCreate(savedInstanceState)
15              binding = ActivitySignInBinding.inflate(layoutInflater)
16              setContentView(binding.root)
17
18              viewModel = ViewModelProvider(this).get(SignInViewModel::class.java)
19              binding.apply {
20                  editTexts = arrayListOf(signinEmail, signinPassword)
21                  viewsAsButton = arrayListOf(signinSubmit,
22                      signinSignUp,
23                      signinForgotPassword)
24                  utility = ViewUtility(
25                      context = this@SignInActivity,
26                      circularProgressBar = signinSubmit,
27                      TextInputEditTexts = editTexts,
28                      viewsAsButton = viewsAsButton,
29                      checkBoxes = arrayListOf(signinRemember),
30                      actionBar = supportActionBar)
31
32                  signinRemember.setOnCheckedChangeListener(this@SignInActivity)
33                  viewsAsButton.forEach { it.setOnClickListener(this@SignInActivity) }
34                  editTexts.forEach { it.addTextChangedListener(this@SignInActivity) }
35                  checkEmpty()
36              }
37
38          override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
39              if (currentFocus != null) {
40                  val imm: InputMethodManager =
41                      getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
42                  imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
43              }
44              return super.dispatchTouchEvent(ev)
45          }
46
47          override fun onClick(v: View?) {
48              when(v){
49                  binding.signinSubmit -> {
50                      utility.isLoading = true
51                      viewModel.signIn(binding.signinEmail.text.toString(),
52                          → binding.signinPassword.text.toString())
53                      viewModel.isUserSignIn.observe(this@SignInActivity, {
54                          if(it){
55                              DatabaseHandler(this).signIn(viewModel.getDataModel())
56                              utility.isLoading = false
57                              startActivity(Intent(this@SignInActivity,
58                                  → MainActivity::class.java))
59                              finish()
60                          } else {
61                              utility.isLoading = false
62                              viewModel.signInError.observe(this, {
```

```

61             it?.let {
62                 if (it.isNotEmpty()) {
63                     Toast.makeText(this@SignInActivity, it,
64                         → Toast.LENGTH_SHORT)
65                         .show()
66                     Log.i(TAG, it)
67                     viewModel.signInError.value = null
68                 }
69             }
70         }
71     }
72 }
73 binding.signinSignUp -> startActivity(Intent(this,
74     → SignUpActivity::class.java))
75 binding.signinForgotPassword -> startActivity(Intent(this,
76     → ForgetPasswordActivity::class.java))
77 }
78
79 override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after:
80     → Int) {}
81
82 override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
83     checkEmpty()
84 }
85
86 override fun afterTextChanged(s: Editable?) {}
87
88 private fun checkEmpty() {
89     viewModel.checkNotEmpty(utility.isEmptys(editTexts)).observe(this, {
90         binding.signinSubmit.isEnabled = it
91     })
92 }
93
94 override fun onCheckedChanged(buttonView: CompoundButton?, isChecked: Boolean) {
95     viewModel.setRememberChecked(isChecked)
96 }
97 }
```

Kode 29: Source Code untuk Sign In Activity

3. SIGN UP ACTIVITY

```

1 class SignUpActivity : AppCompatActivity(), View.OnClickListener, TextWatcher {
2     private lateinit var binding : ActivitySignUpBinding
3     private lateinit var viewModel : SignUpViewModel
4     private lateinit var utility : ViewUtility
5     private lateinit var editTexts : ArrayList<TextInputEditText>
6     private lateinit var viewsAsButton : ArrayList<View>
7
8     companion object{
9         const val TAG = "signUp"
10    }
11
12    override fun onCreate(savedInstanceState: Bundle?) {
13        super.onCreate(savedInstanceState)
14        binding = ActivitySignUpBinding.inflate(layoutInflater)
15        setContentView(binding.root)
16
17        viewModel = ViewModelProvider(this).get(SignUpViewModel::class.java)
18        supportActionBar?.title = null
19        supportActionBar?.elevation= Of
20        supportActionBar?.setDisplayHomeAsUpEnabled(true)
21        supportActionBar?.setDisplayShowHomeEnabled(false)
22
23        binding.apply {
24            viewsAsButton = arrayListOf(signupSignIn, signupSubmit)
25            editTexts = arrayListOf(signupEmail, signupPassword, signupConfirmPassword)
26        }
27    }
28 }
```

```

26     utility = ViewUtility(
27         context = this@SignUpActivity,
28         circularProgressBar = signupSubmit,
29         textInputEditTexts = editTexts,
30         viewsAsButton = viewsAsButton,
31         actionBar = supportActionBar
32     )
33
34     viewsAsButton.forEach { it.setOnClickListener(this@SignUpActivity) }
35     editTexts.forEach { it.addTextChangedListener(this@SignUpActivity) }
36     checkEmpty()
37 }
38
39
40     override fun onSupportNavigateUp(): Boolean {
41         super.onBackPressed()
42         return true
43     }
44
45     override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
46         if (currentFocus != null) {
47             val imm: InputMethodManager =
48                 getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
49             imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
50         }
51         return super.dispatchTouchEvent(ev)
52     }
53
54     override fun onClick(v: View?) {
55         when(v){
56             binding.signupSubmit -> {
57                 utility.isLoading = true
58                 viewModel.signUp(binding.signupEmail.text.toString(),
59                             binding.signupPassword.text.toString())
59                 viewModel.isUserSignUp.observe(this@SignUpActivity, {
60                     if(it){
61                         utility.isLoading = false
62                         startActivity(Intent(this,
63                                     CreateProfileActivity::class.java))
64                         finish()
65                     } else {
66                         viewModel.signUpError.observe(this, {
67                             it?.let {
68                                 if (it.isNotEmpty()) {
69                                     Toast.makeText(this@SignUpActivity, it,
70                                     Toast.LENGTH_SHORT)
71                                     .show()
72                                     Log.i(TAG, it)
73                                     viewModel.signUpError.value = ""
74                                     utility.isLoading = false
75                                 }
76                             }
77                         })
78                     }
79                 })
80             binding.signupSignIn -> super.onBackPressed()
81         }
82     }
83
84     override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after:
85                                     Int) {}
86
87     override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
88         checkEmpty()
89     }
90
91     override fun afterTextChanged(s: Editable?) {}

```

```

92     private fun checkEmpty() {
93         val passwordMatch = binding.signupPassword.text.toString() ==
94             binding.signupConfirmPassword.text.toString()
95         binding.signupConfirmPasswordLayout.error = if(passwordMatch){
96             null
97         } else {
98             "Confirm password should be same"
99         }
100    viewModel.apply {
101        checkNotEmpty(utility.isEmptys(editTexts) &&
102            passwordMatch).observe(this@SignUpActivity, {
103            binding.signupSubmit.isEnabled = it
104        })
105    }
106 }

```

Kode 30: *Source Code* untuk *Sign Up Activity*

4. FORGET PASSWORD ACTIVITY

```

1  class ForgetPasswordActivity : AppCompatActivity(), View.OnClickListener, TextWatcher {
2     private lateinit var binding : ActivityForgetPasswordBinding
3     private lateinit var viewModel : ForgetPasswordViewModel
4     private lateinit var utility : ViewUtility
5     private lateinit var viewsAsButton : ArrayList<View>
6
7     companion object{
8         const val TAG = "forgetPasswordActivity"
9     }
10
11    override fun onCreate(savedInstanceState: Bundle?) {
12        super.onCreate(savedInstanceState)
13        binding = ActivityForgetPasswordBinding.inflate(layoutInflater)
14        setContentView(binding.root)
15
16        viewModel = ViewModelProvider(this).get(ForgetPasswordViewModel::class.java)
17        supportActionBar?.title = null
18        supportActionBar?.elevation= 0f
19        supportActionBar?.setDisplayHomeAsUpEnabled(true)
20        supportActionBar?.setDisplayShowHomeEnabled(false)
21
22        binding.apply {
23            viewsAsButton = arrayListOf(forgetSubmit, forgetResend)
24            utility = ViewUtility(
25                context = this@ForgetPasswordActivity,
26                circularProgressButton = forgetSubmit,
27                textInputEditTexts = arrayListOf(forgetEmail),
28                viewsAsButton = viewsAsButton,
29                actionBar = supportActionBar
30            )
31            utility.onLoadingChangeListener{
32                if(it){
33                    binding.forgetSuccess.visibility = View.GONE
34                }
35            }
36
37            viewsAsButton.forEach { it.setOnClickListener(this@ForgetPasswordActivity) }
38            getEmail.addTextChangedListener(this@ForgetPasswordActivity)
39            checkEmpty()
40        }
41    }
42
43    override fun onSupportNavigateUp(): Boolean {
44        super.onBackPressed()
45        return true
46    }
47

```

```

48     override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
49         if (currentFocus != null) {
50             val imm: InputMethodManager =
51                 getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
52             imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
53         }
54         return super.dispatchTouchEvent(ev)
55     }
56
57     override fun onClick(v: View?) {
58         when(v){
59             binding.forgetSubmit, binding.forgetResend -> {
60                 utility.isLoading = true
61                 viewModel.sendEmailReset(binding.getEmail.text.toString())
62                 viewModel.isUserForgotPassword.observe(this, {
63                     if(it){
64                         utility.isLoading = false
65                         handleDynamicLink()
66                     } else {
67                         utility.isLoading = false
68                         viewModel.getPasswordError.observe(this, {
69                             if (it.isNotEmpty()) {
70                                 Toast.makeText(this@ForgetPasswordActivity, it,
71                                     &gt; Toast.LENGTH_SHORT)
72                                     .show()
73                                 viewModel.getPasswordError.value = ""
74                             }
75                         })
76                     }
77                 })
78             }
79         }
80
81     override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after:
82         &gt; Int) {}
83
84     override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
85         checkEmpty()
86     }
87
88     override fun afterTextChanged(s: Editable?) {}
89
90     private fun checkEmpty() {
91         viewModel.checkNotEmpty(utility.isEmpty(binding.getEmail)).observe(this, {
92             binding.forgetSubmit.isEnabled = it
93         })
94     }
95
96     fun handleDynamicLink() {
97         // FirebaseDynamicLinks.getInstance().getDynamicLink(intent)
98         // .addOnSuccessListener(this) { pendingDynamicLinkData:
99         // &gt; PendingDynamicLinkData ->
100         //     Log.i("forgetPasswordActivity", "Dynamic link detected")
101         //     val oobCode =
102         //         pendingDynamicLinkData.link!!.getQueryParameter("oobCode")
103         //     if (oobCode != null) {
104         //         auth.checkActionCode(oobCode).addOnSuccessListener { result ->
105         //             when (result.getOperation()) {
106         //                 ActionCodeResult.VERIFY_EMAIL -> {
107         //                     auth.applyActionCode(oobCode)
108         //                     .addOnSuccessListener { resultCode ->
109         //                         Log.i("forgetPasswordActivity", "Verified
110         // &gt; email")
111         //                         finish()
112         //                     }.addOnFailureListener { resultCode ->
113         //                         Log.w(
114         //                             "forgetPasswordActivity",
115         //                             "Failed to Verified Email",
116         //                             resultCode
117         //                         )
118         //                     }
119         //                 }
120         //             }
121         //         }
122         //     }
123         // }
124     }

```

```

114     /**
115      *
116      */
117     /**
118      */
119     /**
120      */
121     /**
122      */
123     /**
124      */
125     /**
126      */
127     /**
128      */
129     /**
130      */
131     /**
132      */
133     /**
134      */
135 }

```

Kode 31: Source Code untuk Forget Password Activity

5. CREATE PROFILE ACTIVITY

```

1  class CreateProfileActivity : AppCompatActivity() {
2      private lateinit var binding : ActivityCreateProfileBinding
3      lateinit var viewModel : CreateProfileViewModel
4
5      companion object{
6          const val TAG = "createProfile"
7      }
8
9      override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         binding = ActivityCreateProfileBinding.inflate(layoutInflater)
12         setContentView(binding.root)
13
14         viewModel = ViewModelProvider(this).get(CreateProfileViewModel::class.java)
15         viewModel.setCurrentUser(intent.getParcelableExtra<UserModel>(BuildConfig.CURRE_
16             ↪ NT_USER))
17     }
18
19     override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
20         if (currentFocus != null) {
21             val imm: InputMethodManager =
22                 getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
23             imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
24         }
25         return super.dispatchTouchEvent(ev)
26     }
27
28     val startForResult =
29         registerForActivityResult(ActivityResultContracts.StartActivityForResult()) {
30         data -
31             Log.i(TAG, "${data}")
32             try{
33                 if(data?.resultCode == Activity.RESULT_OK){
34                     CropImage.getActivityResult(data.data)?.let{
35                         val mimeTypeMap = MimeTypeMap.getSingleton()
36                         val fileExtension = mimeTypeMap.getExtensionFromMimeType(contentRes_
37                             ↪ olver.getType(it.uriContent!!))
38                         viewModel.setPhotoProfile(it.uriContent!!, fileExtension)
39                     }
40                 }
41             }
42         }

```

```

37         data.data?.getParcelableExtra<UserModel>(BuildConfig.SELECTED_USER)?.let {
38             it
39             viewModel.addStaff(it)
40         }
41     } catch (e: Exception){
42         Toast.makeText(this, e.message.toString(), Toast.LENGTH_SHORT).show()
43     }
44 }
45
46 fun changeImageProfile(){
47     val intent = CropImage.activity()
48         .setActivityTitle("Edit Photo")
49         .setAspectRatio(1,1)
50         .setGuidelines(CropImageView.Guidelines.ON)
51         .setAutoZoomEnabled(true)
52         .setAllowFlipping(false)
53         .getIntent(this)
54     startForResult.launch(intent)
55 }
56
57 fun searchUser(){
58     val intent = Intent(this, SearchActivity::class.java)
59     intent.putExtra("search", TAG)
60     startForResult.launch(intent)
61 }
62
63 }

```

Kode 32: *Source Code* untuk *Create Profile Activity* beserta *fragments* terkait

a) CREATE USER FRAGMENT

```

1  class CreateUserFragment : Fragment(), View.OnClickListener,
2   → SegmentedButtonGroup.OnPositionChangedListener, TextWatcher {
3     private var _binding : FragmentCreateUserBinding? = null
4     private val binding get() = _binding!!
5     private lateinit var viewModel : CreateProfileViewModel
6     private lateinit var utility: ViewUtility
7     private lateinit var editTexts : ArrayList<TextInputEditText>
8     private lateinit var viewsAsButton : ArrayList<View>
9
9     companion object {
10         const val TAG = "createUserFragment"
11     }
12
13     override fun onCreateView(
14         inflater: LayoutInflater, container: ViewGroup?,
15         savedInstanceState: Bundle?
16     ): View {
17         _binding = FragmentCreateUserBinding.inflate(inflater, container, false)
18         return binding.root
19     }
20
21     override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
22         super.onViewCreated(view, savedInstanceState)
23         requireActivity().title = getString(R.string.create_profile)
24         viewModel = (requireActivity() as CreateProfileActivity).viewModel
25
26         binding.apply {
27             viewsAsButton = arrayListOf(roleOwner,
28                 createUserDOB,
29                 createUserSubmit,
30                 createUserEditPhoto,
31                 createUserPhoto)
32             editTexts = arrayListOf(
33                 createUserName,
34                 createUserPhone,

```

```

35             createUserAddress,
36             createUserBio)
37     utility = ViewUtility(
38         context = requireContext(),
39         circularProgressBar = createUserSubmit,
40         textInputEditTexts = editTexts,
41         viewsAsButton = viewsAsButton,
42         actionBar = (requireActivity() as
43             CreateProfileActivity).supportActionBar)
44     utility.onLoadingChangeListener {
45         binding.roleOwner.isClickable = it
46         binding.roleStaff.isClickable = it
47     }
48
49     editTexts.forEach { it.addTextChangedListener(this@CreateUserFragment) }
50     viewsAsButton.forEach { it.setOnClickListener(this@CreateUserFragment) }
51     createUserGender.onPositionChangedListener = this@CreateUserFragment
52     createUserPhoto.setOnLongClickListener { createUserEditPhoto.performClick()
53         -->
54     }
55     checkEmpty()
56
57     createUserEmail.setText(Firebase.auth.currentUser?.email)
58     roleOwner.performClick()
59 }
60
61 viewModel.getPhotoProfile().observe(requireActivity(), {
62     it?.let {
63         Glide.with(this)
64             .load(it.uri)
65             .circleCrop()
66             .into(binding.createUserPhoto)
67     }
68 })
69
70 viewModel.isUserCreated.observe(viewLifecycleOwner, {
71     if(it){
72         utility.isLoading = false
73         Toast.makeText(requireContext(), "User created",
74             -->
75             Toast.LENGTH_SHORT).show()
76         if (Role.getType(viewModel.getCurrentUserModel()?)?.role!!) ==
77             -->
78             Role.OWNER) {
79             Navigation.findNavController(binding.root).navigate(R.id.action_createUserFragment_to_createFarmFragment)
80         } else {
81             val intent = Intent(requireContext(), MainActivity::class.java)
82             intent.putExtra(BuildConfig.CURRENT_USER,
83                 -->
84                 viewModel.getCurrentUserModel())
85             startActivity(intent)
86             requireActivity().finish()
87         }
88     } else {
89         utility.isLoading = false
90         viewModel.createProfileError.observe(viewLifecycleOwner, {
91             if(it.isNotEmpty()){
92                 Toast.makeText(requireContext(), it, Toast.LENGTH_SHORT).show()
93                 Log.i(TAG, it)
94                 viewModel.createProfileError.value = ""
95             }
96         })
97     }
98 })
99
100 override fun onClick(v: View?) {
101     when(v){
102         binding.roleOwner -> setRoleType(Role.OWNER)
103         binding.roleStaff -> setRoleType(Role.STAFF)
104         binding.createUserDOB -> {
105             val calendar = Calendar.getInstance(TimeZone.getTimeZone("WIB"))
106             calendar[Calendar.MONTH] = Calendar.DECEMBER

```

```

99         val constraintBuilder = CalendarConstraints.Builder()
100        constraintBuilder.setValidator(DateValidatorPointBackward.now()).setEnd |
101           (calendar.timeInMillis)
102
103        val datePicker = MaterialDatePicker.Builder.datePicker()
104            .setTitleText("Date of Birth")
105            .setSelection(viewModel.getDOB())
106            .setInputMode(MaterialDatePicker.INPUT_MODE_CALENDAR)
107            .setCalendarConstraints(constraintBuilder.build())
108            .build()
109        datePicker.show(childFragmentManager, "DATE_PICKER")
110        datePicker.addOnPositiveButtonClickListener{
111            binding.createUserDOB.setText(viewModel.setDOB(datePicker.selection))
112            checkEmpty()
113        }
114        datePicker.addOnNegativeButtonClickListener{}
115        datePicker.isCancelable = false
116
117        binding.createUserEditPhoto ->
118            utility.openEditPhoto(binding.createUserPhoto, ProfileType.USER)
119        binding.createUserPhoto -> utility.openImage(binding.createUserPhoto)
120        binding.createUserSubmit -> {
121            utility.isLoading = true
122            viewModel.createUserProfile(
123                binding.createUserName.text.toString(),
124                binding.createUserPhone.text.toString(),
125                binding.createUserAddress.text.toString(),
126                binding.createUserBio.text.toString()
127            )
128        }
129
130    override fun onPositionChanged(position: Int) {
131        when(position){
132            0 -> viewModel.setGender(Gender.MALE)
133            1 -> viewModel.setGender(Gender.FEMALE)
134        }
135        checkEmpty()
136    }
137
138    private fun setRoleType(role : Role){
139        viewModel.setRole(role)
140        when(role){
141            Role.OWNER -> {
142                binding.roleOwner.isChecked = true
143                binding.roleStaff.isChecked = false
144                binding.createUserRoleDescription.text =
145                    getString(R.string.owner_description)
146            }
147            Role.STAFF -> {
148                binding.roleStaff.isChecked = true
149                binding.roleOwner.isChecked = false
150                binding.createUserRoleDescription.text =
151                    getString(R.string.staff_description)
152            }
153        }
154        checkEmpty()
155    }
156
157    override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after:
158        Int) {}
159
160    override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
161        checkEmpty()
162    }
163
164    override fun afterTextChanged(s: Editable?) {}

```

```

163     private fun checkEmpty() {
164         val checkerEditText = arrayListOf<TextInputEditText>(
165             binding.createUserName,
166             binding.createUserPhone,
167             binding.createUserAddress,
168             binding.createUserDOB)
169         viewModel.checkNotNull(
170             utility.isEmpties(checkerEditText) && (binding.roleOwner.isChecked)
171         ).observe(viewLifecycleOwner, {
172             binding.createUserSubmit.isEnabled = it
173         })
174     }
175 }
```

Kode 33: *Source Code untuk Create User Fragment*

b) CREATE FARM FRAGMENT

```

1  class CreateFarmFragment : Fragment(), View.OnClickListener, TextWatcher {
2      private var _binding : FragmentCreateFarmBinding? = null
3      private val binding get() = _binding!!
4      private lateinit var viewModel : CreateProfileViewModel
5      private lateinit var utility: ViewUtility
6      private lateinit var editTexts : ArrayList<TextInputEditText>
7
8      companion object {
9          const val TAG = "createFarm"
10     }
11
12     override fun onCreateView(
13         inflater: LayoutInflater, container: ViewGroup?,
14         savedInstanceState: Bundle?
15     ): View {
16         _binding = FragmentCreateFarmBinding.inflate(inflater, container, false)
17         return binding.root
18     }
19
20     override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
21         super.onViewCreated(view, savedInstanceState)
22         requireActivity().title = getString(R.string.create_farm)
23         viewModel =
24             ViewModelProvider(requireActivity()).get(CreateProfileViewModel::class.java)
25         binding.apply {
26             editTexts.visibility = View.GONE
27             editTexts = arrayListOf(
28                 createFarmName,
29                 createFarmLoc,
30                 createFarmDesc
31             )
32             utility = ViewUtility(
33                 context = requireContext(),
34                 circularProgressButton = createFarmSubmit,
35                 TextInputEditTexts = editTexts,
36                 actionBar = (requireActivity() as
37                     CreateProfileActivity).supportActionBar
38             )
39
40             createFarmSubmit.setOnClickListener(this@CreateFarmFragment)
41             editTexts.forEach { it.addTextChangedListener(this@CreateFarmFragment) }
42             checkEmpty()
43         }
44
45         val mapFragment = childFragmentManager.findFragmentById(R.id.map) as
46             SupportMapFragment?
47         mapFragment?.getMapAsync(callback)
48     }
49
50     override fun onClick(v: View?) {
51         when(v){
```

```

49     binding.createFarmSubmit -> {
50         utility.isLoading = true
51         viewModel.createFarmProfile(
52             binding.createFarmName.text.toString(),
53             binding.createFarmDesc.text.toString(),
54             binding.createFarmLoc.text.toString()
55         )
56         viewModel.isFarmCreated.observe(this, {
57             if(it){
58                 utility.isLoading = false
59                 Toast.makeText(requireContext(), "Farm created",
60                     Toast.LENGTH_SHORT).show()
61                 val intent = Intent(requireContext(), MainActivity::class.java)
62                 intent.putExtra(BuildConfig.CURRENT_USER,
63                     viewModel.getCurrentUserModel())
64                 intent.putExtra(BuildConfig.CURRENT_FARM,
65                     viewModel.getCurrentFarmModel())
66                 startActivity(intent)
67                 Navigation.findNavController(binding.root).navigate(R.id.action_
68                     _createGardenFragment_to_createStaffFragment)
69             } else {
70                 utility.isLoading = false
71                 viewModel.createProfileError.observe(this, {
72                     if(it.isNotEmpty()){
73                         Toast.makeText(requireContext(), it,
74                             Toast.LENGTH_SHORT).show()
75                         Log.i(TAG, it)
76                         viewModel.createProfileError.value = ""
77                     }
78                 })
79             }
80         }
81         private val callback = OnMapReadyCallback { googleMap ->
82             val sydney = LatLng(-34.0, 151.0)
83             googleMap.addMarker(MarkerOptions().position(sydney).title("Marker in Sydney"))
84             googleMap.moveCamera(CameraUpdateFactory.newLatLng(sydney))
85             //todo : Maps. posisi awal sesuai current location, tambahin zoom
86         }
87         override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after:
88             Int) {}
89         override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
90             checkEmpty()
91         }
92         override fun afterTextChanged(s: Editable?) {}
93         private fun checkEmpty() {
94             viewModel.checkNotEmpty(utility.isEmptys(editTexts)).observe(viewLifecycleOwner, {
95                 r,
96                 {
97                     binding.createFarmSubmit.isEnabled = it
98                 }
99             })
100        }
}

```

Kode 34: Source Code untuk Create Farm Fragment

6. MAIN ACTIVITY

```

1 class MainActivity : AppCompatActivity(), View.OnClickListener,
2     SwipeRefreshLayout.OnRefreshListener,
3     NavigationView.OnNavigationItemSelectedListener {

```

```

4     private lateinit var binding : ActivityMainBinding
5     private lateinit var viewModel : MainViewModel
6     private lateinit var navHostFragment : NavHostFragment
7     private lateinit var navController : NavController
8     private lateinit var actionBarToggle : ActionBarDrawerToggle
9     lateinit var swipeRefresh: SwipeRefreshLayout
10    lateinit var drawerLayout : DrawerLayout
11
12    companion object{
13        const val TAG = "mainActivity"
14    }
15
16    override fun onCreate(savedInstanceState: Bundle?) {
17        super.onCreate(savedInstanceState)
18        binding = ActivityMainBinding.inflate(layoutInflater)
19        setContentView(binding.root)
20
21        viewModel = ViewModelProvider(this).get(MainViewModel::class.java)
22        viewModel.setCurrentData(
23            intent.getParcelableExtra<UserModel>(BuildConfig.CURRENT_USER),
24            intent.getParcelableExtra<FarmModel>(BuildConfig.CURRENT_FARM))
25
26        navHostFragment =
27            supportFragmentManager.findFragmentById(R.id.mainFragmentContainer) as
28            NavHostFragment
29        navController = navHostFragment.navController
30
31        binding.apply {
32            mainBottomNavigation.setupWithNavController(navController)
33            mainVersion.text = BuildConfig.VERSION_NAME
34            swipeRefresh = mainSwipeRefresh
35            mainAddFragment.setOnClickListener(this@MainActivity)
36            mainSwipeRefresh.setOnRefreshListener(this@MainActivity)
37            mainDrawerNavigation.setNavigationItemSelectedListener(this@MainActivity)
38        }
39
40        setupDrawer()
41    }
42
43    private fun setupDrawer() {
44        drawerLayout = binding.mainDrawerLayout
45        actionBarToggle = ActionBarDrawerToggle(this, drawerLayout, 0,0)
46        drawerLayout.addDrawerListener(actionBarToggle)
47        actionBarToggle.syncState()
48    }
49
50    override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
51        if (currentFocus != null) {
52            val imm: InputMethodManager =
53                getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
54            imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
55        }
56        return super.dispatchTouchEvent(ev)
57    }
58
59    override fun onClick(v: View?) {
60        when(v){
61            binding.mainAddFragment -> MainAddFragment().show(supportFragmentManager,
62                TAG+"FAB")
63        }
64    }
65
66    override fun onRefresh() {
67        when(binding.mainBottomNavigation.selectedItemId){
68            R.id.mainHomeFragment -> viewModel.refreshFarm()
69            R.id.mainSocialFragment -> viewModel.refreshPosts()
70            R.id.mainNotesFragment -> viewModel.refreshNotes()
71            R.id.mainProfileFragment -> {}
72        }
73    }

```

```

71     }
72
73     override fun onNavigationItemSelected(item: MenuItem): Boolean {
74         return when(item.itemId) {
75             R.id.drawer_edit -> {
76                 val intent = Intent(this, EditProfileUserActivity::class.java)
77                 intent.putExtra(BuildConfig.CURRENT_USER,
78                     viewModel.getCurrentUser().value)
79                 Log.i(TAG, "onNavigationItemSelected:
80                     ${viewModel.getCurrentUser().value}")
81                 startActivity(intent)
82                 drawerLayout.closeDrawer(GravityCompat.END)
83                 true
84             }
85             R.id.drawer_signout -> {
86                 AlertDialog.Builder(this)
87                     .setTitle(getString(R.string.sign_out))
88                     .setMessage(getString(R.string.sign_out_warning))
89                     .setPositiveButton(getString(R.string.sign_out)){ _,_ ->
90                         val uid = Firebase.auth.uid!!
91                         viewModel.signOut()
92                         viewModel.isUserSignOut.observe(this@MainActivity, {
93                             it?.let {
94                                 if(it){
95                                     drawerLayout.closeDrawer(GravityCompat.END)
96                                     DatabaseHandler(this@MainActivity).signOut(uid)
97                                     startActivity(Intent(this@MainActivity,
98                                         SignInActivity::class.java))
99                                     finish()
100                                } else {
101                                    dialogButton.forEach { it.isEnabled = true }
102                                    viewModel.signOutError.observe(this@MainActivity, {
103                                        it?.let {
104                                            if (it.isNotEmpty()) {
105                                                Toast.makeText(this@MainActivity, it,
106                                                    Toast.LENGTH_SHORT).show()
107                                                Log.i(TAG, it)
108                                                viewModel.signOutError.value = null
109                                            }
110                                        }
111                                    })
112                                }
113                            }.setNegativeButton(getString(R.string.cancel)){_,_ ->}
114                            .create()
115                            .show()
116                            true
117                        }
118                    R.id.drawer_feedback -> {
119                        val intent = Intent(this, FeedbackActivity::class.java)
120                        intent.putExtra(BuildConfig.CURRENT_USER,
121                            viewModel.getCurrentUser().value)
122                        startActivity(intent)
123                        drawerLayout.closeDrawer(GravityCompat.END)
124                        true
125                    }
126                    R.id.drawer_info -> {
127                        IntentUtility(this).openAppInfo()
128                        drawerLayout.closeDrawer(GravityCompat.END)
129                        true
130                    }
131                    R.id.drawer_language ->{
132                        IntentUtility(this).openLanguageSettings()
133                        drawerLayout.closeDrawer(GravityCompat.END)
134                        true
135                    }
136                    R.id.drawer_about -> {

```

```
136         startActivity(Intent(this, AboutMeActivity::class.java))
137         drawerLayout.closeDrawer(GravityCompat.END)
138         true
139     }
140     R.id.drawer_my_post -> {
141         val intent = Intent(this, ShowRecyclerActivity::class.java)
142         intent.putExtra(BuildConfig.CURRENT_USER,
143             viewModel.getCurrentUser().value)
144         intent.putExtra(BuildConfig.ALL_PLANTS, viewModel.getAllPosts().value)
145         startActivity(intent)
146         true
147     }
148     else -> false
149   }
150 }
```

Kode 35: *Source Code* untuk *Main Activity* beserta *fragments* terkait

a) MAIN HOME FRAGMENT

```
1  class MainHomeFragment : Fragment(), View.OnClickListener, CardStackListener{
2      private var _binding : FragmentMainHomeBinding? = null
3      private val binding get() = _binding!!
4      private lateinit var viewModel : MainViewModel
5      private lateinit var viewsAsButton : ArrayList<View>
6      private lateinit var cardStackLayoutManager: CardStackLayoutManager
7      private lateinit var cardStackAdapter : RecyclerView.Adapter<*>
8
9      companion object {
10          const val TAG = "mainHomeFragment"
11      }
12
13      override fun onCreateView(
14          inflater: LayoutInflater, container: ViewGroup?,
15          savedInstanceState: Bundle?
16      ): View {
17          _binding = FragmentMainHomeBinding.inflate(inflater, container, false)
18          return binding.root
19      }
20
21      override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
22          super.onViewCreated(view, savedInstanceState)
23          setHasOptionsMenu(true)
24          requireActivity().title = getString(R.string.home)
25          viewModel = ViewModelProvider(requireActivity()).get(MainViewModel::class.java)
26
27          binding.apply {
28              viewsAsButton = arrayListOf(mainCardPrevious, mainCardNext)
29              viewsAsButton.forEach { it.setOnClickListener(this@MainHomeFragment) }
30
31              viewModel.getCurrentFarm().observe(viewLifecycleOwner, {
32                  it?.let {
33                      mainHomeFarmName.text = it.name.toString()
34                      Log.i(TAG, "onViewCreated: update")
35                  }
36              })
37          }
38
39          setupCardStack()
40      }
41
42      private fun setupCardStack() {
43          val data : ArrayList<KitModel> = arrayListOf()
44          cardStackAdapter = AdapterType.DATA_MONITORING.getAdapter(requireContext(),
45          → data)
46          viewModel.getCurrentKits().observe(viewLifecycleOwner, {
47              val temp = it ?: arrayListOf()
48              data.clear()
49          })
50      }
51
52      private fun setupCardStack() {
53          val data : ArrayList<KitModel> = arrayListOf()
54          cardStackAdapter = AdapterType.DATA_MONITORING.getAdapter(requireContext(),
55          → data)
56          viewModel.getCurrentKits().observe(viewLifecycleOwner, {
57              val temp = it ?: arrayListOf()
58              data.clear()
59          })
60      }
61
62      private fun setupCardStack() {
63          val data : ArrayList<KitModel> = arrayListOf()
64          cardStackAdapter = AdapterType.DATA_MONITORING.getAdapter(requireContext(),
65          → data)
66          viewModel.getCurrentKits().observe(viewLifecycleOwner, {
67              val temp = it ?: arrayListOf()
68              data.clear()
69          })
70      }
71
72      private fun setupCardStack() {
73          val data : ArrayList<KitModel> = arrayListOf()
74          cardStackAdapter = AdapterType.DATA_MONITORING.getAdapter(requireContext(),
75          → data)
76          viewModel.getCurrentKits().observe(viewLifecycleOwner, {
77              val temp = it ?: arrayListOf()
78              data.clear()
79          })
80      }
81
82      private fun setupCardStack() {
83          val data : ArrayList<KitModel> = arrayListOf()
84          cardStackAdapter = AdapterType.DATA_MONITORING.getAdapter(requireContext(),
85          → data)
86          viewModel.getCurrentKits().observe(viewLifecycleOwner, {
87              val temp = it ?: arrayListOf()
88              data.clear()
89          })
90      }
91
92      private fun setupCardStack() {
93          val data : ArrayList<KitModel> = arrayListOf()
94          cardStackAdapter = AdapterType.DATA_MONITORING.getAdapter(requireContext(),
95          → data)
96          viewModel.getCurrentKits().observe(viewLifecycleOwner, {
97              val temp = it ?: arrayListOf()
98              data.clear()
99          })
100     }
101 }
```

```

48     data.addAll(temp)
49     cardStackAdapter.notifyDataSetChanged()
50     (cardStackAdapter as KitModelAdapter).setOnItemClickListener(
51         object : KitModelAdapter.OnItemClickListener{
52             override fun onItemClick(position: Int) {
53                 val intent = Intent(requireContext(),
54                     ProfileKitActivity::class.java)
55                 intent.putExtra(BuildConfig.CURRENT_USER,
56                     viewModel.getCurrentUser().value)
57                 intent.putExtra(BuildConfig.CURRENT_FARM,
58                     viewModel.getCurrentFarm().value)
59                 intent.putExtra(BuildConfig.SELECTED_KIT,
60                     viewModel.getKit(position))
61                 startActivity(intent)
62             }
63         })
64     Log.i(TAG, "adapter count ${cardStackAdapter.itemCount}")
65     (requireActivity() as MainActivity).swipeRefresh.isRefreshing = false
66
67     if(temp.size ==0){
68         binding.mainCardPrevious.visibility = View.GONE
69         binding.mainCardNext.visibility = View.GONE
70     } else {
71         binding.mainCardPrevious.visibility = View.VISIBLE
72         binding.mainCardNext.visibility = View.VISIBLE
73     }
74 }
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110

```

```

111         startActivity(intent)
112     }
113 }
114 return super.onOptionsItemSelected(item)
115 }

116 override fun onClick(v: View?) {
117     when(v){
118         binding.mainCardNext -> {
119             if(cardStackLayoutManager.topPosition < cardStackAdapter.itemCount-1){
120                 binding.mainHomeCardStack.swipe()
121             } else {
122                 binding.mainHomeCardStack.smoothScrollToPosition(0)
123             }
124         }
125         binding.mainCardPrevious -> {
126             if(cardStackLayoutManager.topPosition != 0){
127                 binding.mainHomeCardStack.rewind()
128             } else {
129                 binding.mainHomeCardStack.smoothScrollToPosition(cardStackAdapter.i
130                     ↵ temCount-1)
131             }
132         }
133     }
134 }
135

136 override fun onCardDragging(direction: Direction?, ratio: Float) {}

137 override fun onCardSwiped(direction: Direction?) {
138     if(cardStackLayoutManager.topPosition == cardStackAdapter.itemCount){
139         binding.mainHomeCardStack.scrollToPosition(0)
140     }
141 }
142

143 override fun onCardRewound() {}

144 override fun onCardCanceled() {}

145

146 override fun onCardAppeared(view: View?, position: Int) {
147     Log.i(TAG, "appear $position")
148 }
149

150 override fun onCardDisappeared(view: View?, position: Int) {}
151
152
153
154 }
```

Kode 36: *Source Code* untuk *Main Home Fragment*

b) MAIN POSTS FRAGMENT

```

1 class MainPostsFragment : Fragment() {
2     private var _binding : FragmentMainPostsBinding? = null
3     private val binding get() = _binding!!
4     private lateinit var viewModel : MainViewModel
5
6     companion object {
7         const val TAG = "mainSocialFragment"
8     }
9
10    override fun onCreateView(
11        inflater: LayoutInflater, container: ViewGroup?,
12        savedInstanceState: Bundle?
13    ): View {
14        _binding = FragmentMainPostsBinding.inflate(inflater, container, false)
15        return binding.root
16    }
17
18    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
```

```

19         super.onViewCreated(view, savedInstanceState)
20         requireActivity().title = getString(R.string.social)
21         viewModel = ViewModelProvider(requireActivity()).get(MainViewModel::class.java)
22         setupRecyclerView()
23     }
24
25     private fun setupRecyclerView() {
26         val plantModels : ArrayList<PlantModel> = arrayListOf()
27         val userModels : ArrayList<UserModel> = arrayListOf()
28         val rowAdapter = AdapterType.POST_PLANT.getAdapter(requireContext(),
29             ↪ plantModels, allUsers = userModels)
30         viewModel.getAllPosts().observe(viewLifecycleOwner, {
31             plantModels.clear()
32             plantModels.addAll(it ?: arrayListOf())
33             rowAdapter.notifyDataSetChanged()
34             (rowAdapter as PostModelAdapter).setOnItemClickListener(
35                 object : PostModelAdapter.OnItemClickListener{
36                     override fun onItemClick(position: Int,
37                         itemView: View,
38                         v: RowItemPostBinding
39                     ) {
40                         when(itemView){
41                             v.postImageContent -> {
42                                 IntentUtility(requireContext()).openImage(itemView,
43                                     ↪ viewModel.getPostUser(position)?.name ?: "Photo"
44                                     ↪ "Profile" )
45                             }
46                             v.postRoot -> {
47                                 val intent = Intent(requireContext(),
48                                     ↪ ProfileUserActivity::class.java)
49                                 intent.putExtra(BuildConfig.SELECTED_USER,
50                                     ↪ viewModel.getPostUser(position))
51                                 startActivity(intent)
52                             }
53                             v.postOptions -> {
54                                 val items = arrayOf("Edit", "Delete")
55                                 MaterialAlertDialogBuilder(context!!)
56                                 .setItems(items){_, which ->
57                                     when(which){
58                                         0 -> gotoPost(position)
59                                         1 -> {
60                                             viewModel.deletePost(position)
61                                             viewModel.isPostDeleted.observe(requireActivity(),
62                                                 { it ->
63                                     when(it){
64                                         true -> Toast.makeText(requireActivity(),
65                                             "Post deleted.", "Post deleted.",
66                                             Toast.LENGTH_SHORT).show()
67                                         }
68                                     })
69                                         }
70                                         }
71                                     Log.i(TAG, "${rowAdapter.itemCount}")
72                                 })
73
74         viewModel.getAllUsers().observe(viewLifecycleOwner, {
75             userModels.clear()
76             userModels.addAll(it ?: arrayListOf())
77             rowAdapter.notifyDataSetChanged()
78         })
79

```

```

80         binding.mainSocialsRecyclerView.apply {
81             adapter = rowAdapter
82             layoutManager = LinearLayoutManager(requireContext())
83             addItemDecoration(object : DividerItemDecoration(requireContext(),
84                 VERTICAL) {})
85             setHasFixedSize(true)
86         }
87     }
88
89     private fun gotoPost(position:Int){
90         val intent = Intent(requireContext(), AddPlantActivity::class.java)
91         intent.putExtra(BuildConfig.SELECTED_PLANT, viewModel.getPost(position))
92         startActivity(intent)
93     }
94 }
```

Kode 37: *Source Code* untuk *Main Posts Fragment*

c) MAIN ADD FRAGMENT

```

1  class MainAddFragment : BottomSheetDialogFragment(),
2      NavigationView.OnNavigationItemSelectedListener {
3      private var _binding : FragmentMainAddBinding? = null
4      private val binding get() = _binding!!
5      private lateinit var viewModel : MainViewModel
6
7      override fun onCreateView(
8          inflater: LayoutInflater, container: ViewGroup?,
9          savedInstanceState: Bundle?
10     ): View {
11         _binding = FragmentMainAddBinding.inflate(inflater, container, false)
12         return binding.root
13     }
14
15     override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
16         super.onViewCreated(view, savedInstanceState)
17         viewModel = ViewModelProvider(requireActivity()).get(MainViewModel::class.java)
18         binding.mainAddMenu.itemIconTintList = null
19         binding.mainAddMenu.setNavigationItemSelectedListener(this)
20     }
21
22     override fun onNavigationItemSelected(item: MenuItem): Boolean {
23         return when(item.itemId){
24             R.id.addDataMonitoring    ->
25                 gotoActivity(AddDataMonitoringActivity::class.java)
26             R.id.addCrops              -> gotoActivity(AddCropsActivity::class.java)
27             R.id.addNote               -> gotoActivity(AddNoteActivity::class.java)
28             R.id.addKit                -> gotoActivity(AddKitActivity::class.java)
29             R.id.addPlant              -> gotoActivity(AddPlantActivity::class.java)
30             else                      -> false
31         }
32     }
33
34     private fun gotoActivity(destination : Class<*>) : Boolean{
35         dismiss()
36         val intent = Intent(requireActivity(), destination)
37         intent.putExtra(BuildConfig.CURRENT_USER, viewModel.getCurrentUser().value)
38         intent.putExtra(BuildConfig.CURRENT_FARM, viewModel.getCurrentFarm().value)
39         startActivity(intent)
40         return true
41     }
42 }
```

Kode 38: *Source Code* untuk *Main Add Fragment*

d) MAIN NOTES FRAGMENT

```
1  class MainNotesFragment : Fragment(), SwipeRefreshLayout.OnRefreshListener {
2      private var _binding : FragmentMainNotesBinding? = null
3      private val binding get() = _binding!!
4      private lateinit var viewModel : MainViewModel
5
6      companion object {
7          const val TAG = "mainNotesFragment"
8      }
9
10     override fun onCreateView(
11         inflater: LayoutInflater, container: ViewGroup?,
12         savedInstanceState: Bundle?
13     ): View {
14         _binding = FragmentMainNotesBinding.inflate(inflater, container, false)
15         return binding.root
16     }
17
18     override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
19         super.onViewCreated(view, savedInstanceState)
20         requireActivity().title = getString(R.string.notes)
21         viewModel = ViewModelProvider(requireActivity()).get(MainViewModel::class.java)
22         setupRecyclerView()
23     }
24
25     private fun setupRecyclerView() {
26         val data : ArrayList<NoteModel> = arrayListOf()
27         val rowAdapter = AdapterType.NOTE.getAdapter(requireContext(), data)
28         viewModel.getCurrentNotes().observe(viewLifecycleOwner, {
29             val currentData : ArrayList<NoteModel> = it ?: arrayListOf()
30             data.clear()
31             data.addAll(currentData)
32             rowAdapter.notifyDataSetChanged()
33             (rowAdapter as NoteModelAdapter).setOnItemClickListener(
34                 object : NoteModelAdapter.OnItemClickListener{
35                     override fun onItemClick(
36                         position: Int,
37                         itemView: View,
38                         v: RowItemNoteBinding
39                     ) {
40                         gotoNote(position)
41                     }
42
43                     override fun onItemLongClicked(position: Int) {
44                         val items = arrayOf("Edit", "Delete")
45                         MaterialAlertDialogBuilder(context!!)
46                             .setItems(items){_, which ->
47                                 when(which){
48                                     0 -> gotoNote(position)
49                                     1 -> {
50                                         viewModel.deleteNote(position)
51                                         viewModel.isNoteDeleted.observe(requireActivity())
52                                             .asFlow()
53                                             .collect {
54                                                 when(it){
55                                                     true ->
56                                                     Toast.makeText(requireContext(),
57                                                     "Note deleted.",
58                                                     Toast.LENGTH_SHORT).show()
59                                                 }
60                                             }
61                                         }
62                                     }
63                                 Log.i(TAG, "${rowAdapter.itemCount}")
64                             }
65                         .show()
66                     }
67                 })
68             }
69         )
70     }
71 }
```

```
64     (requireActivity() as MainActivity).swipeRefresh.isRefreshing = false
65     binding.noteLine.visibility = if(currentData.size==0){
66         View.GONE
67     } else {
68         View.VISIBLE
69     }
70 }
71
72 binding.mainNotesRecyclerView.apply {
73     adapter = rowAdapter
74     layoutManager = LinearLayoutManager(requireContext())
75 }
76
77
78 override fun onRefresh() {
79     viewModel.refreshNotes()
80 }
81
82 private fun gotoNote(position:Int){
83     val intent = Intent(requireContext(), AddNoteActivity::class.java)
84     intent.putExtra(BuildConfig.SELECTED_NOTE, viewModel.getNote(position))
85     startActivity(intent)
86 }
87 }
```

Kode 39: *Source Code untuk Main Notes Fragment*

e) MAIN PROFILE FRAGMENT

```

41             getString(R.string.joined_company_since, it.joinedSince)
42             mainProfileBio.text = it.bio
43             it.performanceRate?.let {
44                 mainProfileRate.rating = it
45             }
46             mainProfileEmail.text = it.email
47             mainProfileRole.text = getString(
48                 R.string.role_in_profile,
49                 it.role!!.replaceFirstChar { it.uppercase() },
50                 viewModel.getCurrentFarm()?.value?.name.toString()
51             )
52             Glide.with(requireActivity())
53                 .load(it.photo_url ?: R.drawable.bg_farmer)
54                 .centerCrop()
55                 .into(mainProfilePhoto)
56             }
57         } catch (e:Exception){
58             Log.e(TAG, "currentUser value null in some field", e)
59         }
60     })
61 }
62 }
63
64 override fun onCreateOptionsMenu(menu: Menu, inflater: MenuInflater) {
65     super.onCreateOptionsMenu(menu, inflater)
66     menu.clear()
67     inflater.inflate(R.menu.menu_main_profile, menu)
68 }
69
70 override fun onOptionsItemSelected(item: MenuItem): Boolean {
71     when(item.itemId){
72         R.id.settings -> {
73             if(drawerLayout.isDrawerOpen(GravityCompat.END)){
74                 drawerLayout.closeDrawer(GravityCompat.END)
75             }
76             else{
77                 drawerLayout.openDrawer(GravityCompat.END)
78             }
79         }
80     }
81     return super.onOptionsItemSelected(item)
82 }
83
84 override fun onClick(v: View?) {
85     when(v){
86         binding.mainProfileEmail -> IntentUtility(requireContext()).openEmail(binding.mainProfileEmail.text.toString())
87         binding.mainProfilePhoneGroup ->
88             RequestPermission().requestPermission(requireActivity(),
89             Manifest.permission.CALL_PHONE, "Phone call",
90             binding.mainProfilePhone.text.toString())
91         binding.mainProfileAddressGroup -> IntentUtility(requireContext()).openMaps((binding.mainProfileAddress.text.toString()))
92         binding.mainProfilePhoto ->
93             IntentUtility(requireContext()).openImage(binding.mainProfilePhoto, binding.mainProfileName.text.toString())
94     }
95 }
96 }

```

Kode 40: Source Code untuk Main Profile Fragment

7. ADD CROPS ACTIVITY

```

1 class AddCropsActivity : AppCompatActivity(), View.OnClickListener {
2     private lateinit var binding : ActivityAddCropsBinding
3     private lateinit var viewModel : AddDataMonitoringViewModel
4     private lateinit var utility: ViewUtility

```

```

5   private lateinit var viewsAsButton : ArrayList<View>
6
7   companion object{
8     const val TAG = "addCrops"
9   }
10
11  override fun onCreate(savedInstanceState: Bundle?) {
12    super.onCreate(savedInstanceState)
13    binding = ActivityAddCropsBinding.inflate(layoutInflater)
14    setContentView(binding.root)
15
16    viewModel = ViewModelProvider(this).get(AddDataMonitoringViewModel::class.java)
17    viewModel.setCurrentData(intent.getParcelableExtra<UserModel>(BuildConfig.CURRENT_USER),
18      intent.getStringExtra(FARM_MODEL))
19    isAddCrops = true
20    supportActionBar?.title = getString(R.string.add_new_crops)
21    supportActionBar?.subtitle = viewModel.currentFarmModel.value?.name
22    supportActionBar?.setDisplayHomeAsUpEnabled(true)
23    supportActionBar?.setDisplayShowHomeEnabled(false)
24
25    binding.apply {
26      viewsAsButton = arrayListOf(acSelectPlant, acNewPlant, acCropsSubmit)
27      viewsAsButton.forEach { it.setOnClickListener(this@AddCropsActivity) }
28
29      utility = ViewUtility(
30        context = this@AddCropsActivity,
31        circularProgressBar = acCropsSubmit,
32        viewsAsButton = viewsAsButton,
33        actionBar = supportActionBar
34    )
35
36    viewModel.getKitSelector().observe(this@AddCropsActivity, {
37      it?.let{
38        val adapter = ArrayAdapter(this@AddCropsActivity,
39          R.layout.row_item_list, it)
40        (acSelectKit.editText as? AutoCompleteTextView)?.setAdapter(adapter)
41        Log.i(TAG, "${adapter.count} ${it}")
42
43        (acSelectKit.editText as?
44          AutoCompleteTextView)?.setOnItemClickListener { parent, view,
45          position, id ->
46          viewModel.setCurrentKit(position)
47          checkEmpty()
48        }
49      }
50    })
51
52    viewModel.getCurrentPlant().observe(this@AddCropsActivity, {
53      if(it != null){
54        try{
55          val estimated : String =
56            utility.formatDateToString(utility.formatStringToDate(est =
57              it.growthTime!!)).toString()
58          acPlantName.setText(it.name)
59          acPlantDescription.setText(it.description)
60          acPlantGrowth.setText(getString(R.string.d_days, it.growthTime))
61          acPlantEstimated.setText(estimated)
62          acPlantTempMin.setText(getString(R.string.temp_value,
63            it.tempLv?.min))
64          acPlantTempMax.setText(getString(R.string.temp_value,
65            it.tempLv?.max))
66          acPlantHumidMin.setText(getString(R.string.humid_value,
67            it.humidLv?.min))
68          acPlantHumidMax.setText(getString(R.string.humid_value,
69            it.humidLv?.max))
70          acPlantAcidMin.setText(it.phLv?.min.toString())
71          acPlantAcidMax.setText(it.phLv?.max.toString())
72          Glide.with(this@AddCropsActivity)
73            .load(it.photo_url)
74        }
75      }
76    })
77  }
78
79  
```

```
65             .circleCrop()
66             .into(acPlantPhoto)
67         } catch (e:Exception){
68             Log.e(AddDataMonitoringActivity.TAG, "cannot parse plant data",
69                  e)
70         }
71     } else {
72         acPlantSelector.visibility = View.VISIBLE
73         acCropsContent.visibility = View.GONE
74     }
75     checkEmpty()
76 }
77 checkEmpty()
78 }
79
80 private fun checkEmpty() {
81     viewModel.checkNotEmpty(
82         viewModel.getCurrentKit().value != null &&
83         viewModel.getCurrentPlant().value != null
84     ).observe(this, {
85         binding.acCropsSubmit.isEnabled = it
86     })
87 }
88
89 override fun onSupportNavigateUp(): Boolean {
90     super.onBackPressed()
91     finish()
92     return true
93 }
94
95 override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
96     if (currentFocus != null) {
97         val imm: InputMethodManager =
98             getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
99             imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
100    }
101    return super.dispatchTouchEvent(ev)
102 }
103
104 override fun onClick(v: View?) {
105     when(v){
106         binding.acNewPlant -> {
107             val intent = Intent(this, AddPlantActivity::class.java)
108             intent.putExtra("from", TAG)
109             startForResult.launch(intent)
110         }
111         binding.acSelectPlant -> {
112             val intent = Intent(this, SearchActivity::class.java)
113             intent.putExtra("search", TAG)
114             startForResult.launch(intent)
115         }
116         binding.acCropsSubmit -> {
117             utility.isLoading = true
118             viewModel.addCrops()
119             viewModel.isCropsAdd.observe(this, {
120                 if(it){
121                     utility.isLoading = false
122                     Toast.makeText(this, "New crops added",
123                         Toast.LENGTH_SHORT).show()
124                     super.onBackPressed()
125                     finish()
126                 } else {
127                     utility.isLoading = false
128                     viewModel.addCropsError.observe(this, {
129                         if(it.isNotEmpty()){
130                             Toast.makeText(this, it, Toast.LENGTH_SHORT).show()
131                             Log.i(CreateFarmFragment.TAG, it)
132                             viewModel.addCropsError.value = ""
133                         }
134                     })
135                 }
136             })
137         }
138     }
139 }
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2190
2191
2192
2193
2194
2195
2195
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2
```

```

133             })
134         }
135     }
136 }
137 }
138 }
139
140 private val startForResult =
141     registerForActivityResult(ActivityResultContracts.StartActivityForResult()) {
142     data ->
143         Log.i(CreateProfileActivity.TAG, "$data")
144         try{
145             if(data?.resultCode == Activity.RESULT_OK){
146                 data.data?.getParcelableExtra<PlantModel>(BuildConfig.SELECTED_PLANT)?.
147                     let {
148                         binding.acPlantSelector.visibility = View.GONE
149                         binding.acCropsContent.visibility = View.VISIBLE
150                         viewModel.setCurrentPlant(it)
151                     }
152             }
153         } catch (e: Exception){
154             Toast.makeText(this, e.message.toString(), Toast.LENGTH_SHORT).show()
155         }
156     }

```

Kode 41: *Source Code* untuk *Add Crops Activity*

8. ADD DATA MONITORING ACTIVITY

```

1 class AddDataMonitoringActivity : AppCompatActivity(), View.OnClickListener {
2     private lateinit var binding : ActivityAddDataMonitoringBinding
3     private lateinit var bindingFragment : ActivityAddCropsBinding
4     private lateinit var viewModel : AddDataMonitoringViewModel
5     private lateinit var utility: ViewUtility
6     private lateinit var tooltips : ArrayList<ImageButton>
7     private lateinit var viewsAsButton : ArrayList<View>
8     private lateinit var numberPickers : ArrayList<ClickNumberPickerView>
9
10    companion object{
11        const val TAG = "addDataMonitoring"
12    }
13
14    //todo : kurang tombol buat panen, nampilin yg msh ada tanemannya
15    override fun onCreate(savedInstanceState: Bundle?) {
16        super.onCreate(savedInstanceState)
17        binding = ActivityAddDataMonitoringBinding.inflate(layoutInflater)
18        setContentView(binding.root)
19
20        viewModel = ViewModelProvider(this).get(AddDataMonitoringViewModel::class.java)
21        supportActionBar?.title = getString(R.string.add_recent_data_monitoring)
22        viewModel.setCurrentData(intent.getParcelableExtra<UserModel>(BuildConfig.CURREN
23            NT_USER),
24            intent.getParcelableExtra<FarmModel>(BuildConfig.CURRENT_FARM),
25            isAddCrops = false)
26        supportActionBar?.subtitle = viewModel.currentFarmModel.value?.name
27        supportActionBar?.setDisplayHomeAsUpEnabled(true)
28        supportActionBar?.setDisplayShowHomeEnabled(false)
29
30        //todo : tooltip semuanya
31        binding.apply {
32            bindingFragment = binding.addMonitoringCropsCreate
33            bindingFragment.apply {
34                acSelectKit.visibility = View.GONE
35                acCropsSubmit.visibility = View.GONE

```

```

36     viewsAsButton = arrayListOf(
37         addMonitoringSubmit,
38         addMonitoringCropsButton,
39         acNewPlant,
40         acSelectPlant
41     )
42     numberPickers = arrayListOf(
43         addMonitoringTemperature,
44         addMonitoringHumidity,
45         addMonitoringAcidity,
46         addMonitoringWaterTank,
47         addMonitoringNutrientTank,
48         addMonitoringWaterTurbidity
49     )
50     tooltips = arrayListOf(admTempInfo,
51         admHumidInfo,
52         admAcidityInfo,
53         admWaterLevelInfo,
54         admNutrientLevelInfo,
55         admWaterTurbidityInfo)
56
57     utility = ViewUtility(
58         context = this@AddDataMonitoringActivity,
59         circularProgressBar = addMonitoringSubmit,
60         viewsAsButton = viewsAsButton,
61         actionBar = supportActionBar
62     )
63
64     numberPickers.forEachIndexed { i, view ->
65         val numberPickerDataType = getNumberPickerType(view)
66         viewModel.getNumberPickerValue(numberPickerDataType)?.observe(this@]
67             &-> AddDataMonitoringActivity, {
68                 if(it != view.value){
69                     view.setPickerValue(it)
70                 }
71             })
72
73         numberPickers[i].setOnClickListener { previousValue,
74             &-> currentValue, pickerClickType ->
75                 viewModel.setNumberPickerValue(currentValue,
76                     &-> numberPickerDataType)
77                 checkEmpty()
78             }
79         viewsAsButton.forEach {
80             &-> it.setOnClickListener(this@AddDataMonitoringActivity) }
81         tooltips.forEach {
82             it.setOnClickListener {
83                 it.performLongClick()
84             }
85         }
86         admHumidInfo.tooltipText = "Min : \n" +
87             "Max : "
88
89         viewModel.getCurrentCrops().observe(this@AddDataMonitoringActivity, {
90             if(it == null){ //crops not found
91                 addMonitoringCropsPlanted.root.visibility = View.GONE
92                 addMonitoringNewCrops.visibility = View.VISIBLE
93             } else{ //crops found
94                 if(it.cropsId != null) {
95                     addMonitoringCropsPlanted.root.visibility = View.VISIBLE
96                     addMonitoringNewCrops.visibility = View.GONE
97                     addMonitoringCropsPlanted.apply {
98                         //todo kurang moreday
99                         val plantedDate : String =
100                             utility.formatTimestampToDate(it.timestamp)
101                         val estimateDate : Long =
102                             utility.formatStringToDate(plantedDate, it.plantModel?.growthTime)
103                             this.kitPlantName.text = it.plantModel?.name.toString()
104                             this.kitPlantDate.text = plantedDate
105                         }
106                     }
107                 }
108             }
109         )
110     }

```

```

100         this.kitPlantHarvest.text = getString(
101             R.string.can_be_harvested_at_s,
102             "now"
103         )//utility.formatDateToString(estimateDate)
104         Glide.with(this@AddDataMonitoringActivity)
105             .load(it.plantModel?.photo_url)
106             .circleCrop()
107             .into(this.kitPlantPhoto)
108     }
109   }
110 }
111
112 viewModel.getCurrentPlant().observe(this@AddDataMonitoringActivity, {
113     if(it != null){
114         try{
115             acPlantSelector.visibility = View.GONE
116             acCropsContent.visibility = View.VISIBLE
117             addMonitoringCropsButton.visibility = View.GONE
118
119             val estimated : String = utility.formatDateToString(utility]
120             ↪ .formatStringToDate(est =
121             ↪ it.growthTime!!)).toString()
122             acPlantName.setText(it.name)
123             acPlantDescription.setText(it.description)
124             acPlantGrowth.setText(getString(R.string.d_days,
125             ↪ it.growthTime))
126             acPlantEstimated.setText(estimated)
127             acPlantTempMin.setText(getString(R.string.temp_value,
128             ↪ it.tempLv?.min))
129             acPlantTempMax.setText(getString(R.string.temp_value,
130             ↪ it.tempLv?.max))
131             acPlantHumidMin.setText(getString(R.string.humid_value,
132             ↪ it.humidLv?.min))
133             acPlantHumidMax.setText(getString(R.string.humid_value,
134             ↪ it.humidLv?.max))
135             acPlantAcidMin.setText(it.phLv?.min.toString())
136             acPlantAcidMax.setText(it.phLv?.max.toString())
137             Glide.with(this@AddDataMonitoringActivity)
138                 .load(it.photo_url)
139                 .circleCrop()
140                 .into(acPlantPhoto)
141         } catch (e:Exception){
142             Log.e(TAG, "cannot parse plant data", e)
143         }
144     } else {
145         acPlantSelector.visibility = View.VISIBLE
146         acCropsContent.visibility = View.GONE
147         addMonitoringCropsButton.visibility = View.VISIBLE
148     }
149     checkEmpty()
150 }
151
152 viewModel.getKitSelector().observe(this@AddDataMonitoringActivity, {
153     it?.let{
154         val adapter = ArrayAdapter(this@AddDataMonitoringActivity,
155             ↪ R.layout.row_item_list, it)
156         (admSelectKit.editText as?
157             ↪ AutoCompleteTextView)?.setAdapter(adapter)
158         Log.i(TAG, "${adapter.count} ${it}")
159
160         (admSelectKit.editText as?
161             ↪ AutoCompleteTextView)?.setOnItemClickListener { parent,
162             ↪ view, position, id ->
163                 viewModel.setCurrentKit(position)
164                 checkEmpty()
165             }
166     }
167 }
168
169 checkEmpty()

```

```

159         }
160     }
161 }
162
163     private fun checkEmpty() {
164         val currTemp : Float = viewModel.getNumberPickerValue(getNumberPickerType(binding))
165             .value ?: 0f
166         val currHumid : Float = viewModel.getNumberPickerValue(getNumberPickerType(binding))
167             .value ?: 0f
168         val currAcid : Float = viewModel.getNumberPickerValue(getNumberPickerType(binding))
169             .value ?: 0f
170         val currWaterTank : Float = viewModel.getNumberPickerValue(getNumberPickerType(binding))
171             .value ?: 0f
172         val currNutrientTank : Float = viewModel.getNumberPickerValue(getNumberPickerType(binding))
173             .value ?: 0f
174         val currentTurbidity : Float = viewModel.getNumberPickerValue(getNumberPickerType(binding))
175             .value ?: 0f
176
177         viewModel.checkNotEmpty(
178             viewModel.getCurrentKit().value != null &&
179                 currTemp > 0f &&
180                 currHumid > 0f &&
181                 currAcid > 0f &&
182                 currWaterTank > 0f &&
183                 currNutrientTank > 0f &&
184                 currentTurbidity > 0f
185         ).observe(this, {
186             binding.addMonitoringSubmit.isEnabled = it
187         })
188     }
189
190     override fun onSupportNavigateUp(): Boolean {
191         super.onBackPressed()
192         finish()
193         return true
194     }
195
196     override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
197         if (currentFocus != null) {
198             val imm: InputMethodManager =
199                 getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
200             imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
201         }
202         return super.dispatchTouchEvent(ev)
203     }
204
205     override fun onClick(v: View?) {
206         when(v){
207             binding.addMonitoringSubmit -> {
208                 utility.isLoading = true
209                 when(binding.addCropsLayout.visibility){
210                     View.GONE -> viewModel.addDataMonitoring()
211                     View.VISIBLE -> viewModel.addCrops()
212                 }
213                 viewModel.isDataMonitoringAdd.observe(this, {
214                     if(it){
215                         utility.isLoading = false
216                         Toast.makeText(this, "New data monitoring added",
217                             Toast.LENGTH_SHORT).show()
218                         super.onBackPressed()
219                         finish()
220                     } else {
221                         utility.isLoading = false
222                         viewModel.addDataMonitoringError.observe(this, {

```

```

216         if(it.isNotEmpty()){
217             Toast.makeText(this, it, Toast.LENGTH_SHORT).show()
218             Log.i(CreateFarmFragment.TAG, it)
219             viewModel.addDataMonitoringError.value = ""
220         }
221     })
222   })
223 }
224 binding.addMonitoringCropsButton ->{
225   when(binding.addCropsLayout.visibility){
226     View.GONE -> {
227       binding.addCropsLayout.visibility = View.VISIBLE
228       binding.addMonitoringCropsButton.text =
229           getString(R.string.cancel)
230     }
231     View.VISIBLE -> {
232       binding.addCropsLayout.visibility = View.GONE
233       binding.addMonitoringCropsButton.text =
234           getString(R.string.add_new_crops)
235     }
236   }
237   bindingFragment.acNewPlant -> {
238     val intent = Intent(this, AddPlantActivity::class.java)
239     intent.putExtra("from", TAG)
240     startForResult.launch(intent)
241   }
242   bindingFragment.acSelectPlant -> {
243     val intent = Intent(this, SearchActivity::class.java)
244     intent.putExtra("search", TAG)
245     startForResult.launch(intent)
246   }
247 }
248 }
249
250 private val startForResult =
251   registerForActivityResult(ActivityResultContracts.StartActivityForResult()) {
252   data ->
253   Log.i(CreateProfileActivity.TAG, "$data")
254   try{
255     if(data?.resultCode == Activity.RESULT_OK){
256       data.data?.getParcelableExtra<PlantModel>(BuildConfig.SELECTED_PLANT)?.
257           let {
258             {
259               bindingFragment.acPlantSelector.visibility = View.GONE
260               bindingFragment.acCropsContent.visibility = View.VISIBLE
261               viewModel.setCurrentPlant(it)
262             }
263           }
264     }
265   } catch (e: Exception){
266     Toast.makeText(this, e.message.toString(), Toast.LENGTH_SHORT).show()
267   }
268 }
269
270 private fun getNumberPickerType(view: ClickNumberPickerView) : NumberPickerType?{
271   return when (view) {
272     binding.addMonitoringTemperature -> NumberPickerType.TEMPERATURE
273     binding.addMonitoringHumidity -> NumberPickerType.HUMIDITY
274     binding.addMonitoringAcidity -> NumberPickerType.ACIDITY
275     binding.addMonitoringWaterTank -> NumberPickerType.WATER_TANK
276     binding.addMonitoringNutrientTank -> NumberPickerType.NUTRIENT_TANK
277     binding.addMonitoringWaterTurbidity -> NumberPickerType.TURBIDITY
278     else -> null
279   }
280 }
281 }
282 }

```

Kode 42: *Source Code* untuk *Add Data Monitoring Activity*

9. ADD KIT ACTIVITY

```
1  class AddKitActivity : AppCompatActivity(), TextWatcher{
2      private lateinit var binding : ActivityAddKitBinding
3      private lateinit var viewModel : AddKitViewModel
4      private lateinit var utility : ViewUtility
5      private lateinit var editTexts : ArrayList<TextInputEditText>
6      private lateinit var tooltips : ArrayList<ImageButton>
7      private lateinit var numberPickers : ArrayList<ClickNumberPickerView>
8      private var strEdt : HashMap<String, TextInputEditText> = hashMapOf()
9
10     companion object {
11         const val TAG = "addKitActivity"
12     }
13
14     override fun onCreate(savedInstanceState: Bundle?) {
15         super.onCreate(savedInstanceState)
16         binding = ActivityAddKitBinding.inflate(layoutInflater)
17         setContentView(binding.root)
18
19         viewModel = ViewModelProvider(this).get(AddKitViewModel::class.java)
20         viewModel.setCurrentData(intent.getParcelableExtra<UserModel>(BuildConfig.CURRENT_USER),
21             intent.getParcelableExtra<FarmModel>(BuildConfig.CURRENT_FARM),
22             intent.getParcelableExtra<KitModel>(BuildConfig.SELECTED_KIT))
23
24         supportActionBar?.title = viewModel.getCurrentKit().value?.name ?: getString(R.string.create_new_kit)
25         supportActionBar?.subtitle = viewModel.getCurrentFarm().value?.name
26         supportActionBar?.setDisplayHomeAsUpEnabled(true)
27         supportActionBar?.setDisplayShowHomeEnabled(false)
28
29         binding.apply {
30             setupDefaultData()
31             editTexts = arrayListOf(addKitName, addKitPosition)
32             tooltips = arrayListOf(akSizeInfo,
33                 akWaterLevelInfo,
34                 akNutrientLevelInfo,
35                 akTurbidityLevelInfo)
36             numberPickers = arrayListOf(addKitWidth,
37                 addKitLength,
38                 addKitWaterMin,
39                 addKitWaterMax,
40                 addKitNutrientMin,
41                 addKitNutrientMax,
42                 addKitTurbidityMin,
43                 addKitTurbidityMax)
44             utility = ViewUtility(context = this@AddKitActivity,
45                 circularProgressBar = addKitSubmit,
46                 textViewEdit = editTexts,
47                 numberPickers = numberPickers,
48                 actionBar = supportActionBar)
49
50             tooltips.forEach {
51                 it.setOnClickListener {
52                     it.performLongClick()
53                 }
54             }
55             akSizeInfo.tooltipText = "How many holes on each side length and width  
(must more than 0)."
56             akWaterLevelInfo.tooltipText = getString(R.string.level_tooltip)
57             akNutrientLevelInfo.tooltipText = getString(R.string.level_tooltip)
58             akTurbidityLevelInfo.tooltipText = "belum ditentuin"
59             //todo : turbidity tooltip
60
61             editTexts.forEach { it.addTextChangedListener(this@AddKitActivity) }
62             numberPickers.forEachIndexed { i, view ->
63                 val numberPickerDataType = getNumberPickerType(view)
64                 viewModel.getNumberPickerValue(numberPickerDataType)?.observe(this@AddKitActivity, {
```

```

65             if(it != view.value){
66                 view.setPickerValue(it)
67             }
68         })
69
70         numberPickers[i].setClickNumberPickerListener { previousValue,
71             currentValue, pickerClickType ->
72             viewModel.setNumberPickerValue(currentValue, numberPickerDataType)
73             checkEmpty()
74         }
75     }
76
77     addKitSubmit.setOnClickListener {
78         utility.isLoading = true
79         viewModel.createKit(binding.addKitName.text.toString(),
80                             binding.addKitPosition.text.toString())
81         viewModel.isKitAdd.observe(this@AddKitActivity, {
82             if(it){
83                 utility.isLoading = false
84                 val toastTxt = if(intent.getParcelableExtra<NoteModel>(BuildCon_
85                     ↪ fig.SELECTED_NOTE) !=
86                     ↪ null){
87                         "Kit updated."
88                     } else {
89                         "New kit created."
90                     }
91                 Toast.makeText(this@AddKitActivity, toastTxt,
92                     ↪ Toast.LENGTH_SHORT).show()
93                 super.onBackPressed()
94                 finish()
95             } else {
96                 utility.isLoading = false
97                 viewModel.addKitError.observe(this@AddKitActivity, {
98                     if(it.isNotEmpty()){
99                         Toast.makeText(this@AddKitActivity, it,
100                            ↪ Toast.LENGTH_SHORT).show()
101                         Log.i(CreateFarmFragment.TAG, it)
102                         viewModel.addKitError.value = ""
103                     }
104                 })
105             }
106         })
107     }
108     private fun setupDefaultData(){
109         binding.apply {
110             viewModel.getCurrentKit().observe(this@AddKitActivity, { it ->
111                 it.kitId?.let { kitId ->
112                     addKitName.setText(it.name)
113                     addKitPosition.setText(it.position)
114                     addKitWidth.setPickerValue(it.width?.toFloat() ?: 0f)
115                     addKitLength.setPickerValue(it.length?.toFloat() ?: 0f)
116                     addKitWaterMin.setPickerValue(it.waterLv?.min ?: 0f)
117                     addKitWaterMax.setPickerValue(it.waterLv?.max ?: 0f)
118                     addKitNutrientMin.setPickerValue(it.nutrientLv?.min ?: 0f)
119                     addKitNutrientMax.setPickerValue(it.nutrientLv?.max ?: 0f)
120                     addKitTurbidityMin.setPickerValue(it.turbidityLv?.min ?: 0f)
121                     addKitTurbidityMax.setPickerValue(it.turbidityLv?.max ?: 0f)
122
123                     strEdt[it.name ?: ""].text = addKitName
124                     strEdt[it.position ?: ""].text = addKitPosition
125                 }
126             }
127         }
128     }
129     override fun onSupportNavigateUp(): Boolean {

```

```

130     if(intent.getParcelableExtra<KitModel>(BuildConfig.SELECTED_KIT) != null){
131         viewModel.isNotEmpties.observe(this, {
132             when(it){
133                 true -> binding.addKitSubmit.performClick()
134                 else -> super.onBackPressed()
135             }
136         })
137     } else {
138         super.onBackPressed()
139     }
140     return true
141 }
142
143     override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
144         if (currentFocus != null) {
145             val imm: InputMethodManager =
146                 getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
147             imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
148         }
149         return super.dispatchTouchEvent(ev)
150     }
151
152     override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after:
153         Int) {}
154
155     override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
156         checkEmpty()
157     }
158
159     override fun afterTextChanged(s: Editable?) {}
160
161     private fun checkEmpty(){
162         val currWidth : Float = viewModel.getNumberPickerValue(getNumberPickerType(binding.addKitWidth))?.value ?: 0f
163         val currLength : Float = viewModel.getNumberPickerValue(getNumberPickerType(binding.addKitLength))?.value ?: 0f
164         val currWaterMin : Float? = viewModel.getNumberPickerValue(getNumberPickerType(binding.addKitWaterMin))?.value
165         val currWaterMax : Float? = viewModel.getNumberPickerValue(getNumberPickerType(binding.addKitWaterMax))?.value
166         val currNutrientMin : Float? = viewModel.getNumberPickerValue(getNumberPickerType(binding.addKitNutrientMin))?.value
167         val currentNutrientMax : Float? = viewModel.getNumberPickerValue(getNumberPickerType(binding.addKitNutrientMax))?.value
168         val currentTurbidityMin : Float? = viewModel.getNumberPickerValue(getNumberPickerType(binding.addKitTurbidityMin))?.value
169         val currentTurbidityMax : Float? = viewModel.getNumberPickerValue(getNumberPickerType(binding.addKitTurbidityMax))?.value
170         val hashMap : HashMap<Float?, Float?> = hashMapOf()
171         hashMap[currWaterMin] = currWaterMax
172         hashMap[currNutrientMin] = currentNutrientMax
173         hashMap[currentTurbidityMin] = currentTurbidityMax
174
175         if(intent.getParcelableExtra<KitModel>(BuildConfig.SELECTED_KIT) != null){
176             viewModel.getCurrentKit().value?.let {
177                 val floNumPick : HashMap<Float, Float> = hashMapOf()
178                 floNumPick[it.width?.toFloat() ?: 0f] = currWidth
179                 floNumPick[it.length?.toFloat() ?: 0f] = currLength
180                 floNumPick[it.waterLv?.min ?: 0f] = currWaterMin ?: 0f
181                 floNumPick[it.waterLv?.max ?: 0f] = currWaterMax ?: 0f
182                 floNumPick[it.nutrientLv?.min ?: 0f] = currNutrientMin ?: 0f
183                 floNumPick[it.nutrientLv?.max ?: 0f] = currentNutrientMax ?: 0f
184                 floNumPick[it.turbidityLv?.min ?: 0f] = currentTurbidityMin ?: 0f
185                 floNumPick[it.turbidityLv?.max ?: 0f] = currentTurbidityMax ?: 0f
186
187                 viewModel.checkNotEmpty(utility.isEmptys(editTexts) &&
188                     currWidth > 0f &&
189                     currLength > 0f &&
190                     currWaterMin > 0f &&
191                     currWaterMax > 0f &&
192                     currNutrientMin > 0f &&
193                     currentNutrientMax > 0f &&
194                     currentTurbidityMin > 0f &&
195                     currentTurbidityMax > 0f)
196             }
197         }
198     }

```

```

189         utility.isInRanges(hashMap) &&
190         (utility.isChanges(strEdt) ||
191          ↪ utility.isNumberPickerChanges(floNumPick))
192     ).observe(this@AddKitActivity, {
193       binding.addKitSubmit.isEnabled = it
194     })
195
196     Log.i(
197       TAG,
198       "checkEmpty: ${utility.isEmpties(editTexts)} ${currWidth > 0f}
199         ↪ ${currLength > 0f} ${utility.isInRanges(hashMap)}
200         ↪ ${utility.isChanges(strEdt)}
201         ↪ ${utility.isNumberPickerChanges(floNumPick)}"
202     )
203   }
204 } else {
205   viewModel.checkNotEmpty(
206     utility.isEmpties(editTexts) &&
207       currWidth > 0f &&
208       currLength > 0f &&
209       utility.isInRanges(hashMap)
210     ).observe(this@AddKitActivity, {
211       binding.addKitSubmit.isEnabled = it
212     })
213 }
214
215 private fun getNumberPickerType(view: ClickNumberPickerView) : NumberPickerType?{
216   return when (view) {
217     binding.addKitWidth -> NumberPickerType.KIT_WIDTH
218     binding.addKitLength -> NumberPickerType.KIT_LENGTH
219     binding.addKitWaterMin -> NumberPickerType.WATER_MIN
220     binding.addKitWaterMax -> NumberPickerType.WATER_MAX
221     binding.addKitNutrientMin -> NumberPickerType.NUTRIENT_MIN
222     binding.addKitNutrientMax -> NumberPickerType.NUTRIENT_MAX
223     binding.addKitTurbidityMin -> NumberPickerType.TURBIDITY_MIN
224     binding.addKitTurbidityMax -> NumberPickerType.TURBIDITY_MAX
225     else -> null
226   }
227 }
228 }

```

Kode 43: *Source Code* untuk *Add Kit Activity*

10. ADD NOTE ACTIVITY

```

1  class AddNoteActivity : AppCompatActivity(), View.OnClickListener, TextWatcher {
2    private lateinit var binding : ActivityAddNoteBinding
3    private lateinit var viewModel : AddNoteViewModel
4    private lateinit var utility : ViewUtility
5    private lateinit var editTexts : ArrayList<TextInputEditText>
6    private var strEdt : HashMap<String, TextInputEditText> = hashMapOf()
7
8    companion object {
9      const val TAG = "addNoteActivity"
10    }
11
12    override fun onCreate(savedInstanceState: Bundle?) {
13      super.onCreate(savedInstanceState)
14      binding = ActivityAddNoteBinding.inflate(layoutInflater)
15      setContentView(binding.root)
16
17      viewModel = ViewModelProvider(this).get(AddNoteViewModel::class.java)
18      viewModel.setCurrentData(intent.getParcelableExtra<NoteModel>(BuildConfig.SELECTED_NOTE))
19
20      supportActionBar?.title =
21        if(intent.getParcelableExtra<NoteModel>(BuildConfig.SELECTED_NOTE) == null)
22          {

```

```

22             getString(R.string.add_notes)
23         } else {
24             getString(R.string.edit_notes)
25         }
26         supportActionBar?.setDisplayHomeAsUpEnabled(true)
27         supportActionBar?.setDisplayShowHomeEnabled(false)
28
29         binding.apply {
30             editTexts = arrayListOf(addNoteTitle, addNoteDesc)
31             utility = ViewUtility(
32                 context = this@AddNoteActivity,
33                 circularProgressBar = addNoteSubmit,
34                 textInputEditTexts = editTexts,
35                 actionBar = supportActionBar
36             )
37
38             editTexts.forEach {
39                 it.setOnClickListener(this@AddNoteActivity)
40                 it.addTextChangedListener(this@AddNoteActivity)
41             }
42             addNoteSubmit.setOnClickListener(this@AddNoteActivity)
43
44             viewModel.getCurrentNoteModel().observe(this@AddNoteActivity, {
45                 if(it!=null) {
46                     addNoteTitle.setText(it.title)
47                     addNoteDesc.setText(it.description)
48                     it.date?.let { date->
49                         addNoteDate.setText(date)
50                     }
51                     it.time?.let{ time->
52                         addNoteTime.setText(time)
53                     }
54                     strEdt[it.title ?: ""] = addNoteTitle
55                     strEdt[it.description ?: ""] = addNoteDesc
56                     strEdt[it.date ?: ""] = addNoteDate
57                     strEdt[it.time ?: ""] = addNoteTime
58                 }
59             })
60             checkEmpty()
61         }
62     }
63
64     override fun onSupportNavigateUp(): Boolean {
65         if(intent.getParcelableExtra<NoteModel>(BuildConfig.SELECTED_NOTE) != null){
66             viewModel.isNotEmpties.observe(this, {
67                 when(it){
68                     true -> binding.addNoteSubmit.performClick()
69                     else -> super.onBackPressed()
70                 }
71             })
72         } else {
73             super.onBackPressed()
74         }
75         return true
76     }
77
78     override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
79         if (currentFocus != null) {
80             val imm: InputMethodManager =
81                 getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
82             imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
83         }
84         return super.dispatchTouchEvent(ev)
85     }
86
87     override fun onClick(v: View?) {
88         when(v){
89             binding.addNoteDate -> {
90                 val datePicker = MaterialDatePicker.Builder.datePicker()
91                     .setTitleText("Date of Birth")

```

```

92         .setSelection(viewModel.getDate())
93         .setInputMode(MaterialDatePicker.INPUT_MODE_CALENDAR)
94         .build()
95     datePicker.show(supportFragmentManager, "DATE_PICKER")
96     datePicker.addOnPositiveButtonClickListener{
97         binding.addNoteDate.setText(viewModel.setDate(datePicker.selection))
98     }
99     datePicker.addOnNegativeButtonClickListener{}
100    datePicker.isCancelable = false
101 }
102 binding.addNoteTime -> {
103     viewModel.getTime()
104     val timePicker = MaterialTimePicker.Builder()
105         .setTimeFormat(TimeFormat.CLOCK_24H)
106         .setHour(viewModel.getTimeHour())
107         .setMinute(viewModel.getTimeMinute())
108         .setTitleText("Select a time")
109         .build()
110     timePicker.show(supportFragmentManager, "TIME_PICKER")
111     timePicker.isCancelable = false
112
113     timePicker.addOnPositiveButtonClickListener {
114         binding.addNoteTime.setText(viewModel.setTime(timePicker.hour,
115             timePicker.minute))
116     }
117     timePicker.addOnNegativeButtonClickListener {}
118 }
119 binding.addNoteSubmit -> {
120     utility.isLoading = true
121     viewModel.createNote(binding.addNoteTitle.text.toString(),
122         binding.addNoteDesc.text.toString())
123     viewModel.isNoteAdd.observe(this, {
124         if(it){
125             utility.isLoading = false
126             val toastTxt = if(intent.getParcelableExtra<NoteModel>(BuildCon
127                 fig.SELECTED_NOTE) !=
128                 null){
129                 "Note updated."
130             } else {
131                 "New note created."
132             }
133             Toast.makeText(this, toastTxt, Toast.LENGTH_SHORT).show()
134             super.onBackPressed()
135             finish()
136         } else {
137             utility.isLoading = false
138             viewModel.addNoteError.observe(this, {
139                 if(it.isNotEmpty()){
140                     Toast.makeText(this, it, Toast.LENGTH_SHORT).show()
141                     Log.i(CreateFarmFragment.TAG, it)
142                     viewModel.addNoteError.value = ""
143                 }
144             })
145         }
146     })
147 }
148 override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after:
149     Int) {}
150
151 override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
152     checkEmpty()
153 }
154
155 override fun afterTextChanged(s: Editable?) {}
156
157 private fun checkEmpty(){
158     if(intent.getParcelableExtra<NoteModel>(BuildConfig.SELECTED_NOTE) != null){

```

```

158     viewModel.apply {
159         checkNotEmpty(utility.isEmptys(editTexts) && utility.isChanges(strEdt))
160         .observe(this@AddNoteActivity, {
161             binding.addNoteSubmit.isEnabled = it
162         })
163     }
164 } else {
165     viewModel.apply {
166         checkNotEmpty(utility.isEmptys(editTexts)).observe(this@AddNoteActivit
167         ↪ y,
168         ↪ {
169             binding.addNoteSubmit.isEnabled = it
170         }
171     }
172 }
```

Kode 44: *Source Code* untuk *Add Note Activity*

11. ADD PLANT ACTIVITY

```

1 class AddPlantActivity : AppCompatActivity(), TextWatcher, View.OnClickListener {
2     private lateinit var binding : ActivityAddPlantBinding
3     private lateinit var viewModel : AddPlantViewModel
4     private lateinit var utility: ViewUtility
5     private lateinit var editTexts : ArrayList<TextInputEditText>
6     private lateinit var tooltips : ArrayList<ImageButton>
7     private lateinit var numberPickers : ArrayList<ClickNumberPickerView>
8     private lateinit var viewsAsButton : ArrayList<View>
9     private var strEdt : HashMap<String, TextInputEditText> = hashMapOf()
10
11    companion object{
12        const val TAG = "addPlantActivity"
13    }
14
15    override fun onCreate(savedInstanceState: Bundle?) {
16        super.onCreate(savedInstanceState)
17        binding = ActivityAddPlantBinding.inflate(layoutInflater)
18        setContentView(binding.root)
19
20        viewModel = ViewModelProvider(this).get(AddPlantViewModel::class.java)
21        viewModel.setCurrentData(intent.getParcelableExtra<PlantModel>(BuildConfig.SELE
22        ↪ CTED_PLANT))
23
24        supportActionBar?.title =
25            if(intent.getParcelableExtra<PlantModel>(BuildConfig.SELECTED_PLANT) ==
26            null) {
27                getString(R.string.add_new_plant)
28            } else {
29                getString(R.string.edit_plant)
30            }
31        supportActionBar?.setDisplayHomeAsUpEnabled(true)
32        supportActionBar?.setDisplayShowHomeEnabled(false)
33
34        binding.apply {
35            tooltips = arrayListOf(apGrowthInfo,
36                apTempInfo,
37                apHumidInfo)
38            viewsAsButton = arrayListOf(addPlantPhoto, addPlantEditPhoto,
39            ↪ addPlantSubmit)
40            editTexts = arrayListOf(addPlantName, addPlantDescription)
41            numberPickers = arrayListOf(addPlantGrowthTime,
42                addPlantTempMin,
43                addPlantTempMax,
44                addPlantHumidMin,
45                addPlantHumidMax,
46                addPlantAcidMin,
```

```

43         addPlantAcidMax)
44     utility = ViewUtility(
45         context = this@AddPlantActivity,
46         circularProgressButton = addPlantSubmit,
47         textInputEditTexts = editTexts,
48         viewsAsButton = viewsAsButton,
49         numberPickers = numberPickers,
50         actionBar = supportActionBar)
51
52     tooltips.forEach {
53         it.setOnClickListener {
54             it.performLongClick()
55         }
56     }
57 //todo : tooltip
58
59     viewsAsButton.forEach { it.setOnClickListener(this@AddPlantActivity) }
60     addPlantPhoto.setOnLongClickListener { addPlantEditPhoto.performClick() }
61     editTexts.forEach { it.addTextChangedListener(this@AddPlantActivity) }
62     numberPickers.forEachIndexed { i, view ->
63         val numberPickerDataType = getNumberPickerType(view)
64         viewModel.getNumberPickerValue(numberPickerDataType)?.observe(this@AddP
65             → lantActivity, {
66                 if(it != view.value){
67                     view.setPickerValue(it)
68                 }
69             })
70
71         numberPickers[i].setClickNumberPickerListener { previousValue,
72             → currentValue, pickerClickType ->
73                 viewModel.setNumberPickerValue(currentValue, numberPickerDataType)
74                 checkEmpty()
75             }
76     }
77
78     viewModel.getPhotoProfile().observe(this@AddPlantActivity, {
79         it?.let {
80             Glide.with(this@AddPlantActivity)
81                 .load(it.uri)
82                 .circleCrop()
83                 .into(addPlantPhoto)
84         }
85     })
86
87     viewModel.getCurrentPlant().observe(this@AddPlantActivity, { it ->
88         it.plantId?.let { plantId ->
89             supportActionBar?.title = it.name
90             addPlantName.setText(it.name)
91             addPlantDescription.setText(it.description)
92             addPlantGrowthTime.setPickerValue(it.growthTime?.toFloat() ?: 0f)
93             addPlantTempMin.setPickerValue(it.tempLv?.min ?: 0f)
94             addPlantTempMax.setPickerValue(it.tempLv?.max ?: 0f)
95             addPlantHumidMin.setPickerValue(it.humidLv?.min ?: 0f)
96             addPlantHumidMax.setPickerValue(it.humidLv?.max ?: 0f)
97             addPlantAcidMin.setPickerValue(it.phLv?.min ?: 0f)
98             addPlantAcidMax.setPickerValue(it.phLv?.max ?: 0f)
99
100            Glide.with(this@AddPlantActivity)
101                .load(it.photo_url ?: R.drawable.bg_plant)
102                .circleCrop()
103                .into(addPlantPhoto)
104
105            if (it.userId != Firebase.auth.uid){
106                addPlantSubmit.visibility = View.GONE
107                utility.isLoading = true
108                supportActionBar?.setDisplayHomeAsUpEnabled(false)
109            }
110            Log.i(TAG, "onCreate: ${it.userId} ${Firebase.auth.uid}")
111            strEdt[it.name ?: ""] = addPlantName
112        }
113    }

```

```

111             strEdt[it.description ?: ""] = addPlantDescription
112         }
113     })
114
115     checkEmpty()
116 }
117
118
119     override fun onSupportNavigateUp(): Boolean {
120         if(intent.getParcelableExtra<PlantModel>(BuildConfig.SELECTED_PLANT) != null &&
121             viewModel.getCurrentPlant().value?.userId == Firebase.auth.uid){
122             viewModel.isNotEmpties.observe(this, {
123                 when(it){
124                     true -> binding.addPlantSubmit.performClick()
125                     else -> super.onBackPressed()
126                 }
127             })
128         } else {
129             super.onBackPressed()
130         }
131         return true
132     }
133
134     override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
135         if (currentFocus != null) {
136             val imm: InputMethodManager =
137                 getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
138             imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
139         }
140         return super.dispatchTouchEvent(ev)
141     }
142
143     override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after:
144         Int) {}
145
146     override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
147         checkEmpty()
148     }
149
150     override fun afterTextChanged(s: Editable?) {}
151
152     private fun checkEmpty(){
153         val currGrowthTime : Float = viewModel.getNumberPickerValue(getNumberPickerType(
154             (binding.addPlantGrowthTime))?.value ?:
155             Of
156
157         val currTempMin : Float? = viewModel.getNumberPickerValue(getNumberPickerType(
158             (binding.addPlantTempMin))?.value
159
160         val currTempMax : Float? = viewModel.getNumberPickerValue(getNumberPickerType(
161             (binding.addPlantTempMax))?.value
162
163         val currHumidMin : Float? = viewModel.getNumberPickerValue(getNumberPickerType(
164             (binding.addPlantHumidMin))?.value
165
166         val currHumidMax : Float? = viewModel.getNumberPickerValue(getNumberPickerType(
167             (binding.addPlantHumidMax))?.value
168
169         val currAcidMin : Float? = viewModel.getNumberPickerValue(getNumberPickerType(
170             (binding.addPlantAcidMin))?.value
171
172         val currAcidMax : Float? = viewModel.getNumberPickerValue(getNumberPickerType(
173             (binding.addPlantAcidMax))?.value
174
175         val hashMap : HashMap<Float?, Float?> = hashMapOf()
176         hashMap[currTempMin] = currTempMax
177         hashMap[currHumidMin] = currHumidMax
178         hashMap[currAcidMin] = currAcidMax
179
180
181         if(intent.getParcelableExtra<PlantModel>(BuildConfig.SELECTED_PLANT) != null){
182             viewModel.getCurrentPlant().value?.let {
183                 val floNumPick: HashMap<Float, Float> = hashMapOf()
184                 floNumPick[it.growthTime?.toFloat() ?: Of] = currGrowthTime
185                 floNumPick[it.tempLv?.min ?: Of] = currTempMin ?: Of
186                 floNumPick[it.tempLv?.max ?: Of] = currTempMax ?: Of
187                 floNumPick[it.humidLv?.min ?: Of] = currHumidMin ?: Of
188                 floNumPick[it.humidLv?.max ?: Of] = currHumidMax ?: Of
189             }
190         }
191     }

```

```

172     floNumPick[it.phLv?.min ?: 0f] = currAcidMin ?: 0f
173     floNumPick[it.phLv?.max ?: 0f] = currAcidMax ?: 0f
174
175     viewModel.checkNotEmpty(utility.isEmptys(editTexts) &&
176         currGrowthTime > 0f &&
177         utility.isInRanges(hashMap) &&
178         (utility.isChanges(strEdt) ||
179          → utility.isNumberPickerChanges(floNumPick))
180     ).observe(this@AddPlantActivity, {
181         binding.addPlantSubmit.isEnabled = it
182     })
183     Log.i(TAG, "checkEmpty: ${utility.isEmptys(editTexts)}\n" +
184         → "${currGrowthTime > 0f} ${utility.isInRanges(hashMap)}" +
185         " ${utility.isChanges(strEdt)}\n" +
186         → "${utility.isNumberPickerChanges(floNumPick)}")
187 }
188 } else {
189     viewModel.checkNotEmpty(
190         utility.isEmptys(editTexts) &&
191         currGrowthTime > 0f &&
192         utility.isInRanges(hashMap)
193     ).observe(this, {
194         binding.addPlantSubmit.isEnabled = it
195     })
196 }
197
198 private fun getNumberPickerType(view: ClickNumberPickerView) : NumberPickerType?{
199     return when (view) {
200         binding.addPlantGrowthTime -> NumberPickerType.GROWTH_TIME
201         binding.addPlantTempMin -> NumberPickerType.TEMP_MIN
202         binding.addPlantTempMax -> NumberPickerType.TEMP_MAX
203         binding.addPlantHumidMin -> NumberPickerType.HUMID_MIN
204         binding.addPlantHumidMax -> NumberPickerType.HUMID_MAX
205         binding.addPlantAcidMin -> NumberPickerType.ACID_MIN
206         binding.addPlantAcidMax -> NumberPickerType.ACID_MAX
207         else -> null
208     }
209 }
210
211 override fun onClick(v: View?) {
212     when(v){
213         binding.addPlantPhoto -> utility.openImage(binding.addPlantPhoto)
214         binding.addPlantEditPhoto -> utility.openEditPhoto(imageView =
215             → binding.addPlantPhoto, profileType = ProfileType.PLANT)
216         binding.addPlantSubmit -> {
217             utility.isLoading = true
218             viewModel.createPlant(binding.addPlantName.text.toString(),
219                 binding.addPlantDescription.text.toString())
220             viewModel.isPlantAdd.observe(this@AddPlantActivity, {
221                 if(it){
222                     utility.isLoading = false
223                     if(intent.getStringExtra("from") != null){
224                         val intent = Intent()
225                         intent.putExtra(BuildConfig.SELECTED_PLANT,
226                             → viewModel.plantModel.value)
227                         setResult(RESULT_OK, intent)
228                     } else {
229                         val toastTxt = if(intent.getParcelableExtra<PlantModel>(Bui
230                             → ldConfig.SELECTED_PLANT) !=
231                             null){
232                             "Plant updated."
233                         } else {
234                             "New plant added."
235                         }
236                         Toast.makeText(this, toastTxt, Toast.LENGTH_SHORT).show()
237                         super.onBackPressed()
238                         finish()
239                     }
240                 }
241             } else {
242                 "New plant added."
243             }
244             Toast.makeText(this, toastTxt, Toast.LENGTH_SHORT).show()
245             super.onBackPressed()
246             finish()
247         }
248     }
249 }

```

```
235     utility.isLoading = false
236     viewModel.addPlantError.observe(this@AddPlantActivity, {
237         if(it.isNotEmpty()){
238             Toast.makeText(this@AddPlantActivity, it,
239                 → Toast.LENGTH_SHORT).show()
240             Log.i(CreateFarmFragment.TAG, it)
241             viewModel.addPlantError.value = ""
242         }
243     })
244 }
245 }
246 }
247 }
248
249 val startForResult =
250     registerForActivityResult(ActivityResultContracts.StartActivityForResult()) {
251     data ->
252     Log.i(CreateProfileActivity.TAG, "$data")
253     try{
254         if(data?.resultCode == Activity.RESULT_OK){
255             CropImage.getActivityResult(data.data)?.let{
256                 val mimeTypeMap = MimeTypeMap.getSingleton()
257                 val fileExtension = mimeTypeMap.getExtensionFromMimeType(contentRes
258                 → olver.getType(it.uriContent!!))
259                 viewModel.setPhotoPlant(it.uriContent!!, fileExtension)
260             }
261         }
262     }
263
264     } catch (e: Exception){
265         Toast.makeText(this, e.message.toString(), Toast.LENGTH_SHORT).show()
266     }
267 }
```

Kode 45: *Source Code untuk Add Plant Activity*

12. PROFILE KIT ACTIVITY

```
1 class ProfileKitActivity : AppCompatActivity() {
2     private lateinit var binding : ActivityProfileKitBinding
3     lateinit var viewModel : ProfileKitViewModel
4
5     companion object {
6         const val TAG = "kitProfileActivity"
7         val TAB_LAYOUT = arrayListOf<String>("Overview", "Data Monitoring", "Crops")
8     }
9
10    override fun onCreate(savedInstanceState: Bundle?) {
11        super.onCreate(savedInstanceState)
12        binding = ActivityProfileKitBinding.inflate(layoutInflater)
13        setContentView(binding.root)
14
15        viewModel = ViewModelProvider(this).get(ProfileKitViewModel::class.java)
16        viewModel.setCurrentKit(intent.getParcelableExtra<UserModel>(BuildConfig.CURRENT_KIT,
17            T_USER),
18            intent.getParcelableExtra<FarmModel>(BuildConfig.CURRENT_FARM),
19            intent.getParcelableExtra<KitModel>(BuildConfig.SELECTED_KIT))
20        supportActionBar?.title = getString(R.string.hydroponic_kit)
21        supportActionBar?.setDisplayHomeAsUpEnabled(true)
22        supportActionBar?.setDisplayShowHomeEnabled(false)
23        supportActionBar?.elevation = 0f
24
25        val tabLayoutAdapter = object : FragmentStateAdapter(supportFragmentManager,
26            lifecycle){
27            override fun getItemCount(): Int {
28                return TAB_LAYOUT.size
29            }
30        }
31    }
32}
```

```

28
29         override fun createFragment(position: Int): Fragment {
30             return when(position){
31                 1 -> KitMonitoringFragment()
32                 2 -> KitCropsFragment()
33                 else -> KitOverviewFragment()
34             }
35         }
36     }
37
38     binding.profileKitViewPager.adapter = tabLayoutAdapter
39     TabLayoutMediator(binding.profileKitTabLayout, binding.profileKitViewPager) {
40         → tab, position ->
41             tab.text = TAB_LAYOUT[position]
42         }.attach()
43     }
44
45     override fun onSupportNavigateUp(): Boolean {
46         super.onBackPressed()
47         finish()
48         return true
49     }
50
51     override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
52         if (currentFocus != null) {
53             val imm: InputMethodManager =
54                 getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
55             imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
56         }
57         return super.dispatchTouchEvent(ev)
58     }
59
60     override fun onCreateOptionsMenu(menu: Menu?): Boolean {
61         menuInflater.inflate(R.menu.menu_profile_kit, menu)
62         return super.onCreateOptionsMenu(menu)
63     }
64
65     override fun onOptionsItemSelected(item: MenuItem): Boolean {
66         when(item.itemId){
67             R.id.kitSettings -> {
68                 val intent = Intent(this, AddKitActivity::class.java)
69                 intent.putExtra(BuildConfig.CURRENT_USER,
70                     → viewModel.getCurrentUser().value)
71                 intent.putExtra(BuildConfig.CURRENT_FARM,
72                     → viewModel.getCurrentFarm().value)
73                 intent.putExtra(BuildConfig.SELECTED_KIT,
74                     → viewModel.getCurrentKit().value)
75                 startActivity(intent)
76             }
77             R.id.kitHistory -> startActivity(Intent(this,
78                     → ShowRecyclerActivity::class.java))
79         }
80         return super.onOptionsItemSelected(item)
81     }
82 }

```

Kode 46: *Source Code* untuk *Profile Kit Activity* beserta *fragments* terkait

a) KIT OVERVIEW FRAGMENT

```

1  class KitOverviewFragment : Fragment() {
2      private var _binding : FragmentKitOverviewBinding? = null
3      private val binding get() = _binding!!
4      private lateinit var bindingCondition : RowItemKitBinding
5      private lateinit var bindingPlanted : LayoutPlantedCropsBinding
6      private lateinit var viewModel : ProfileKitViewModel
7
8      companion object {
9          const val TAG = "kitOverViewFragment"

```

```

10     }
11
12     override fun onCreateView(
13         inflater: LayoutInflater, container: ViewGroup?,
14         savedInstanceState: Bundle?
15     ): View {
16         _binding = FragmentKitOverviewBinding.inflate(inflater, container, false)
17         return binding.root
18     }
19
20     override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
21         super.onViewCreated(view, savedInstanceState)
22         viewModel = (requireActivity() as ProfileKitActivity).viewModel
23         bindingCondition = binding.kitCondition
24         bindingPlanted = binding.kitPlantedCrops
25         bindingCondition.apply {
26             kitName.visibility = View.GONE
27             kitShowDetail.visibility = View.GONE
28         }
29
30         binding.apply {
31             bindingCondition.apply {
32                 bindingPlanted.apply {
33                     viewModel.getCurrentKit().observe(viewLifecycleOwner, { kit ->
34                         kit?.let {
35                             (requireActivity() as
36                                 ProfileKitActivity).supportActionBar?.title =
37                                 kit.name.toString()
38                             if(it.isPlanted == true){
39                                 kitPlantedRoot.visibility = View.GONE
40                             } else {
41                                 kitPlantedRoot.visibility = View.GONE
42                             }
43                         })
44
45                         viewModel.getCurrentMonitoring().observe(viewLifecycleOwner, {
46                             monitorings ->
47                             monitorings?.let {
48                                 if(monitorings.size > 0)
49                                 {
50                                     monitorings[0].apply {
51                                         kitLastMonitoring.visibility = View.VISIBLE
52                                         kitTimestamp.visibility = View.VISIBLE
53                                         kitTimestamp.text =
54                                             getString(R.string.updated_on_s, timestamp)
55                                         kitTemp.text = temperature.toString()
56                                         kitHumid.text = humidity.toString()
57                                         kitAcidity.text = ph.toString()
58                                         kitWaterTank.text = waterTank.toString()
59                                         kitNutrientTank.text = nutrientTank.toString()
60                                         kitTurbidity.text = turbidity.toString()
61
62                                     }
63                                 }
64                             })
65
66                         viewModel.getCurrentCrops().observe(viewLifecycleOwner, { crops ->
67                             crops?.let{
68                                 if(crops.size > 0){
69                                     crops[0].apply {
70                                         plantModel?.let {
71                                             kitPlantTitle.text = it.name.toString()
72                                             kitPlantDate.text = this.timestamp.toString()
73
74                                         }
75                                     }
76                                 }
77                             }
78                         })
79
80                         } ?: kotlin.run {
81                             kitPlantedRoot.visibility = View.GONE
82                         }
83                     }
84                 }
85             }
86         }
87     }

```

```

76         }
77     })
78 }
79 }
80 }
81 }
82 }
```

Kode 47: Source Code untuk *KitOverview Fragment*

b) KIT MONITORING FRAGMENT

```

1  class KitMonitoringFragment : Fragment() {
2     private var _binding : FragmentKitMonitoringBinding? = null
3     private val binding get() = _binding!!
4     private lateinit var viewModel : ProfileKitViewModel
5
6     companion object {
7         const val TAG = "kitMonitoringFragment"
8     }
9
10    override fun onCreateView(
11        inflater: LayoutInflater, container: ViewGroup?,
12        savedInstanceState: Bundle?
13    ): View {
14        _binding = FragmentKitMonitoringBinding.inflate(inflater, container, false)
15        return binding.root
16    }
17
18    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
19        super.onViewCreated(view, savedInstanceState)
20        viewModel = (requireActivity() as ProfileKitActivity).viewModel
21
22        viewModel.getCurrentMonitoring().observe(viewLifecycleOwner, {
23            val tempList : ArrayList<Entry> = arrayListOf()
24            val humidList : ArrayList<Entry> = arrayListOf()
25            val acidList : ArrayList<Entry> = arrayListOf()
26            val waterList : ArrayList<Entry> = arrayListOf()
27            val nutrientList : ArrayList<Entry> = arrayListOf()
28            val turbidityList : ArrayList<Entry> = arrayListOf()
29
30            it?.forEachIndexed { index, dataMonitoringModel ->
31                tempList.add(Entry(index.toFloat(), dataMonitoringModel.temperature ?:
32                    0f))
32                humidList.add(Entry(index.toFloat(), dataMonitoringModel.humidity ?:
33                    0f))
33                acidList.add(Entry(index.toFloat(), dataMonitoringModel.ph ?: 0f))
34                waterList.add(Entry(index.toFloat(), dataMonitoringModel.waterTank ?:
35                    0f))
35                nutrientList.add(Entry(index.toFloat(),
36                    dataMonitoringModel.nutrientTank ?: 0f))
36                turbidityList.add(Entry(index.toFloat(), dataMonitoringModel.turbidity
37                    ?: 0f))
37            }
38
39            binding.apply {
40                kitCropsTemp.data = LineData(LineDataSet(tempList,
41                    getString(R.string.temperature_only)))
41                kitCropsHumid.data = LineData(LineDataSet(humidList,
42                    getString(R.string.humidity_only)))
42                kitCropsAcid.data = LineData(LineDataSet(tempList,
43                    getString(R.string.acidity_only)))
43                kitCropsWaterTank.data = LineData(LineDataSet(tempList,
44                    getString(R.string.water_level_only)))
44                kitCropsNutrientTank.data = LineData(LineDataSet(tempList,
45                    getString(R.string.nutrient_level_only)))
45                kitCropsTurbidity.data = LineData(LineDataSet(tempList,
46                    getString(R.string.turbidity_only)))
46            }
47        })
48    }
49}
```

```

47         kitCropsTemp.invalidate()
48         kitCropsHumid.invalidate()
49         kitCropsAcid.invalidate()
50         kitCropsWaterTank.invalidate()
51         kitCropsNutrientTank.invalidate()
52         kitCropsTurbidity.invalidate()
53     }
54 }
55 }
56 }
```

Kode 48: *Source Code* untuk *KitMonitoring Fragment*

c) KIT CROPS FRAGMENT

```

1  class KitCropsFragment : Fragment() {
2      private var _binding : FragmentKitCropsBinding? = null
3      private val binding get() = _binding!!
4      private lateinit var viewModel : ProfileKitViewModel
5
6      override fun onCreateView(
7          inflater: LayoutInflater, container: ViewGroup?,
8          savedInstanceState: Bundle?
9      ): View {
10         _binding = FragmentKitCropsBinding.inflate(inflater, container, false)
11         return binding.root
12     }
13
14     override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
15         super.onViewCreated(view, savedInstanceState)
16         viewModel = (requireActivity() as ProfileKitActivity).viewModel
17
18         val data : ArrayList<CropsModel> = arrayListOf()
19         val rowAdapter = AdapterType.PROFILE_CROPS.getAdapter(requireContext(), data)
20         viewModel.getCurrentCrops().observe(viewLifecycleOwner, {
21             data.clear()
22             data.addAll(it ?: arrayListOf())
23             rowAdapter.notifyDataSetChanged()
24         })
25
26         binding.kitCropsRecyclerView.apply {
27             adapter = rowAdapter
28             layoutManager = LinearLayoutManager(requireContext())
29         }
30     }
31 }
```

Kode 49: *Source Code* untuk *Kit Crops Fragment*

13. PROFILE USER ACTIVITY

```

1  class ProfileUserActivity : AppCompatActivity() {
2     private lateinit var binding : ActivityProfileUserBinding
3     private lateinit var viewModel : ProfileUserViewModel
4
5     companion object {
6         const val TAG = "profileUserActivity"
7     }
8
9     override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         binding = ActivityProfileUserBinding.inflate(layoutInflater)
12         setContentView(binding.root)
13
14         viewModel = ViewModelProvider(this).get(ProfileUserViewModel::class.java)
15         viewModel.setUserModel(intent.getParcelableExtra<UserModel>(BuildConfig.SELECTE
16             → D_USER))
```

```

16     supportActionBar?.setDisplayHomeAsUpEnabled(true)
17     supportActionBar?.setDisplayShowHomeEnabled(false)
18     supportActionBar?.elevation= 0f
19
20     binding.apply {
21         viewModel.getUserModel().observe(this@ProfileUserActivity, {
22             it?.let {
23                 supportActionBar?.title = it.name
24                 profileUserName.text = it.name
25                 it.bio?.let{
26                     profileUserBio.text = it
27                 } ?: kotlin.run {
28                     profileUserBio.visibility = View.GONE
29                 }
30                 it.photo_url?.let {
31                     Glide.with(this@ProfileUserActivity)
32                         .load(it)
33                         .circleCrop()
34                         .into(profileUserPhoto)
35                 }
36             }
37         })
38
39         profileUserPhoto.setOnClickListener {
40             IntentUtility(this@ProfileUserActivity).openImage(profileUserPhoto,
41                 viewModel.getUserModel().value?.name ?: "Photo Profile")
42         }
43     }
44 }
45
46 private fun setupRecyclerView() {
47     val userModel : ArrayList<UserModel> = arrayListOf()
48     val plantModels : ArrayList<PlantModel> = arrayListOf()
49     val rowAdapter = AdapterType.POST_PLANT.getAdapter(this, plantModels, allUsers
50     ↪ = userModel)
51     viewModel.getUserPosts().observe(this,{ 
52         plantModels.clear()
53         plantModels.addAll(it ?: arrayListOf())
54         rowAdapter.notifyDataSetChanged()
55     })
56
57     viewModel.getUserModel().observe(this, {
58         userModel.clear()
59         userModel.addAll(arrayListOf(it))
60         rowAdapter.notifyDataSetChanged()
61     })
62
63     binding.profileUserRecyclerView.apply {
64         adapter = rowAdapter
65         layoutManager = LinearLayoutManager(this@ProfileUserActivity)
66         addItemDecoration(object : DividerItemDecoration(this@ProfileUserActivity,
67             ↪ VERTICAL) {})
68         setHasFixedSize(true)
69     }
70
71     override fun onSupportNavigateUp(): Boolean {
72         super.onBackPressed()
73         finish()
74         return true
75     }
76
77     override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
78         if (currentFocus != null) {
79             val imm: InputMethodManager =
80                 getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
81             imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
82         }
83         return super.dispatchTouchEvent(ev)

```

```
83     }
84 }
```

Kode 50: Source Code untuk *Profile User Activity*

14. EDIT PROFILE FARM ACTIVITY

```
1  class EditProfileFarmActivity : AppCompatActivity(), View.OnClickListener, TextWatcher {
2      private lateinit var binding : ActivityEditProfileFarmBinding
3      private lateinit var bindingFragment : FragmentCreateFarmBinding
4      private lateinit var viewModel : CreateProfileViewModel
5      private lateinit var viewsAsButton : ArrayList<View>
6      private lateinit var utility: ViewUtility
7      private lateinit var editTexts : java.util.ArrayList<TextInputEditText>
8      private var strEdt : HashMap<String, TextInputEditText> = hashMapOf()
9
10     companion object {
11         const val TAG = "editProfileFarm"
12     }
13
14     override fun onCreate(savedInstanceState: Bundle?) {
15         super.onCreate(savedInstanceState)
16         binding = ActivityEditProfileFarmBinding.inflate(layoutInflater)
17         setContentView(binding.root)
18
19         viewModel = ViewModelProvider(this).get(CreateProfileViewModel::class.java)
20         viewModel.setCurrentFarm(intent.getParcelableExtra<FarmModel>(BuildConfig.CURRENT_FARM),
21             intent.getParcelableExtra<UserModel>(BuildConfig.CURRENT_USER))
22         supportActionBar?.title = getString(R.string.farm)
23         supportActionBar?.setDisplayHomeAsUpEnabled(true)
24         supportActionBar?.setDisplayShowHomeEnabled(false)
25
26         bindingFragment = binding.editFarmFragment
27         bindingFragment.apply {
28             setupDefaultData()
29             viewsAsButton = arrayListOf(esAddStaff, createFarmSubmit)
30             editTexts = arrayListOf(
31                 createFarmName,
32                 createFarmLoc,
33                 createFarmDesc
34             )
35             utility = ViewUtility(
36                 context = this@EditProfileFarmActivity,
37                 circularProgressBar = createFarmSubmit,
38                 editTexts = editTexts,
39                 viewsAsButton = viewsAsButton,
40                 actionBar = supportActionBar
41             )
42
43             viewsAsButton.forEach { it.setOnClickListener(this@EditProfileFarmActivity) }
44             editTexts.forEach { it.addTextChangedListener(this@EditProfileFarmActivity) }
45             setupRecyclerView()
46             checkUpdate()
47         }
48
49         val mapFragment = supportFragmentManager.findFragmentById(R.id.map) as
50             SupportMapFragment?
51         mapFragment?.getMapAsync(callback)
52     }
53
54     private fun setupDefaultData() {
55         bindingFragment.apply {
56             createFarmSubmit.text = getString(R.string.save)
57             editStaff.visibility = View.VISIBLE
58         }
59     }
60 }
```

```

57
58     viewModel.getCurrentFarmModel()?.let {
59         createFarmName.setText(it.name ?: "")
60         createFarmDesc.setText(it.description ?: "")
61         createFarmLoc.setText(it.location ?: "")
62
63         strEdt[it.name ?: ""] = createFarmName
64         strEdt[it.description ?: ""] = createFarmDesc
65         strEdt[it.location ?: ""] = createFarmLoc
66     }
67 }
68
69
70 private fun setupRecyclerView(){
71     val data : ArrayList<UserModel> = arrayListOf()
72     val rowAdapter = AdapterType.SEARCH_USER.getAdapter(this, data, type =
73         AdapterType.Companion.SearchSelectType.SELECT)
74     viewModel.updateStaff()
75     viewModel.getStaff().observe(this, {
76         data.clear()
77         data.addAll(it ?: arrayListOf())
78         rowAdapter.notifyDataSetChanged()
79         (rowAdapter as UserModelAdapter).setOnItemClickListener(
80             object : UserModelAdapter.OnItemClickListener{
81                 override fun onItemClick(
82                     userModel: UserModel,
83                     position: Int,
84                     itemView: View,
85                     v: RowItemSearchBinding
86                 ) {
87                     when(itemView){
88                         v.searchRoot -> {
89                             val intent = Intent(this@EditProfileFarmActivity,
90                                 ProfileUserActivity::class.java)
91                             intent.putExtra(BuildConfig.SELECTED_USER, userModel)
92                             startActivity(intent)
93                         }
94                         v.searchClose -> viewModel.removeStaff(position)
95                     }
96                 }
97             }
98         )
99     })
100     bindingFragment.esRecyclerView.adapter = rowAdapter
101     bindingFragment.esRecyclerView.layoutManager = LinearLayoutManager(this)
102     bindingFragment.esRecyclerView.addItemDecoration(object :
103         DividerItemDecoration(this, VERTICAL) {})
104     bindingFragment.esRecyclerView.setHasFixedSize(true)
105 }
106
107 private fun checkUpdate(){
108     viewModel.checkNotEmpty(
109         utility.isChanges(strEdt) && utility.isEmptys(editTexts)
110     ).observe(this, {
111         bindingFragment.createFarmSubmit.isEnabled = it
112     })
113
114     override fun onSupportNavigateUp(): Boolean {
115         viewModel.isNotEmptys.observe(this, {
116             when(it){
117                 true -> bindingFragment.createFarmSubmit.performClick()
118                 else -> super.onBackPressed()
119             }
120         })
121         return true
122     }
123
124     override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {

```

```

124     if (currentFocus != null) {
125         val imm: InputMethodManager =
126             getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
127         imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
128     }
129     return super.dispatchTouchEvent(ev)
130 }
131
132 override fun onClick(v: View?) {
133     when(v){
134         bindingFragment.esAddStaff -> {
135             val intent = Intent(this, SearchActivity::class.java)
136             intent.putExtra("search", TAG)
137             intent.putExtra(BuildConfig.EXCEPT_USERS, viewModel.getStaff().value)
138             startForResult.launch(intent)
139         }
140         bindingFragment.createFarmSubmit -> {
141             utility.isLoading = true
142             viewModel.createFarmProfile(bindingFragment.createFarmName.text.toString()
143             → g(),
144                 bindingFragment.createFarmDesc.text.toString(),
145                 bindingFragment.createFarmLoc.text.toString())
146             viewModel.isFarmCreated.observe(this, {
147                 if(it){
148                     utility.isLoading = false
149                     Toast.makeText(this, "Farm updated.", Toast.LENGTH_SHORT).show()
150                     super.onBackPressed()
151                 } else {
152                     utility.isLoading = false
153                     viewModel.createProfileError.observe(this, {
154                         if(it.isNotEmpty()){
155                             Toast.makeText(this, it, Toast.LENGTH_SHORT).show()
156                             Log.i(CreateFarmFragment.TAG, it)
157                             viewModel.createProfileError.value = ""
158                         }
159                     })
160                 }
161             })
162             viewModel.createStaff()
163             viewModel.isStaffAdded.observe(this, {
164                 if(it){
165                     utility.isLoading = false
166                     Toast.makeText(this, "Farm updated", Toast.LENGTH_SHORT).show()
167                     super.onBackPressed()
168                     finish()
169                 }
170             })
171         }
172     }
173
174     private val callback = OnMapReadyCallback { googleMap ->
175         val sydney = LatLng(-34.0, 151.0)
176         googleMap.addMarker(MarkerOptions().position(sydney).title("Marker in Sydney"))
177         googleMap.moveCamera(CameraUpdateFactory.newLatLng(sydney))
178         //todo posisi awal sesuai dari data starter
179     }
180
181     private val startForResult =
182         registerForActivityResult(ActivityResultContracts.StartActivityForResult()) {
183             data ->
184             Log.i(CreateProfileActivity.TAG, "${data}")
185             try{
186                 if(data?.resultCode == Activity.RESULT_OK){
187                     data.data?.getParcelableExtra<UserModel>(BuildConfig.SELECTED_USER)?.let {
188                         t
189                         {
190                             viewModel.addStaff(it)
191                             checkUpdate()
192                         }
193                     }
194                 }
195             }
196         }

```

```

189         }
190
191     } catch (e: Exception){
192         Toast.makeText(this, e.message.toString(), Toast.LENGTH_SHORT).show()
193     }
194 }
195
196 override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after:
197 → Int) {}
198
199 override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
200     checkUpdate()
201 }
202
203 override fun afterTextChanged(s: Editable?) {}

```

Kode 51: *Source Code untuk Edit Profile Farm Activity*

15. EDIT PROFILE USER ACTIVITY

```

1 class EditProfileUserActivity : AppCompatActivity(), View.OnClickListener,
2 → SegmentedButtonGroup.OnPositionChangedListener, TextWatcher {
3     private lateinit var binding : ActivityEditProfileUserBinding
4     private lateinit var bindingFragment : FragmentCreateUserBinding
5     private lateinit var viewModel : CreateProfileViewModel
6     private lateinit var utility: ViewUtility
7     private lateinit var viewsAsButton : ArrayList<View>
8     private lateinit var editTexts : ArrayList<TextInputEditText>
9     private var strEdt : HashMap<String, TextInputEditText> = hashMapOf()
10
11     companion object {
12         const val TAG = "editProfileUserActivity"
13     }
14
15     override fun onCreate(savedInstanceState: Bundle?) {
16         super.onCreate(savedInstanceState)
17         binding = ActivityEditProfileUserBinding.inflate(layoutInflater)
18         setContentView(binding.root)
19
20         viewModel = ViewModelProvider(this).get(CreateProfileViewModel::class.java)
21         viewModel.setCurrentUser(intent.getParcelableExtra<UserModel>(BuildConfig.CURRE_
22 → NT_USER))
23         supportActionBar?.title = getString(R.string.edit_profile)
24         supportActionBar?.setDisplayHomeAsUpEnabled(true)
25         supportActionBar?.setDisplayShowHomeEnabled(false)
26
27         bindingFragment = binding.editProfileFragment
28         bindingFragment.apply {
29             setupDefaultData()
30             editTexts = arrayListOf(
31                 createUserName,
32                 createUserPhone,
33                 createUserAddress,
34                 createUserDOB,
35                 createUserBio)
36             viewsAsButton = arrayListOf(createUserDOB,
37                                         createUserSubmit,
38                                         createUserEditPhoto,
39                                         createUserPhoto)
40             utility = ViewUtility(
41                 context = this@EditProfileUserActivity,
42                 circularProgressBar = createUserSubmit,
43                 editTexts = editTexts,
44                 viewsAsButton = viewsAsButton,
45                 actionBar = supportActionBar
46             )
47     }

```

```
46
47     createUserGender.onPositionChangedListener = this@EditProfileUserActivity
48     editTexts.forEach { it.addTextChangedListener(this@EditProfileUserActivity)
49         ↪ }
50     viewsAsButton.forEach { it.setOnClickListener(this@EditProfileUserActivity)
51         ↪ }
52     createUserPhoto.setOnLongClickListener { createUserEditPhoto.performClick()
53         ↪ }
54     checkUpdate()
55 }
56
57
58
59
60
61
62
63     viewModel.getPhotoProfile().observe(this, {
64         it?.let {
65             Glide.with(this)
66                 .load(it.uri)
67                 .circleCrop()
68                 .into(bindingFragment.createUserPhoto)
69         }
70     })
71 }
72
73
74
75
76
77
78
79
80
81
82     override fun onSupportNavigateUp(): Boolean {
83         viewModel.isNotEmpty.observe(this, {
84             when(it){
85                 true -> bindingFragment.createUserSubmit.performClick()
86                 else -> super.onBackPressed()
87             }
88         })
89         return true
90     }
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2296
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2396
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473

```

```

109         bindingFragment.createUserPhone.text.toString(),
110         bindingFragment.createUserAddress.text.toString(),
111         bindingFragment.createUserBio.text.toString()
112     )
113     viewModel.isUserCreated.observe(this, {
114         if(it){
115             utility.isLoading = false
116             Toast.makeText(this, "Profile updated.",
117                         Toast.LENGTH_SHORT).show()
118             super.onBackPressed()
119             finish()
120         } else {
121             viewModel.createProfileError.observe(this, {
122                 if(it.isNotEmpty()){
123                     Toast.makeText(this, it, Toast.LENGTH_SHORT).show()
124                     viewModel.createProfileError.value = ""
125                 }
126             })
127         }
128     })
129 }
130 }
131 }
132
133 override fun onPositionChanged(position: Int) {
134     when(position){
135         0 -> viewModel.setGender(Gender.MALE)
136         1 -> viewModel.setGender(Gender.FEMALE)
137     }
138     checkUpdate()
139 }
140
141 private fun setupDefaultData(){
142     bindingFragment.apply {
143         createUserRoleGroup.visibility = View.GONE
144         createUserSubmit.text = getString(R.string.save)
145
146         viewModel.getCurrentUserModel()?.let {
147             createUserEmail.setText(Firebase.auth.currentUser?.email)
148             createUserName.setText(it.name ?: "")
149             createUserPhone.setText(it.phone ?: "")
150             createUserAddress.setText(it.address ?: "")
151             createUserBio.setText(it.bio ?: "")
152             createUserDOB.setText(it.dob ?: "")
153             Glide.with(this@EditProfileUserActivity)
154                 .load(it.photo_url ?: R.drawable.bg_farmer)
155                 .centerCrop()
156                 .into(createUserPhoto)
157             createUserGender.setPosition(Gender.getType(it.gender!!)!!.getPosition(
158                 ),
159                 ),
160                 false)
161
162             strEdt[it.name ?: ""] = createUserName
163             strEdt[it.phone ?: ""] = createUserPhone
164             strEdt[it.address ?: ""] = createUserAddress
165             strEdt[it.bio ?: ""] = createUserBio
166             strEdt[it.dob ?: ""] = createUserDOB
167         }
168     }
169 }
170
171
172
173
174

```

```

175
176     private fun checkUpdate() {
177         viewModel.checkNotEmpty(
178             utility.isEmptys(editTexts) &&
179                 (utility.isChanges(strEdt) ||
180                  viewModel.getUpdateGender() != bindingFragment.createUserGender.position ||
181                  viewModel.getPhotoProfile().value != null)
182         ).observe(this, {
183             bindingFragment.createUserSubmit.isEnabled = it
184         })
185     }
186
187     val startForResult =
188         registerForActivityResult(ActivityResultContracts.StartActivityForResult()) {
189             data ->
190             Log.i(TAG, "$data")
191             try{
192                 if(data?.resultCode == Activity.RESULT_OK){
193                     CropImage.getActivityResult(data.data)?.let{
194                         val mimeTypeMap = MimeTypeMap.getSingleton()
195                         val fileExtension = mimeTypeMap.getExtensionFromMimeType(contentRes)
196                         olver.getType(it.uriContent!!)
197                         viewModel.setPhotoProfile(it.uriContent!!, fileExtension)
198                         checkUpdate()
199                     }
200                 }
201             } catch (e: Exception){
202                 Toast.makeText(this, e.message.toString(), Toast.LENGTH_SHORT).show()
203             }
204         }
205     }

```

Kode 52: *Source Code untuk Edit Profile User Activity*

16. SEARCH MODEL

```

1  class SearchActivity : AppCompatActivity(), SearchView.OnQueryTextListener,
2     SwipeRefreshLayout.OnRefreshListener {
3     private lateinit var binding : ActivitySearchBinding
4     private lateinit var viewModel : SearchViewModel
5     private lateinit var objectSearch : ProfileType
6     private lateinit var rowAdapter : RecyclerView.Adapter<*>
7
7     companion object {
8         const val TAG = "searchActivity"
9     }
10
11    override fun onCreate(savedInstanceState: Bundle?) {
12        super.onCreate(savedInstanceState)
13        binding = ActivitySearchBinding.inflate(layoutInflater)
14        setContentView(binding.root)
15
16        objectSearch = ProfileType.getType(intent.getStringExtra("search").toString())
17
18        viewModel = ViewModelProvider(this).get(SearchViewModel::class.java)
19        supportActionBar?.title = getString(R.string.search) + " $objectSearch"
20        supportActionBar?.elevation = 0f
21        supportActionBar?.setDisplayHomeAsUpEnabled(true)
22        supportActionBar?.setDisplayShowHomeEnabled(false)
23
24        val searchManager : SearchManager = getSystemService(Context.SEARCH_SERVICE) as
25            SearchManager
26        binding.searchView.setSearchableInfo(searchManager.getSearchableInfo(componentName))
27        binding.searchView.setOnQueryTextListener(this)
28        binding.searchRefresh.setOnRefreshListener(this)

```

```

29         when(objectSearch){
30             ProfileType.PLANT -> setupRecyclerViewPlant()
31             ProfileType.USER -> setupRecyclerViewUser()
32         }
33     }
34
35     private fun setupRecyclerViewPlant(){
36         val plantModels : ArrayList<PlantModel> = arrayListOf()
37         val allUsers : ArrayList<UserModel> = viewModel.getAllUsers().value ?:
38             arrayListOf()
39         viewModel.getAllPlants()
40         rowAdapter = AdapterType.SEARCH_PLANT.getAdapter(this, plantModels,
41             allUsers,
42             type = AdapterType.Companion.SearchSelectType.SEARCH)
43         viewModel.getPlants().observe(this, {
44             plantModels.clear()
45             plantModels.addAll(it ?: arrayListOf())
46             rowAdapter.notifyDataSetChanged()
47             (rowAdapter as PlantModelAdapter).setOnItemClickListener(
48                 object : PlantModelAdapter.OnItemClickListener{
49                     override fun onItemClickClicked(
50                         position: Int,
51                         itemView: View,
52                         v: RowItemSearchBinding
53                     ) {
54                         when(itemView){
55                             v.searchRoot -> {
56                                 val intent = Intent()
57                                 intent.putExtra(BuildConfig.SELECTED_PLANT,
58                                     viewModel.getPlant(position))
59                                 setResult(RESULT_OK, intent)
60                                 this@SearchActivity.onBackPressed()
61                                 finish()
62                             }
63                         }
64                     }
65                 }
66             )
67             binding.searchRefresh.isRefreshing = false
68             Log.i(TAG, "setupRecyclerViewPlant: ${rowAdapter.itemCount}")
69         })
70         viewModel.getAllUsers().observe(this, {
71             allUsers.clear()
72             allUsers.addAll(it ?: arrayListOf())
73             rowAdapter.notifyDataSetChanged()
74         })
75         setupRecyclerView()
76     }
77
78     private fun setupRecyclerViewUser(){
79         viewModel.setExceptUsers(intent.getParcelableArrayListExtra<UserModel>(BuildCon
80             ↪ fig.EXCEPT_USERS))
81         val data : ArrayList<UserModel> = arrayListOf()
82         rowAdapter = AdapterType.SEARCH_USER.getAdapter(this, data,
83             type = AdapterType.Companion.SearchSelectType.SEARCH)
84         viewModel.getUsers().observe(this, {
85             data.clear()
86             data.addAll(it ?: arrayListOf())
87             rowAdapter.notifyDataSetChanged()
88             (rowAdapter as UserModelAdapter).setOnItemClickListener(
89                 object : UserModelAdapter.OnItemClickListener{
90                     override fun onItemClickClicked(
91                         userModel: UserModel,
92                         position: Int,
93                         itemView: View,
94                         v: RowItemSearchBinding
95                     ) {
96                         when(itemView){
97                             v.searchRoot -> {

```

```

96             val intent = Intent()
97             intent.putExtra(BuildConfig.SELECTED_USER,
98             → viewModel.getUser(position))
99             setResult(RESULT_OK, intent)
100            this@SearchActivity.onBackPressed()
101            finish()
102        }
103    }
104 }
105 binding.searchRefresh.isRefreshing = false
106 Log.i(TAG, "setupRecyclerViewUser: ${rowAdapter.itemCount}")
107 })
108 setupRecyclerView()
109 }
110
111 private fun setupRecyclerView(){
112     binding.searchRecyclerView.adapter = rowAdapter
113     binding.searchRecyclerView.layoutManager = LinearLayoutManager(this)
114     binding.searchRecyclerView.addItemDecoration(object :
115         → DividerItemDecoration(this, VERTICAL) {})
116     binding.searchRecyclerView.setHasFixedSize(true)
117 }
118
119 override fun onSupportNavigateUp(): Boolean {
120     super.onBackPressed()
121     return true
122 }
123
124 override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
125     if (currentFocus != null) {
126         val imm: InputMethodManager =
127             getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
128         imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
129     }
130     return super.dispatchTouchEvent(ev)
131 }
132
133 private fun searchKeyword(key:String?) : Boolean{
134     when(objectSearch){
135         ProfileType.PLANT -> viewModel.searchPlants(key)
136         ProfileType.USER -> viewModel.searchUsers(key)
137         else -> Log.i(TAG, "enum class not found")
138     }
139     return true
140 }
141
142 override fun onQueryTextSubmit(query: String?): Boolean {
143     binding.searchView.clearFocus()
144     return searchKeyword(query)
145 }
146
147 override fun onQueryTextChange(newText: String?): Boolean {
148     return searchKeyword(newText)
149 }
150
151 override fun onRefresh() {
152     when(objectSearch){
153         ProfileType.PLANT -> {
154             viewModel.getAllPlants(binding.searchView.query.toString())
155         }
156         ProfileType.USER ->
157             → viewModel.getAllUsers(binding.searchView.query.toString())
158         else -> Log.i(TAG, "enum class not found")
159     }
160 }

```

Kode 53: *Source Code* untuk *Search Activity*

17. FEEDBACK ACTIVITY

```
1  class FeedbackActivity : AppCompatActivity(), TextWatcher,
2    → RatingBar.OnRatingBarChangeListener {
3      private lateinit var binding : ActivityFeedbackBinding
4      private lateinit var viewModel : FeedbackViewModel
5      private lateinit var utility : ViewUtility
6
6      companion object {
7        const val TAG = "feedback"
8      }
9
10     override fun onCreate(savedInstanceState: Bundle?) {
11       super.onCreate(savedInstanceState)
12       binding = ActivityFeedbackBinding.inflate(layoutInflater)
13       setContentView(binding.root)
14
15       viewModel = ViewModelProvider(this).get(FeedbackViewModel::class.java)
16       supportActionBar?.title = getString(R.string.feedback)
17       supportActionBar?.setDisplayHomeAsUpEnabled(true)
18       supportActionBar?.setDisplayShowHomeEnabled(false)
19
20       binding.apply {
21         utility = ViewUtility(context = this@FeedbackActivity,
22           circularProgressButton = feedbackSubmit,
23           textInputEditTexts = arrayListOf(feedbackContent),
24           actionBar = supportActionBar)
25
26         feedbackRatingbar.rating = 5f
27         feedbackRatingbar.onRatingBarChangeListener = this@FeedbackActivity
28         feedbackContent.addTextChangedListener(this@FeedbackActivity)
29         checkEmpty()
30
31         feedbackSubmit.setOnClickListener {
32           utility.isLoading = true
33           viewModel.submitFeedback(feedbackContent.text.toString())
34           viewModel.isFeedbackSubmit.observe(this@FeedbackActivity, {
35             if(it){
36               utility.isLoading = false
37               Toast.makeText(this@FeedbackActivity, "Thank you for your
38               → feedback.", Toast.LENGTH_SHORT).show()
39               super.onBackPressed()
39               finish()
40             } else {
41               utility.isLoading = false
42               viewModel.feedbackSubmitError.observe(this@FeedbackActivity, {
43                 if(it.isNotEmpty()){
44                   Toast.makeText(this@FeedbackActivity, it,
45                   → Toast.LENGTH_SHORT).show()
45                   Log.i(SignInActivity.TAG, it)
46                   viewModel.feedbackSubmitError.value = ""
47                 }
48               })
49             }
50           })
51         }
52       }
53     }
54
55     override fun onSupportNavigateUp(): Boolean {
56       super.onBackPressed()
57       return true
58     }
59
60     override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
61       if (currentFocus != null) {
62         val imm: InputMethodManager =
63           getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
64         imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
65       }
66     }
67   }
68 }
```

```

66         return super.dispatchTouchEvent(ev)
67     }
68
69     override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after:
70         Int) {}
71
72     override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
73         checkEmpty()
74     }
75
76     override fun afterTextChanged(s: Editable?) {}
77
78     private fun checkEmpty() {
79         viewModel.checkNotEmpty(
80             utility.isEmpty(binding.feedbackContent) &&
81             viewModel.getRating() ?: 0f > 0f
82         ).observe(this, {
83             binding.feedbackSubmit.isEnabled = it
84         })
85     }
86
87     override fun onRatingChanged(ratingBar: RatingBar?, rating: Float, fromUser:
88         Boolean) {
89         viewModel.setRating(rating)? .observe(this, {
90             it.rating?.let {
91                 binding.feedbackRatingtxt.text = Rating.getType(rating).toString()
92             }
93         })
94     }

```

Kode 54: *Source Code* untuk *Feedback Activity*

18. SHOW RECYCLER ACTIVITY

```

1  class ShowRecyclerActivity : AppCompatActivity() {
2     private lateinit var binding : ActivityShowRecyclerBinding
3     private lateinit var viewModel : ShowRecyclerViewModel
4
5     override fun onCreate(savedInstanceState: Bundle?) {
6         super.onCreate(savedInstanceState)
7         binding = ActivityShowRecyclerBinding.inflate(layoutInflater)
8         setContentView(binding.root)
9
10        viewModel = ViewModelProvider(this).get(ShowRecyclerViewModel::class.java)
11        viewModel.setCurrentPosts(intent.getParcelableArrayListExtra(BuildConfig.ALL_PL
12        ↵ ANTS),
13            intent.getParcelableExtra<UserModel>(BuildConfig.CURRENT_USER))
14        supportActionBar?.title = getString(R.string.history)
15        supportActionBar?.setDisplayHomeAsUpEnabled(true)
16        supportActionBar?.setDisplayShowHomeEnabled(false)
17
18        setupRecyclerView()
19    }
20
21    private fun setupRecyclerView() {
22        val plantModels : ArrayList<PlantModel> = arrayListOf()
23        val userModels : ArrayList<UserModel> = arrayListOf()
24        val rowAdapter = AdapterType.POST_PLANT.getAdapter(this, plantModels, allUsers
25        ↵ = userModels)
26        viewModel.getCurrentPosts().observe(this, {
27            plantModels.clear()
28            plantModels.addAll(it ?: arrayListOf())
29            rowAdapter.notifyDataSetChanged()
30            (rowAdapter as PostModelAdapter).setOnItemClickListener(
31                object : PostModelAdapter.OnItemClickListener{
32                    override fun onItemClickClicked(
33                        position: Int,
34

```

```

32             itemView: View,
33             v: RowItemPostBinding
34         ) {
35             when(itemView){
36                 v.postRoot -> {
37                     gotoPost(position)
38                 }
39                 v.postOptions -> {
40                     val items = arrayOf("Edit", "Delete")
41                     MaterialAlertDialogBuilder(this@ShowRecyclerActivity)
42                         .setItems(items){_, which ->
43                             when(which){
44                                 0 -> gotoPost(position)
45                                 1 -> {
46                                     viewModel.deletePost(position)
47                                     viewModel.isPostDeleted.observe(this@ShowRecyclerActivity, {
48                                         it -> {
49                                             when(it){
50                                                 true -> Toast.makeText(this@ShowRecyclerActivity, "Post deleted.", Toast.LENGTH_SHORT).show()
51                                             false -> Log.e(MainPostsFragment.TAG, "Post not deleted")
52                                         }
53                                         }.show()
54                                     }
55                                 }
56                             }
57                         }
58                     }
59                 }
60             }
61         }
62         Log.i(MainPostsFragment.TAG, "${rowAdapter.itemCount}")
63     })
64
65     viewModel.getCurrentUser().observe(this, {
66         userModels.clear()
67         userModels.addAll(arrayListOf(it))
68         rowAdapter.notifyDataSetChanged()
69     })
70
71     binding.recyclerView.apply {
72         adapter = rowAdapter
73         layoutManager = LinearLayoutManager(this@ShowRecyclerActivity)
74         addItemDecoration(object : DividerItemDecoration(this@ShowRecyclerActivity, VERTICAL) {})
75         setHasFixedSize(true)
76     }
77 }
78
79 override fun onSupportNavigateUp(): Boolean {
80     super.onBackPressed()
81     return true
82 }
83
84 override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
85     if (currentFocus != null) {
86         val imm: InputMethodManager =
87             getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
88         imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
89     }
90     return super.dispatchTouchEvent(ev)
91 }
92
93 private fun gotoPost(position:Int){
94     val intent = Intent(this, AddPlantActivity::class.java)
95     intent.putExtra(BuildConfig.SELECTED_PLANT, viewModel.getPost(position))

```

```
96         startActivity(intent)
97     }
98 }
```

Kode 55: *Source Code* untuk *Show Recycler Activity*

19. SHOW PICTURE ACTIVITY

```
1  class ShowPictureActivity : AppCompatActivity() {
2      private lateinit var binding : ActivityShowPictureBinding
3
4      private var currentImage : Bitmap? = null
5      override fun onCreate(savedInstanceState: Bundle?) {
6          super.onCreate(savedInstanceState)
7          binding = ActivityShowPictureBinding.inflate(layoutInflater)
8          setContentView(binding.root)
9
10         val titleBar = intent.getStringExtra(" actionBarTitle")
11
12         window.statusBarColor = Color.BLACK
13         supportActionBar?.setDisplayHomeAsUpEnabled(true)
14         supportActionBar?.setDisplayHomeAsUpEnabled(true)
15         supportActionBar?.setBackgroundDrawable(ColorDrawable(ContextCompat.getColor(th
16             ↵ is,
17             ↵ R.color.black)))
18         supportActionBar?.title = if(titleBar.isNullOrEmpty()){"Profile Picture"}
19             ↵ else{titleBar}
20
21         val filename = intent.getStringExtra(" photoContent")
22         try {
23             val `is`: FileInputStream = openFileInput(filename)
24             currentImage = BitmapFactory.decodeStream(`is`)
25             `is`.close()
26         } catch (e: Exception) {
27             e.printStackTrace()
28         }
29         binding.showPicture.setImageBitmap(currentImage)
30
31     override fun onSupportNavigateUp(): Boolean {
32         super.onBackPressed()
33         return true
34     }
}
```

Kode 56: *Source Code* untuk *Show Picture Activity*

20. ABOUT ME ACTIVITY

```
1  class AboutMeActivity : AppCompatActivity() {
2      private lateinit var binding: ActivityAboutMeBinding
3      private var intentData = IntentUtility(this)
4
5      private fun setupToolbar(){
6          supportActionBar?.setDisplayHomeAsUpEnabled(true)
7          supportActionBar?.setDisplayHomeAsUpEnabled(false)
8          supportActionBar?.title = getString(R.string.about_me)
9      }
10
11     private fun setupHeader(){
12         binding.aboutEmail.text = BuildConfig.CREATOR_EMAIL
13         binding.aboutDescription.text = listOf(getString(R.string.tab),
14             ↵ getString(R.string.profile_description)).joinToString(" ")
15         Glide.with(this)
16             .load(R.drawable.profile_photo)
17             .circleCrop()
```

```

17     .into(binding.profilePicture)
18 }
19
20 private fun setupFindMe(){
21     binding.aboutFindMe.itemIconTintList = null
22     binding.aboutFindMe.setNavigationItemSelectedListener {
23         when(it.itemId) {
24             R.id.about_whatsapp -> {
25                 try{
26                     val intent = Intent()
27                     applicationContext.packageManager.getPackageInfo("com.whatsapp" ]
28                         ,
29                         PackageManager.GET_META_DATA)
30                     intent.action = Intent.ACTION_VIEW
31                     intent.type = "text/plain"
32                     intent.data = Uri.parse(BuildConfig.CREATOR_WHATSAPP)
33                     intent.setPackage("com.whatsapp")
34                     startActivity(intent)
35                 } catch (e : PackageManager.NameNotFoundException) {
36                     Toast.makeText(this, e.toString(), Toast.LENGTH_SHORT).show()
37                 }
38                 true
39             R.id.about_email ->{
40                 intentData.openEmail(BuildConfig.CREATOR_EMAIL)
41                 true
42             R.id.about_github -> {
43                 intentData.openBrowser(BuildConfig.CREATOR_GITHUB)
44                 true
45             R.id.about_project -> {
46                 intentData.openBrowser(BuildConfig.PROJECT_URL)
47                 true
48             else -> false
49         }
50     }
51
52     private fun setupCredit() {
53         binding.aboutCredit.itemIconTintList = null
54         binding.aboutCredit.setNavigationItemSelectedListener {
55             when (it.itemId) {
56                 R.id.about_lottie ->
57                     intentData.openBrowser("https://lottiefiles.com/45869-farmers")
58                 R.id.about_freepik -> intentData.openBrowser("https://www.freepik.com/")
59             }
60             true
61         }
62
63     override fun onCreate(savedInstanceState: Bundle?) {
64         super.onCreate(savedInstanceState)
65         binding = ActivityAboutMeBinding.inflate(layoutInflater)
66         setContentView(binding.root)
67         setSupportActionBar(binding.toolbar)
68         setupHeader()
69         setupFindMe()
70         setupCredit()
71     }
72
73     override fun onSupportNavigateUp(): Boolean {
74         onBackPressed()
75         return true
76     }
77 }

```

Kode 57: *Source Code* untuk *About Me Activity*