KERJA PRAKTIK - IF184801

Implementasi Aplikasi Monitoring Kinerja FTEIC ITS berbasis Android

Fakultas Teknologi Elektro dan Informatika Cerdas - ITS Surabaya
Gedung Rektorat Lt.3 Kampus ITS Sukolilo Surabaya
Periode: 1 Agustus 2020 - 31 Oktober 2020

Oleh:
M. Rafi Fadhilah 05111740000075

Pembimbing Jurusan
Waskitho Wibisono, S.Kom., M.Eng., Ph.D
Pembimbing Lapangan
Dr. I Ketut Eddy Purnama, S.T., M.T.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

KERJA PRAKTIK – IF184801

Implementasi Aplikasi Monitoring Kinerja FTEIC ITS berbasis Android

Fakultas Teknologi Elektro dan Informatika Cerdas – ITS Surabaya
Gedung Rektorat Lt.3 Kampus ITS Sukolilo Surabaya
Periode: 1 Agustus 2020 – 31 Oktober 2020

Oleh:
M. Rafi Fadhilah 05111740000075

Pembimbing Jurusan
Waskitho Wibisono, S.Kom., M.Eng., Ph.D
Pembimbing Lapangan
Dr. I Ketut Eddy Purnama, S.T., M.T.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN
KERJA PRAKTIK

Implementasi Aplikasi Monitoring Kinerja FTEIC ITS berbasis Android

Oleh:

M. Rafi Fadhilah 05111740000075

Disetujui oleh Pembimbing Kerja Praktik:

1. Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
   NIP. 197512202001122002

(Pembimbing Departemen)

2. Dr. I Ketut Eddy Purnama, S.T., M.T.
   NIP. 196907301995121001

(Pembimbing Lapangan)

[Halaman ini sengaja dikosongkan]

Implementasi Aplikasi Monitoring Kinerja FTEIC ITS berbasis Android

Nama Mahasiswa : M. Rafi Fadhilah
NRP : 05111740000075
Departemen : Informatika FTEIC-ITS
Pembimbing Jurusan : Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

ABSTRAK

Dosen serta tenaga kependidikan di Fakultas Teknologi Elektro dan Informatika Cerdas memilki *track record* yang sangat banyak tiap tahunnya. Mulai dari prestasi, kuliah tamu, training, sertifikasi, konferensi dan jurnal. Karena masalah tersebut, diperlukan adanya suatu sistem informasi untuk menampung semua catatan yang ada agar data yang tersimpan dapat digunakan dengan lebih baik.

Sehingga perlu dibangun aplikasi untuk melakukan monitoring terhadap kinerja yang telah dilakukan terhadap semua kegiatan yang terjadi di FTEIC. Aplikasi ini mampu untuk melakukan proses administratif sekaligus melakukan masukan setiap catatan pada kategorinya masing-masing.

Pembuatan aplikasi menggunakan beberapa *framework*. Untuk pembuatan aplikasi menggunakan Android Studio sebagai IDE dengan bahasa pemrograman Kotlin. Lalu untuk *backend* menggunakan Django. basis data yang digunakan adalah MySQL.

Kata kunci:
FTEIC, Monitoring, Android

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena atas berkat limpahan rahmat dan lindungan-Nya penulis dapat melaksanakan salah satu kewajiban sebagai mahasiswa Teknik Informatika ITS yaitu Kerja Praktik (KP).

Penulis menyadari masih terdapat banyak kekurangan baik dalam pelaksanaan kerja praktik maupun penyusunan buku laporan ini, namun kami berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi. Penulis mengharapkan kritik dan saran yang membangun untuk kesempurnaan penulisan buku laporan ini.

Melalui laporan ini penulis juga ingin menyampaikan rasa terima kasih kepada kepada orang-orang yang telah membantu dalam pelaksanaan kerja praktik hingga penyusunan laporan Kerja praktik baik secara langsung maupun tidak langsung. Orang-orang tersebut antara lain adalah:

1. Orang tua penulis,
2. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D. selaku dosen pembimbing kerja praktik yang telah membimbing penulis selama kerja praktik berlangsung.
3. Bapak Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D selaku koordinator Kerja Praktik.
4. Bapak Dr. I Ketut Eddy Purnama, S.T., M.T. selaku pembimbing lapangan selama kerja praktik yang telah memberikan bimbingan serta ilmunya kepada penulis.

Surabaya, Januari 2021

Penulis

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dari waktu ke waktu, setiap kegiatan yang dilaksanakan di Fakultas Teknologi Elektro dan Informatika Cerdas tidak tercatat dengan baik. Akibatnya, pencarian kegiatan-kegiatan tersebut sulit dilakukan di waktu mendatang.

Masalah tersebut memunculkan ide untuk membangun sebuah aplikasi untuk melakukan pengawasan terhadap seluruh catatan yang dapat disajikan secara terperinci dan terstruktur.

Aplikasi ini akan mengelompokkan kegiatan berdasarkan kategorinya dan untuk setiap kategori dapat diurutkan berdasarkan tahun, departemen, dan detil kegiatan lainnya.

1.2. Tujuan

Tujuan Kerja praktik kali ini adalah melakukan implementasi aplikasi monitoring kinerja FTEIC ITS sekaligus menuntaskan kewajiban kuliah kerja praktik di Institut Teknologi Sepuluh Nopember.

1.3. Manfaat

Manfaat dari implementasi aplikasi ini adalah memudahkan pencarian dan penyortiran catatan kegiatan yang ada pada FTEIC ITS yang akan berguna di waktu yang akan datang.

1.4. Rumusan Permasalahan

Berikut rumusan masalah dalam pelaksanaan kerja praktik implementasi aplikasi monitoring kinerja FTEIC ITS berbasis Android:

- Bagaimana membangun aplikasi monitoring kinerja FTEIC ITS berbasis android?
- Bagaimana mengelola aplikasi monitoring kinerja FTEIC ITS berbasis android agar lebih fleksibel dan optimal?

1.5. Lokasi dan Waktu Kerja Praktik

Kerja praktik kali ini dilaksanakan pada waktu dan tempat sebagai berikut:

Lokasi	: *Online*
Alamat	: Perumdos Blok U no. 169, Kampus Sukolilo ITS Surabaya
Waktu	: 1 Oktober 2020 – 31 Desember 2020
Hari Kerja	: Setiap Hari
Jam kerja	: Fleksibel

1.6. Metodologi Kerja Praktik

1.6.1. Perumusan Masalah

Untuk mengetahui domain dan fungsionalitas, dijelaskan secara rinci bagaimana sistem yang harus dibuat. Penjelasan oleh pembimbing lapangan kerja praktik kali ini menghasilkan beberapa catatan mengenai gambaran secara garis besar tentang sistem berbasis android yang sebelumnya telah diterapkan. Setelah mendapatkan gambaran sistem, diskusi lebih lanjut dilakukan guna menentukan rancangan serta *tools* pendukung pembuatan sistem.

1.6.2. Studi Literatur

Pada tahap ini, setelah ditentukannya rancangan *database*, bahasa pemrograman sampai dengan teknologi beserta *tools* tambahan yang digunakan, dilakukan studi literatur lanjut mengenai bagaimana penggunaannya dalam membangun sistem sesuai yang diharapkan.
Dikarenakan aplikasi yang akan dibuat merupakan bagian dari sistem yang sudah terbangun, maka secara garis besar tools yang digunakan juga tidak jauh berbeda. Bahasa pemrograman yang digunakan Python untuk *backend* dan *web server*, dengan bantuan kerangka kerja (framework) Django. Kemudian untuk aplikasi android menggunakan Kotlin sebagai bahasa pemrograman.

1.6.3. Analisis dan Perancangan Sistem

Langkah ini meliputi penjelasan awal tentang sistem. Bagaimana cara kerja sistem dengan skenario tertentu. Dari penjelasan awal telah

didapatkan beberapa kebutuhan fungsional secara garis besar. Kemudian dilanjutkan dengan memperjelas dan menspesifikkan kebutuhan- kebutuhan tersebut.

1.6.4. Implementasi Sistem

Implementasi sistem didasarkan oleh perancangan dan analisis sebelumnya. Semua didasari pada rancangan yang sudah ada sebelumnya dan penentuan *tools* yang telah dilakukan sebelumnya. Penentuan tipe data saat pembuatan table baru pada database disesuaikan juga dengan kebutuhan.
Pengerjaan dilakukan dengan progres setiap hari, dengan setiap harinya menargetkan perkembangan dari hari sebelumnya. Progres penyelesaian aplikasi terus dipantau oleh Pembimbing Lapangan.

1.6.5. Pengujian dan Evaluasi

Pengujian dilakukan oleh pembimbing lapangan setiap bagian dari fitur telah selesai dikerjakan untuk memberikan evaluasi ketika ada yang tidak sesuai, dan persetujuan apabila sudah sesuai.

1.7. Sistematika Laporan

Laporan kerja praktik ini terdiri dari 7 bab dengan rincian sebagai berikut:

1.7.1. Bab I: Pendahuluan

Bab ini berisi tentang latar belakan masalah, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

1.7.2. Bab II: Profil Instansi

Bab ini berisi sekilas tentang profil Fakultas Teknologi Elektro dan Informatika Cerdas.

1.7.3. Bab III: Tinjauan Pustaka

Dalam bab ini dibahas mengenai konsep-konsep pembuatan aplikasi, dasar teori, teknologi yang dipakai dalam pembuatan aplikasi.

1.7.4. Bab IV: Implementasi Sistem

Dalam bab ini dibahas tentang lapisan antarmuka, lapisan kontrol, lapisan data, dan antarmuka pengguna.

1.7.5. Bab V: Pengujian dan Evaluasi

Dalam bab ini dibahas tentang lapisan antarmuka, lapisan kontrol, lapisan data, dan antarmuka pengguna.

1.7.6. Bab VI: Kesimpulan dan Saran

Bab ini berisi tentang kesimpulan dan saran yang didapatkan dari tugas selama kerja praktik.

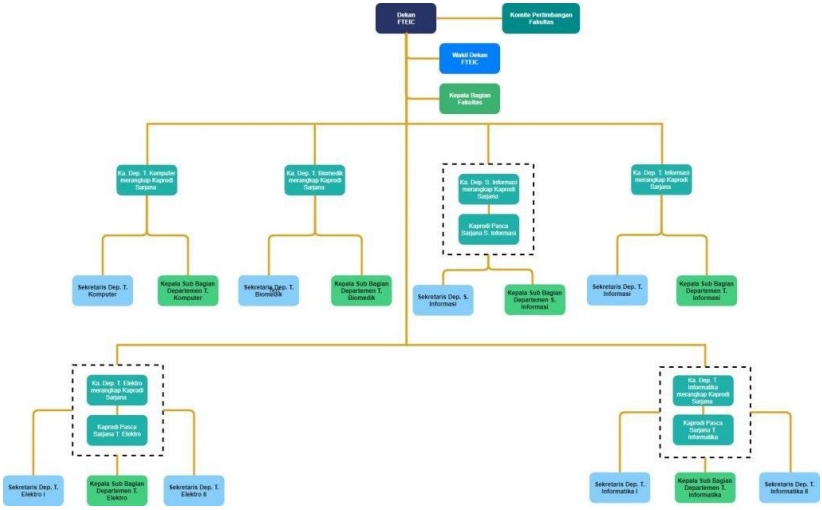
BAB II
PROFIL INSTANSI

2.1. Profil Instansi

Fakultas Teknologi Elektro dan Informatika Cerdas (F-Electics) adalah fakultas unggulan di ITS yang didirikan sebagai gabungan dari Fakultas Teknologi Informasi dan Komunikasi (FTIK) dan Fakultas Teknologi Elektro (FTE) yang terdiri dari 6 Departemen dengan lebih dari 2500 mahasiswa baik di tingkat S1, S2 ataupun S3.F-Electics hadir dengan visi membentuk sumber daya manusia berkarakter, berbudi unggul serta berkelas dunia dalam bidang teknologi elektro, sistem informasi, biomedik, komputer, informatika dan teknologi informasi..

2.2. Struktur Organisasi

Struktur organisasi di FTEIC dapat dilihat pada Gambar 2.1.



Gambar 2.1: Struktur organisasi FTEIC

2.3. Lokasi Instansi

Penampakan depan salah satu departemen yang ada pada FTEIC ITS dapat dilihat pada Gambar 2.2.

Alamat : Jl. Teknik Kimia - Gedung Departemen Teknik Informatika
Kampus Institut Teknologi Sepuluh Nopember Surabaya
Jalan Raya ITS, Sukolilo, Surabaya 60111, Indonesia

Gambar 2.2: Foto Departemen Informatika FTEIC ITS

BAB III
TINJAUAN PUSTAKA

3.1. Android

Merupakan sistem operasi berbasis Linux yang dirancang untuk perangkat bergerak layar sentuh seperti smartphone dan komputer tablet.

3.2. Python

Python adalah Bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Python juga didukung oleh komunitas yang besar.

Python mendukung multi paradigma pemrograman, utamanya; namun tidak dibatasi; pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori otomatis. Seperti halnya pada bahasa pemrograman dinamis lainnya, python umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa skrip. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi.

3.3. Django

Django adalah web framework Python yang didesain untuk membuat aplikasi web yang dinamis, kaya fitur dan aman. Django yang dikembangkan oleh Django Software Foundation terus mendapatkan perbaikan sehingga membuat web framework yang satu ini menjadi pilihan utama bagi banyak pengembang aplikasi web.

3.4. JavaScript

JavaScript adalah sebuah script yang memungkinkan kita mengimplementasikan hal-hal kompleks pada halaman web terutama yang bersifat interaktif. Javascript juga dapat digunakan untuk memanipulasi dan mengirim data pada browser pengguna.

3.5. HTML

Hyper Text Markup Language (HTML) adalah sebuah bahasa markah yang digunakan untuk membuat sebuah halaman web, menampilkan berbagai informasi di dalam sebuah penjelajah web Internet dan pemformatan hiperteks sederhana yang ditulis dalam berkas format ASCII agar dapat menghasilkan tampilan wujud yang terintegerasi.

3.6. Kotlin

Kotlin merupakan bahasa pemrograman modern, disajikan secara statis yang berjalan pada platform Java Virtual Machine (JVM). Kotlin menggunakan compiler LLVM yang artinya, dapat dikompilasi ke dalam kode JavaScript. Bahasa pemrograman yang satu ini banyak diminati oleh para developer dikarenakan mudah untuk dipelajari dan bisa *cross-platform*.

3.7. Web Server

Web Server adalah sebuah perangkat lunak server yang berfungsi menerima permintaan HTTP atau HTTPS dari klien yang dikenal dengan *web browser* dan mengirimkan kembali hasilnya dalam halaman-halaman web yang umumnya berbentuk dokumen HTML.

3.8. Android Studio

Meupakan Integrated Development Environment (IDE) yang dibangun di atas perangkat lunak JetBrains IntelliJ IDEA dan didesain khusus untuk pengembangan Android. Juga terdapat fitur version control menggunakan Git yang memudahkan untuk bekerja dengan tim.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI SISTEM

Bab ini membahas tentang implementasi dari perancangan fitur dan pengaplikasian fitur.

4.1 Implementasi Sistem

Fitur yang dibuat merupakan bahasa kotlin dimana didalamnya terdapat activity yang bertugas untuk menampilkan layout, lalu adapter untuk menampilkan data dalam bentuk list, dan model sebagai representasi data dari suatu modul.

Aplikasi melakukan konsumsi *Application Interface Program* untuk menerima data dari sistem basis data menggunakan Retrofit yang merupakan library tambahan untuk merubah *API interfaces* menjadi *callable objects*.

4.2 Implementasi Tampilan Halaman

4.2.1. Menampilkan Halaman Login

Halaman login adalah halaman default untuk aplikasi ini. Pengguna dapat login menggunakan username dan password field yang disediakan. Kode Sumber 4.1 Menampilkan dependensi dan Kode Sumber 4.2 Menampilkan kode sumber untuk tampilan login.

```
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.view.View
import android.view.inputmethod.InputMethodManager
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.google.android.material.snackbar.Snackbar
import com.kpbois.fteicmobile.retrofit.ApiService
import com.kpbois.fteicmobile.retrofit.LoginResponse
import kotlinx.android.synthetic.main.activity_login.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
```

Kode Sumber 4.1: Dependensi Halaman Login

```
class LoginActivity : AppCompatActivity(),
View.OnClickListener {

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)
        bt_login.setOnClickListener(this)
        if (Helper.getToken(this)!!.isNotEmpty()) {
            val intent = Intent(this,
HomeActivity::class.java)
            startActivity(intent)
            finish()
        }
    }

    override fun onClick(p0: View?) {
        when (p0?.id) {
            R.id.bt_login -> {
                val imm =
getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

imm.hideSoftInputFromWindow(currentFocus?.windowToke
n, 0)
                login()
            }
        }
    }

    private fun login() {
        if (et_username.text.isNullOrBlank()) {
            Toast.makeText(this, "Username tidak
boleh kosong.", Toast.LENGTH_SHORT).show()
            return
        }
        if (et_password.text.isNullOrBlank()) {
            Toast.makeText(this, "Password tidak
boleh kosong.", Toast.LENGTH_SHORT).show()
            return
        }
        loading(true)
        ApiService.endpoint.login(
            et_username.text.toString(),
            et_password.text.toString()
        ).enqueue(object : Callback<LoginResponse>{
            override fun onFailure(call:
Call<LoginResponse>, t: Throwable) {
t.printStackTrace()
```

```
                t.printStackTrace()

Snackbar.make(findViewById(android.R.id.content),
t.localizedMessage, Snackbar.LENGTH_SHORT).show()
                loading(false)
            }

            override fun onResponse(call:
Call<LoginResponse>, response:
Response<LoginResponse>) {
                loading(false)
                when (response.code()) {
                    in 200..299 -> {

Helper.setToken(this@LoginActivity,
response.body()?.token)
                        val intent =
Intent(this@LoginActivity, HomeActivity::class.java)
                        startActivity(intent)
                        finish()
                    }
                    in 400..499 -> {

Snackbar.make(findViewById(android.R.id.content),
response.errorBody()!!.string(),
Snackbar.LENGTH_SHORT).show()
                    }
                    in 500..599 -> {

Snackbar.make(findViewById(android.R.id.content),
"Something went wrong.",
Snackbar.LENGTH_SHORT).show()
                    }
                }
            }
        })
    }
```

```
    private fun loading(state: Boolean) {
        if (state) {
            pb_loading.visibility = View.VISIBLE
            bt_login.visibility = View.GONE
        } else {
            pb_loading.visibility = View.GONE
            bt_login.visibility = View.VISIBLE
        }
    }
}
```

Kode Sumber 4.2: Kode Sumber Halaman Login

4.2.2. Menampilkan Halaman Dashboard

Setelah login, pengguna akan diarahkan ke halaman dashboard. Halaman ini berisi kategori jenis kegiatan yaitu prestasi, sertifikasi, training, journal / konferensi, kuliah tamu. Dependensi dan kode sumber untuk halaman dashboard dapat dilihat pada Kode Suimber 4.3 dan 4.4.

```
import android.content.Intent
import android.os.Build
import android.os.Bundle
import android.view.*
import android.widget.ImageView
import android.widget.Space
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.cardview.widget.CardView
import
androidx.constraintlayout.widget.ConstraintLayout
import
com.kpbois.fteicmobile.konferensiandjurnal.ViewKonju
r
import com.kpbois.fteicmobile.kultam.ViewKultam
import com.kpbois.fteicmobile.prestasi.ViewPrestasi
import com.kpbois.fteicmobile.retrofit.ApiService
import
com.kpbois.fteicmobile.retrofit.DepartemenModel
import
com.kpbois.fteicmobile.submission.ViewSubmission
import com.kpbois.fteicmobile.training.ViewTraining
import
kotlinx.android.synthetic.main.activity_home2.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
```

Kode Sumber 4.3: Dependensi Halaman Dashboard

```
class HomeActivity : AppCompatActivity(),
View.OnClickListener {

    private var validationPage: MenuItem? = null
    private var isAdmin = false

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_home2)
        setSupportActionBar(toolbar)
        supportActionBar?.elevation = 0f

supportActionBar?.setDisplayShowTitleEnabled(false)

        if (Helper.getToken(this).isNullOrEmpty()) {
            val intent = Intent(this,
LoginActivity::class.java)
            startActivity(intent)
            finish()
        }

        home_page.afterMeasured {
            nameLayout(home_page, relativeLayout4)
        }

        relativeLayout4.afterMeasured {
            nameView(relativeLayout4, space,
imageView4)
        }

        cv_training.afterMeasured {
            changeImageSize(cv_training, imageView9)
            changeImageSize(cv_training,
imageView11)
            changeImageSize(cv_training, imageView8)
            changeImageSize(cv_training,
imageView10)

            dynamicMargin(cv_training, textView8)
            dynamicMargin(cv_training, textView9)
            dynamicMargin(cv_training, textView10)
            dynamicMargin(cv_training, textView11)
        }

        getUserInfo()
        getDepartemen()
    }
```

```
override fun onClick(p0: View?) {
        when (p0?.id) {
            R.id.cv_kultam -> {
                val intent = Intent(this,
ViewKultam::class.java)
                startActivity(intent)
            }
            R.id.cv_konjur -> {
                val intent = Intent(this,
ViewKonjur::class.java)
                startActivity(intent)
            }
            R.id.cv_prestasi -> {
                val intent = Intent(this,
ViewPrestasi::class.java)
                startActivity(intent)
//
overridePendingTransition(android.R.anim.fade_in,
android.R.anim.fade_out)
            }
            R.id.cv_training -> {
                val intent = Intent(this,
ViewTraining::class.java)
                startActivity(intent)
            }
        }
    }

    override fun onPrepareOptionsMenu(menu: Menu?):
Boolean {
        validationPage =
menu?.findItem(R.id.action_show_validation)
        validationPage?.isVisible = isAdmin
        return super.onPrepareOptionsMenu(menu)
    }

    override fun onCreateOptionsMenu(menu: Menu?):
Boolean {
        menuInflater.inflate(R.menu.home_menu, menu)
        return true
    }

    override fun onOptionsItemSelected(item:
MenuItem): Boolean {
        return when (item.itemId) {
```

```
return when (item.itemId) {
        R.id.menu_password -> {
            // intent ubah password
            true
        }
        R.id.action_show_validation -> {
            val intent = Intent(this,
ViewSubmission::class.java)
            startActivity(intent)
            true
        }
        R.id.action_logout -> {
            Helper.setToken(this, "")
            Helper.setUserInfo(null)
            startActivity(Intent(this,
LoginActivity::class.java))
            finish()
            true
        }
        else -> super.onOptionsItemSelected(item)
    }
}

inline fun View.afterMeasured(crossinline f: View.()
-> Unit) {

viewTreeObserver.addOnGlobalLayoutListener(object :
ViewTreeObserver.OnGlobalLayoutListener{
        override fun onGlobalLayout() {
            if (measuredHeight > 0 && measuredWidth
> 0) {
                if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.JELLY_BEAN) {

this@afterMeasured.viewTreeObserver.removeOnGlobalLa
youtListener(this)
                }else{

this@afterMeasured.viewTreeObserver.removeGlobalOnLa
youtListener(this)
                }
                f()
            }
        }
    })
}
```

```
private fun changeImageSize(cardView: CardView,
imageView: ImageView){
    val params : ViewGroup.LayoutParams =
imageView.layoutParams
    params.width = cardView.width / 2
    params.height = cardView.height / 2
    imageView.layoutParams = params
}

private fun nameLayout(base: ConstraintLayout,
target: ConstraintLayout) {
    val params : ViewGroup.LayoutParams =
target.layoutParams
    params.height = base.height / 4
    target.layoutParams = params
}

private fun nameView(constraintLayout:
ConstraintLayout, space: Space, imageView:
ImageView){
    val paramsSpace : ViewGroup.LayoutParams =
space.layoutParams
    val paramsImage : ViewGroup.LayoutParams =
imageView.layoutParams
    val width = constraintLayout.width / 2 -
constraintLayout.width / 30
    paramsSpace.width = width
    paramsImage.width = width +
constraintLayout.width / 10
    paramsImage.height = width +
constraintLayout.width / 10
    space.layoutParams = paramsSpace
    imageView.layoutParams = paramsImage
}

private fun dynamicMargin(cardView: CardView,
textView: TextView) {
    val params = textView.layoutParams as
ViewGroup.MarginLayoutParams
    params.setMargins(0, cardView.height/10,0,0)
    textView.layoutParams = params
}
```

```
private fun getUserInfo() {
    val token = "Token " + Helper.getToken(this)

ApiService.endpoint.getUserInfo(token).enqueue(objec
t : Callback<AccountModel> {
        override fun onResponse(call:
Call<AccountModel>, response:
Response<AccountModel>) {
            if (response.isSuccessful &&
response.body() != null) {
                val userInfo = response.body()!!
                tv_nama.text = userInfo.nama
                tv_departemen.text =
userInfo.departemen
                isAdmin = userInfo.isAdmin
                Helper.setUserInfo(userInfo)
            } else {
                if (response.code() == 401) {

startActivity(Intent(this@HomeActivity,
LoginActivity::class.java))
                    finish()
                }
                Toast.makeText(
                    this@HomeActivity,
                    "Failed: ${response.message()}",
                    Toast.LENGTH_SHORT
                ).show()
            }
        }

        override fun onFailure(call:
Call<AccountModel>, t: Throwable) {
            Toast.makeText(
                this@HomeActivity,
                "Failed: ${t.localizedMessage}",
                Toast.LENGTH_SHORT
            ).show()
        }
    })
}
```

```
private fun getDepartemen() {

ApiService.endpoint.getDepartemen().enqueue(object:
Callback<List<DepartemenModel>>{
            override fun onResponse(call:
Call<List<DepartemenModel>>, response:
Response<List<DepartemenModel>>) {
                if (response.isSuccessful &&
response.body() != null) {
                    val result = response.body()!!
                    val departemen =
ArrayList<String>()
                    for (item in result){
                        departemen.add(item.nama)
                    }
                    Helper.setDepartemen(departemen)
                } else {
                    if (response.code() == 401) {

startActivity(Intent(this@HomeActivity,
LoginActivity::class.java))
                        finish()
                    }
                    Toast.makeText(
                        this@HomeActivity,
                        "Failed:
${response.message()}",
                        Toast.LENGTH_SHORT
                    ).show()
                }
            }

            override fun onFailure(call:
Call<List<DepartemenModel>>, t: Throwable) {
                Toast.makeText(
                    this@HomeActivity,
                    "Failed: ${t.localizedMessage}",
                    Toast.LENGTH_SHORT
                ).show()
            }

        })
    }
}
```

Kode Sumber 4.4: Kode Sumber Halaman Dashboard

4.2.3. Menampilkan Halaman Kuliah Tamu

Halaman kuliah tamu berisi daftar kuliah tamu yang tercatat. Field yang ditampilkan adalah nama, pemateri, tanggal, departemen dan tingkat. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.5 dan 4.6.

```
import android.app.Activity
import android.content.Context
import android.content.Intent
import android.os.Build
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.view.View
import android.view.inputmethod.InputMethodManager
import android.widget.SearchView
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.isVisible
import
androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.android.material.snackbar.Snackbar
import com.kpbois.fteicmobile.Helper
import com.kpbois.fteicmobile.LoginActivity
import com.kpbois.fteicmobile.R
import com.kpbois.fteicmobile.retrofit.ApiService
import
kotlinx.android.synthetic.main.activity_view2.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
```

Kode Sumber 4.5: Dependensi Halaman Kuliah Tamu

```
class ViewKultam : AppCompatActivity(),
View.OnClickListener {
    companion object {
        const val FILTER_CODE = 100
        const val ADD_KULTAM = 101
        const val KULTAM_DETAIL = 102
    }
    private lateinit var kultamAdapter:
AdapterKultam
    private var modelFilter = ModelFilter()
    private val listInstitusi =
arrayListOf<String>()
    private val listTingkat = arrayListOf<String>()

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_view2)
        setSupportActionBar(toolbar2)
        supportActionBar?.apply {
            setDisplayHomeAsUpEnabled(true)
            title = "Kuliah Tamu"
        }
        fab_add.setOnClickListener(this)
        sw_list.setOnRefreshListener
{ filterData() }
        prepareRecyclerView()
    }

    override fun onCreateOptionsMenu(menu: Menu?):
Boolean {
        menuInflater.inflate(R.menu.view_menu, menu)

        val searchItem =
menu?.findItem(R.id.action_search)
        val searchView = searchItem?.actionView as
SearchView
        searchView.apply {
            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.Q) {
                isIconifiedByDefault = false
                isIconified = false
            }
            setOnQueryTextListener(object :
SearchView.OnQueryTextListener {
                override fun onQueryTextSubmit(p0:
String?): Boolean {
                    kultamAdapter.filter.filter(p0)
                    return true
```

```
                return true
            }

            override fun onQueryTextChange(p0:
String?): Boolean {
                kultamAdapter.filter.filter(p0)
                return true
            }
        })
    }

    searchItem.setOnActionExpandListener(object :
MenuItem.OnActionExpandListener {
            override fun
onMenuItemActionCollapse(item: MenuItem): Boolean {
                val imm =
getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

imm.hideSoftInputFromWindow(currentFocus?.windowToke
n, 0)
                return true // Return true to
collapse action view
            }

            override fun
onMenuItemActionExpand(item: MenuItem): Boolean {
                searchView.requestFocusFromTouch()
                val imm =
getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

imm.toggleSoftInput(InputMethodManager.SHOW_IMPLICIT
, 0)
                return true // Return true to expand
action view
            }
        }
    )

    return super.onCreateOptionsMenu(menu)
}

override fun onOptionsItemSelected(item: MenuItem):
Boolean {
    return when (item.itemId) {
        R.id.action_filter -> {
            val intent = Intent(this,
```

```
val intent = Intent(this, FilterKultam::class.java)
            intent.putExtra("model_filter",
modelFilter)
            intent.putExtra("list_institusi",
listInstitusi)
            intent.putExtra("list_tingkat",
listTingkat)
            startActivityForResult(intent,
FILTER_CODE)
            true
        }
        else -> super.onOptionsItemSelected(item)
    }
}

override fun onActivityResult(requestCode: Int,
resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode,
data)
    if (resultCode == Activity.RESULT_OK) {
        when (requestCode) {
            FILTER_CODE -> {
                if (data != null) {
                    modelFilter =
data.getParcelableExtra("model_filter")
                    filterData()
                }
            }
            ADD_KULTAM, KULTAM_DETAIL ->
filterData()
        }
    }
}

override fun onClick(p0: View?) {
    when (p0?.id) {
        fab_add.id -> {
            val intent = Intent(this,
AddKultam::class.java)
            startActivityForResult(intent,
ADD_KULTAM)
        }
    }
}
```

```
private fun prepareRecyclerView() {
        kultamAdapter = AdapterKultam().apply {
            onClickListener = object :
AdapterKultam.OnClickListener {
                override fun onClick(data:
ModelKultam) {
                    val intent =
Intent(this@ViewKultam, DetailKultam::class.java)

intent.putExtra(DetailKultam.KULTAM_DETAIL, data)
                    startActivityForResult(intent,
KULTAM_DETAIL)
                }
            }
        }
        rv_list.apply {
            layoutManager =
LinearLayoutManager(this@ViewKultam,
RecyclerView.VERTICAL, false)
            adapter = kultamAdapter
        }
        filterData()
    }
    private fun filterData() {
        sw_list.isRefreshing = false
        rv_list.isVisible = false
        progressBar.isVisible = true
        val token = "Token " + Helper.getToken(this)
        var institusi = ""
        var tingkat = ""
        modelFilter.institusi.forEach { institusi =
"$it,$institusi" }
        modelFilter.tingkat.forEach { tingkat =
"$it,$tingkat" }
        ApiService.endpoint.getFilterKultam(
            token,
            if (institusi.isNotBlank())
institusi.removeSuffix(",") else null,
            if (tingkat.isNotBlank())
tingkat.removeSuffix(",") else null,
            if
(modelFilter.tanggalMulai.isNotBlank())
modelFilter.tanggalMulai else null,
            if
(modelFilter.tanggalSelesai.isNotBlank())
modelFilter.tanggalSelesai else null
        ).enqueue(object :
Callback<List<ModelKultam>> {
            override fun onResponse(
```

```
override fun onResponse(
                call: Call<List<ModelKultam>>,
                response:
Response<List<ModelKultam>>
            ) {
                progressBar.isVisible = false
                if (response.isSuccessful &&
response.body() != null) {
                    val list = response.body()!!
                    list.filter { it.isValidated }

kultamAdapter.setData(ArrayList(list))
                    list.forEach {
                        if
(!listInstitusi.contains(it.institusi))
listInstitusi.add(it.institusi)
                        if
(!listTingkat.contains(it.tingkat))
listTingkat.add(it.tingkat)
                    }
                    if (list.isEmpty()) {
                        tv_notfound.isVisible = true
                        rv_list.isVisible = false
                    } else {
                        tv_notfound.isVisible =
false
                        rv_list.isVisible = true
                    }
                } else {
                    if (response.code() == 401) {

startActivity(Intent(this@ViewKultam,
LoginActivity::class.java))
                        finish()
                    }

Snackbar.make(findViewById(android.R.id.content),
"Something went wrong.",
Snackbar.LENGTH_SHORT).show()
                }
            }
```

```
override fun onFailure(call:
Call<List<ModelKultam>>, t: Throwable) {
                progressBar.isVisible = false
                t.printStackTrace()

Snackbar.make(findViewById(android.R.id.content),
t.localizedMessage, Snackbar.LENGTH_SHORT).show()
            }
        })
    }
}
```

Kode Sumber 4.6: Kode Sumber Halaman Kuliah Tamu

4.2.4. Menampilkan Halaman Konferensi / Jurnal

Halaman konferensi berisi daftar konferensi / jurnal yang tercatat. Field yang ditampilkan adalah nama, pelaku, dan tahun. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.7 dan 4.8.

```
import android.app.Activity
import android.content.Context
import android.content.Intent
import android.os.Build
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
```

```
import android.view.Menu
import android.view.MenuItem
import android.view.View
import android.view.inputmethod.InputMethodManager
import android.widget.SearchView
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.Observer
import
androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.kpbois.fteicmobile.Helper
import com.kpbois.fteicmobile.R
import
com.kpbois.fteicmobile.download.DownloadActivity
import com.kpbois.fteicmobile.retrofit.ApiService
import
com.kpbois.fteicmobile.retrofit.KonferensiResponse
import
kotlinx.android.synthetic.main.activity_view2.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
```

Kode Sumber 4.7: Dependensi Halaman Konferensi / Jurnal

```
class ViewKonjur : AppCompatActivity(),
View.OnClickListener {

private lateinit var recyclerView: RecyclerView
    private val listKonferensi =
arrayListOf<ModelKonferensi>()
    private val listJurnal =
arrayListOf<ModelJurnal>()
    private val listTempat = arrayListOf<String>()
    private val state: MutableLiveData<String> =
MutableLiveData()
    private lateinit var adapterKonferensi:
AdapterKonferensi
    private lateinit var adapterJurnal:
AdapterJurnal
    private var modelFilterKonferensi =
ModelFilterKonferensi(tahun = null, scopus = false)
    private var modelFilterJurnal =
ModelFilterJurnal(tahun = null, scopus = false)
    private var isAdmin = false
    companion object {
        const val DETAIL = 40
        const val ADD = 41
        const val FILTER_KONFERENSI = 42
        const val FILTER_JURNAL = 43
        const val FILTER_DEPARTEMEN = "departemen"
        const val FILTER_TINGKAT = "tingkat"
        const val FILTER_TEMPAT = "tempat"
        const val FILTER_TAHUN = "tahun"
        const val FILTER_SCOPUS = "scopus"
    }
```

```
    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
          setContentView(R.layout.activity_view2)
        setSupportActionBar(toolbar2)
        supportActionBar?.apply {

        setDisplayHomeAsUpEnabled(true)
             setDisplayShowHomeEnabled(true)
}

         val userInfo = Helper.getUserInfo()
         isAdmin = userInfo?.isAdmin as Boolean
         configuration()
    }

  private fun changeAdapterJurnal(string: String?) {
        adapterJurnal = AdapterJurnal()
        adapterJurnal.apply {
             setOnClickListener(object :
AdapterJurnal.OnClickListener {
         override fun onClick(data:    ModelJurnal) {
            val intent =   Intent(this@ViewKonjur,
DetailKonjur::class.java)
            intent.apply {

putExtra(DetailKonjur.KONJUR_DETAIL, data)

intent.putExtra(DetailKonjur.STATE, string)
}
```

```
private fun changeAdapterKonferensi(string: String?)
{
        adapterKonferensi = AdapterKonferensi()
        adapterKonferensi.apply {
            setOnClickListener(object :
AdapterKonferensi.OnClickListener {
                override fun onClick(data:
ModelKonferensi) {
                    val intent =
Intent(this@ViewKonjur, DetailKonjur::class.java)
                    intent.apply {

putExtra(DetailKonjur.KONJUR_DETAIL, data)
                        putExtra(DetailKonjur.STATE,
string)
                    }
                    startActivityForResult(intent,
DETAIL)
                }
            })
            setData(listKonferensi)
        }
        recyclerView.adapter = adapterKonferensi
    }
```

```
private fun configuration() {
        val scrollListener = object :
RecyclerView.OnScrollListener() {
            override fun
onScrollStateChanged(recyclerView: RecyclerView,
newState: Int) {

super.onScrollStateChanged(recyclerView, newState)
                if (newState ==
RecyclerView.SCROLL_STATE_IDLE) {
                    fab_add.show()
                }
                else if (newState ==
RecyclerView.SCROLL_STATE_DRAGGING) {
                    fab_add.hide()
                }
            }
        }
        recyclerView = rv_list
        recyclerView.apply {
            setHasFixedSize(true)
            layoutManager =
LinearLayoutManager(this@ViewKonjur)
            addOnScrollListener(scrollListener)
        }

        fab_add.setOnClickListener(this)

        state.apply {
            observe(this@ViewKonjur, Observer {
string ->
                if
(string.equals("konferensi",true)) {
                    getKonferensi(string)
                }
                else if
(string.equals("jurnal",true)) {
```

```
                    getKonferensi(string)
                }
                else if
(string.equals("jurnal",true)) {
                    getJurnal(string)
                }
            })
            value = "Konferensi"
        }

        sw_list.setOnRefreshListener {
            if
(state.value!!.equals("konferensi",true)) {
                getKonferensi("Konferensi")
                sw_list.isRefreshing = false
                modelFilterKonferensi =
ModelFilterKonferensi(tahun = null, scopus = false)
            }
            else if
(state.value!!.equals("jurnal",true)) {
                getJurnal("Jurnal")
                sw_list.isRefreshing = false
                modelFilterJurnal =
ModelFilterJurnal(tahun = null, scopus = false)
            }
        }
    }
```

```
private fun getKonferensi(string: String) {
        supportActionBar?.title = string
        progressBar.visibility = View.VISIBLE
        listKonferensi.clear()
        val token = "Token " + Helper.getToken(this)
        ApiService.endpoint.getKonferensi(token)
            .enqueue(object :
Callback<List<KonferensiResponse>>{
                override fun onFailure(call:
Call<List<KonferensiResponse>>, t: Throwable) {
                    progressBar.visibility =
View.GONE
                }

override fun onResponse(call:
Call<List<KonferensiResponse>>, response:
Response<List<KonferensiResponse>>) {
     progressBar.visibility = View.GONE
     val resp = response.body()!!
     for (i in resp) {
        if (isAdmin) {
          val model = ModelKonferensi(
             id = i.id,
             judul = i.judul,
             author = i.author,
             departemen = i.departemen,
             published_at = i.published_at,
             url = i.url,
             tahun = i.tahun,
             tingkat = i.tingkat,
             pi = i.pi,
             pn = i.pn,
             konf_hal = i.konf_hal,
             tempat = i.tempat,
```

```
                tanggal_mulai = i.tanggal_mulai,
                tanggal_selesai = i.tanggal_selesai,
                scopus = i.scopus,
                isValidated = i.is_validated
                    )

listKonferensi.add(model)
        }

        else {
          if (i.is_validated.toBoolean()){
            val model = ModelKonferensi(
              id = i.id,
              judul = i.judul,
              author = i.author,
              departemen = i.departemen,
              published_at = i.published_at,
              url = i.url,
              tahun = i.tahun,
              tingkat = i.tingkat,
              pi = i.pi,
              pn = i.pn,
              konf_hal = i.konf_hal,
              tempat = i.tempat,
              tanggal_mulai = i.tanggal_mulai,
              tanggal_selesai = i.tanggal_selesai,
              scopus = i.scopus,
              isValidated = i.is_validated
            )
            listKonferensi.add(model)
        }
      }
      if (!listTempat.contains(i.tempat))
listTempat.add(i.tempat)
      }
```

```
      changeAdapterKonferensi(string)
    }
   })
 }

  private fun getJurnal(string: String) {
    supportActionBar?.title = string
    progressBar.visibility = View.VISIBLE
    listJurnal.clear()
    val token = "Token " + Helper.getToken(this)
    ApiService.endpoint.getJurnal(
token).enqueue(object: Callback<List<
ModelJurnal>>{
      override fun onFailure(call:
Call<List<ModelJurnal>>, t: Throwable) {
         progressBar.visibility = View.GONE
            }

      override fun onResponse(call:
Call<List<ModelJurnal>>, response:
Response<List<ModelJurnal>>) {
         progressBar.visibility = View.GONE
         val resp = response.body()!!
         if (isAdmin) listJurnal.addAll(resp)
         else {
           for (item in resp) {
             if (item.is_validated.
toBoolean()) listJurnal.add(item)
             }
         }
         changeAdapterJurnal(string)
      }

    })
  }
```

```
override fun onCreateOptionsMenu(menu: Menu?):
Boolean {
        menuInflater.inflate(R.menu.konjur_menu,
menu)

        val searchItem =
menu?.findItem(R.id.action_searchKonjur)
        val searchView = searchItem?.actionView as
SearchView
        searchView.apply {
            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.Q) {
                searchView.isIconifiedByDefault =
false
                searchView.isIconified = false
            }
            setOnQueryTextListener(object :
SearchView.OnQueryTextListener{
                override fun
onQueryTextSubmit(query: String?): Boolean {
                    if
(state.value!!.equals("konferensi",true))
adapterKonferensi.filter.filter(query)
                    else if
(state.value!!.equals("Jurnal", true))
adapterJurnal.filter.filter(query)
                    return true
                }

                override fun
onQueryTextChange(newText: String?): Boolean {
                    if
(state.value!!.equals("konferensi",true))
adapterKonferensi.filter.filter(query)
```

```
else if (state.value!!.equals("Jurnal", true))
adapterJurnal.filter.filter(query)
                    return true
                }
            })
        }

searchItem.setOnActionExpandListener(object :
MenuItem.OnActionExpandListener {
            override fun
onMenuItemActionExpand(item: MenuItem?): Boolean {
                searchView.requestFocusFromTouch()
                val imm =
getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

imm.toggleSoftInput(InputMethodManager.SHOW_IMPLICIT
, 0)
                return true
            }

            override fun
onMenuItemActionCollapse(item: MenuItem?): Boolean {
                val imm =
getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

imm.hideSoftInputFromWindow(currentFocus?.windowToke
n, 0)
                return true
            }
        })
        return super.onCreateOptionsMenu(menu)
    }
```

```
override fun onOptionsItemSelected(item: MenuItem):
Boolean {
        when(item.itemId){
            R.id.action_konferensi -> state.value =
"Konferensi"
            R.id.action_jurnal -> state.value =
"Jurnal"
            R.id.action_filterKonjur -> {
                if
(state.value!!.equals("konferensi", true)) {
                    val intent = Intent(this,
FilterKonferensi::class.java)
                    val tingkat =
getList(R.array.tingkat_konferensi)
                    val tempat = listTempat
                    intent.apply {

putExtra(FilterKonferensi.state, state.value)

putExtra(FilterKonferensi.ITEM_TINGKAT, tingkat)

putExtra(FilterKonferensi.ITEM_TEMPAT, tempat)

putExtra(FilterKonferensi.PREVIOUS_FILTER,
modelFilterKonferensi)
                    }
                    startActivityForResult(intent,
FILTER_KONFERENSI)
                }
                else {
                    val intent = Intent(this,
FilterJurnal::class.java)
```

```
val tingkat = getList(R.array.tingkat_jurnal)
                    intent.apply {
                        putExtra(FilterJurnal.state,
state.value)

putExtra(FilterJurnal.ITEM_TINGKAT, tingkat)

putExtra(FilterJurnal.PREVIOUS_FILTER,
modelFilterJurnal)
                    }
                    startActivityForResult(intent,
FILTER_JURNAL)
                }
            }
            R.id.action_download_konjur -> {
                val intent = Intent(this,
DownloadActivity::class.java)
                startActivityForResult(intent, 500)
            }
        }
        return super.onOptionsItemSelected(item)
    }

private fun getList(resource: Int):
ArrayList<String> {
        val array =
resources.getStringArray(resource)
        val arrayList = ArrayList<String>()
        arrayList.addAll(array)
        return arrayList
    }
```

```
override fun onClick(v: View?) {
        when (v?.id) {
            R.id.fab_add -> {
                val intent = Intent(this,
AddKonjur::class.java)
                intent.putExtra(AddKonjur.STATE,
state.value)
                startActivityForResult(intent, ADD)
            }
        }
    }

private fun getFilterValue(values:
ArrayList<String>?): String? {
        if (values!!.isEmpty()) return null
        var result = ""
        for (item in values) {
            if (result != "") result = "$result,"
            result = "$result$item"
        }
        return result
    }

private fun getDepartemenId(values:
ArrayList<String>?): String? {
        if (values!!.isEmpty()) return null
        val dept = Helper.getAllDepartemen()
        var result = ""
        for (item in values) {
            if (result != "") result = "$result,"
            result =
"$result${dept.indexOf(item)+1}"
        }
        return result
    }
```

```
private fun getScopus(scopus: String?): String? {
        return if (scopus == "false") null
        else scopus
    }

override fun onActivityResult(requestCode: Int,
resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode,
resultCode, data)
        when (requestCode) {
            ADD -> {
                if (resultCode ==
Activity.RESULT_OK) {
                    val result =
data?.getStringExtra("result")
                    state.value = result
                }
                else if (resultCode ==
Activity.RESULT_CANCELED) {
                    val result =
data?.getStringExtra("result")
                    state.value = result
                }
            }
            DETAIL -> {
                if (resultCode ==
Activity.RESULT_OK) {
                    val result =
data?.getStringExtra("result")
                    state.value = result
                }
            }
```

```
FILTER_KONFERENSI -> {
   if (resultCode == Activity.RESULT_OK) {
       listKonferensi.clear()
       adapterKonferensi
.setData(listKonferensi)
        progressBar.visibility = View.VISIBLE
        val resultDepartemen = data?
.getStringArrayListExtra(FILTER_DEPARTEMEN)
        val resultTingkat = data?
.getStringArrayListExtra(FILTER_TINGKAT)
        val resultTempat = data?
.getStringArrayListExtra(FILTER_TEMPAT)
        val resultTahun = data?
.getStringExtra(FILTER_TAHUN)
        val scopus = getScopus(data?
.getStringExtra(FILTER_SCOPUS))
        modelFilterKonferensi =
ModelFilterKonferensi(resultDepartemen as
ArrayList<String>, resultTingkat as
ArrayList<String>, resultTempat as
ArrayList<String>, resultTahun, scopus?.toBoolean())
        val token = "Token " + Helper.getToken(this)
        ApiService.endpoint
.getFilterKonferensi(
          token = token,
          departemen = getDepartemenId(
resultDepartemen),
          tingkat = getFilterValue(
resultTingkat),
          tempat = getFilterValue(
resultTempat),
           tahun = resultTahun?.trim(),
           scopus = scopus
).enqueue(object :
```

```
FILTER_KONFERENSI -> {
   if (resultCode == Activity.RESULT_OK) {
       listKonferensi.clear()
       adapterKonferensi
.setData(listKonferensi)
        progressBar.visibility = View.VISIBLE
        val resultDepartemen = data?
.getStringArrayListExtra(FILTER_DEPARTEMEN)
        val resultTingkat = data?
.getStringArrayListExtra(FILTER_TINGKAT)
        val resultTempat = data?
.getStringArrayListExtra(FILTER_TEMPAT)
        val resultTahun = data?
.getStringExtra(FILTER_TAHUN)
        val scopus = getScopus(data?
.getStringExtra(FILTER_SCOPUS))
        modelFilterKonferensi =
ModelFilterKonferensi(resultDepartemen as
ArrayList<String>, resultTingkat as
ArrayList<String>, resultTempat as
ArrayList<String>, resultTahun, scopus?.toBoolean())
        val token = "Token " + Helper.getToken(this)
        ApiService.endpoint
.getFilterKonferensi(
          token = token,
          departemen = getDepartemenId(
resultDepartemen),
          tingkat = getFilterValue(
resultTingkat),
          tempat = getFilterValue(
resultTempat),
           tahun = resultTahun?.trim(),
           scopus = scopus
).enqueue(object :
```

```
Callback<List<KonferensiResponse>>{
                        override fun onFailure(call:
Call<List<KonferensiResponse>>, t: Throwable) {
                            progressBar.visibility =
View.GONE
}

override fun onResponse(call:
Call<List<KonferensiResponse>>, response:
Response<List<KonferensiResponse>>) {
     progressBar.visibility = View.GONE
     val resp = response.body()!!
     for (i in resp) {
       if (isAdmin) {
         val model = ModelKonferensi(
            i.id,
            i.judul,
            i.author,
            i.departemen,
            i.published_at,
            i.url,
            i.tahun,
            i.tingkat,
            i.pi,
            i.pn,
            i.konf_hal,
            i.tempat,
            i.tanggal_mulai,
            i.tanggal_selesai
         )
         listKonferensi.add(model)
       }
```

```
        else {
          if (i.is_validated.toBoolean()) {
            val model = ModelKonferensi(
              i.id,
              i.judul,
              i.author,
              i.departemen,
              i.published_at,
              i.url,
              i.tahun,
              i.tingkat,
              i.pi,
              i.pn,
              i.konf_hal,
              i.tempat,
              i.tanggal_mulai,
              i.tanggal_selesai
            )
            listKonferensi.add(model)
          }
      }
      if (!listTempat.contains(i.tempat))
listTempat.add(i.tempat)
    }
    changeAdapterKonferensi("Konferensi")
    }
   })
 }
}
```

```
FILTER_JURNAL -> {
                if (resultCode ==
Activity.RESULT_OK) {
                    listJurnal.clear()

adapterJurnal.setData(listJurnal)
                    progressBar.visibility =
View.VISIBLE
                    val departemen =
data?.getStringArrayListExtra(FILTER_DEPARTEMEN)
                    val tingkat =
data?.getStringArrayListExtra(FILTER_TINGKAT)
                    val tahun =
data?.getStringExtra(FILTER_TAHUN)
                    val scopus =
getScopus(data?.getStringExtra(FILTER_SCOPUS))
                    modelFilterJurnal =
ModelFilterJurnal(departemen as ArrayList<String>,
tingkat as ArrayList<String>, tahun,
scopus?.toBoolean())
                    val token = "Token " +
Helper.getToken(this)

ApiService.endpoint.getFilterJurnal(
                        token = token,
                        departemen =
getDepartemenId(departemen),
                        tingkat =
getFilterValue(tingkat),
                        tahun = tahun,
                        scopus = scopus
                    ).enqueue(object:
Callback<List<ModelJurnal>>{
```

```
override fun onFailure(call:
Call<List<ModelJurnal>>, t: Throwable) {
                                progressBar.visibility =
View.GONE
                            }

                            override fun
onResponse(call: Call<List<ModelJurnal>>, response:
Response<List<ModelJurnal>>) {
                                progressBar.visibility =
View.GONE
                                val resp =
response.body()!!
                                if (isAdmin)
listJurnal.addAll(resp)
                                else {
                                    for (item in resp) {
                                        if
(item.is_validated.toBoolean()) listJurnal.add(item)
                                    }
                                }

changeAdapterJurnal("Jurnal")
                            }
                        })
                    }
                }
            }
        }
```

Kode Sumber 4.8: Kode Sumber Halaman Konferensi / Jurnal

4.2.5. Menampilkan Halaman Prestasi

Halaman Prestasi berisi daftar prestasi yang tercatat. Field yang ditampilkan adalah nama, pelaku, tahun, dan capaian. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.9 dan 4.10.

```
import android.app.Activity
import android.content.Context
import android.content.Intent
import android.os.Build
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.view.View
import android.view.inputmethod.InputMethodManager
import android.widget.SearchView
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import
androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.kpbois.fteicmobile.Helper
import com.kpbois.fteicmobile.R
import
com.kpbois.fteicmobile.download.DownloadActivity
import
com.kpbois.fteicmobile.retrofit.PrestasiDosenModel
import
kotlinx.android.synthetic.main.activity_view2.*
```

Kode Sumber 4.9: Dependensi Halaman Prestasi Dosen

```
class ViewPrestasi : AppCompatActivity() {

    companion object {
        const val FILTER_CODE = 200
        const val DETAIL_CODE = 201
        const val ADD_CODE = 205
        const val FILTER_KATEGORI_PESERTA =
"kategori_peserta"
        const val FILTER_KATEGORI_PRESTASI =
"tingkat"
        const val FILTER_CAPAIAN = "peringkat"
        const val FILTER_TANGGAL_MULAI = "mulai"
        const val FILTER_TANGGAL_SELESAI = "selesai"
        const val FILTER_DEPARTEMEN = "departemen"
    }
    private lateinit var recyclerView: RecyclerView
    private val list =
mutableListOf<PrestasiDosenModel>()
    private lateinit var prestasiAdapter:
AdapterPrestasi
    private lateinit var vm: ViewModelPrestasi
    private var modelFilterPrestasi =
ModelFilterPrestasi()
    private var isAdmin = false

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_view2)
        setSupportActionBar(toolbar2)
```

```
        supportActionBar?.apply {
            title = "Prestasi"
            setDisplayHomeAsUpEnabled(true)
        }
        recyclerView = rv_list
        val userInfo = Helper.getUserInfo()
        isAdmin = userInfo?.isAdmin as Boolean

        fabOnClick(this)
        setRecyclerView()

        vm = ViewModelProvider(this,
viewModelFactory { ViewModelPrestasi(this,
progressBar, View.VISIBLE, isAdmin) })
            .get(ViewModelPrestasi::class.java)
        vm.getPrestasi().observe(this, Observer {
prestasiSnapshot ->
            list.clear()
            list.addAll(prestasiSnapshot)
            prestasiAdapter.setData(list)
        })
        vm.getIsRefreshing().observe(this, Observer
{ isRefreshing ->
            sw_list.isRefreshing = isRefreshing
        })
        sw_list.setOnRefreshListener {
            vm.getPrestasi(View.GONE, true)
            modelFilterPrestasi =
ModelFilterPrestasi()
        }
    }
```

```
    protected inline fun <VM : ViewModel>
viewModelFactory(crossinline f: () -> VM) =
        object : ViewModelProvider.Factory {
            override fun <T : ViewModel>
create(aClass: Class<T>):T = f() as T
        }

    private fun setRecyclerView() {
        val scrollListener = object :
RecyclerView.OnScrollListener() {
            override fun
onScrollStateChanged(recyclerView: RecyclerView,
newState: Int) {

super.onScrollStateChanged(recyclerView, newState)
                if
(RecyclerView.SCROLL_STATE_IDLE==newState){
                    fab_add.show()
                }
                else if
(RecyclerView.SCROLL_STATE_DRAGGING == newState) {
                    fab_add.hide()
                }
            }
        }

        prestasiAdapter = AdapterPrestasi()
        prestasiAdapter.setOnClickListener(object:
AdapterPrestasi.OnClickListener{
            override fun onClick(data:
PrestasiDosenModel) {
```

```
                val intent =
Intent(this@ViewPrestasi,
DetailPrestasi::class.java)

intent.putExtra(DetailPrestasi.PRESTASI_DETAIL,
data)
                startActivityForResult(intent,
DETAIL_CODE)
            }

        })
        recyclerView.apply {
            layoutManager =
LinearLayoutManager(this@ViewPrestasi)
            setHasFixedSize(true)
            addOnScrollListener(scrollListener)
            adapter = prestasiAdapter
        }
    }

    private fun fabOnClick(context: Context){
        fab_add.setOnClickListener {
            val intent = Intent(context,
AddPrestasi::class.java)
            startActivityForResult(intent, ADD_CODE)
        }
        fab_add.visibility = View.GONE
    }

    override fun onCreateOptionsMenu(menu: Menu?):
Boolean {
        menuInflater.inflate(R.menu.view_menu, menu)

        val searchItem =
menu?.findItem(R.id.action_search)
```

```
        val searchView = searchItem?.actionView as
SearchView
        searchView.apply {
            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.Q) {
                searchView.isIconifiedByDefault =
false
                searchView.isIconified = false
            }
            setOnQueryTextListener(object :
SearchView.OnQueryTextListener{
                override fun
onQueryTextSubmit(query: String?): Boolean {

prestasiAdapter.filter.filter(query)
                    return true
                }

                override fun
onQueryTextChange(newText: String?): Boolean {

prestasiAdapter.filter.filter(query)
                    return true
                }
            })
        }
   searchItem.setOnActionExpandListener(object :
MenuItem.OnActionExpandListener {
            override fun
onMenuItemActionExpand(item: MenuItem?): Boolean {
```

```
                searchView.requestFocusFromTouch()
                val imm =
getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

imm.toggleSoftInput(InputMethodManager.SHOW_IMPLICIT
, 0)
                return true
            }

            override fun
onMenuItemActionCollapse(item: MenuItem?): Boolean {
                val imm =
getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

imm.hideSoftInputFromWindow(currentFocus?.windowToke
n, 0)
                return true
            }
        })
        return super.onCreateOptionsMenu(menu)
    }

    override fun onOptionsItemSelected(item:
MenuItem): Boolean {
        return when (item.itemId){
            R.id.action_filter -> {
                val intent = Intent(this,
FilterPrestasi::class.java)
                val listTingkat =
getList(R.array.tingkat)
                val listPeringkat =
vm.getListPeringkat()
                val listKategoriPeserta =
vm.getListKategoriPeserta()
```

```
intent.putExtra(FilterPrestasi.ITEM_TINGKAT,
listTingkat)

intent.putExtra(FilterPrestasi.ITEM_PERINGKAT,
listPeringkat)

intent.putExtra(FilterPrestasi.ITEM_KATEGORI_PESERTA
, listKategoriPeserta)

intent.putExtra(FilterPrestasi.PREVIOUS_FILTER,
modelFilterPrestasi)
                startActivityForResult(intent,
FILTER_CODE)
                true
            }
            R.id.action_download -> {
                val intent = Intent(this,
DownloadActivity::class.java)
                startActivityForResult(intent, 300)
                true
            }
            else -> return
super.onOptionsItemSelected(item)
        }
    }

    private fun getList(resource: Int):
ArrayList<String> {
        val array =
resources.getStringArray(resource)
        val arrayList = ArrayList<String>()
        arrayList.addAll(array)
        return arrayList
    }
```

```
    private fun getFilterValue(values:
ArrayList<String>?): String? {
        if (values!!.isEmpty()) return null
        var result = ""
        for (item in values) {
            if (result != "") result = "$result,"
            result = "$result$item"
        }
        return result
    }

    private fun getDepartemenId(values:
ArrayList<String>?): String? {
        if (values!!.isEmpty()) return null
        val dept = Helper.getAllDepartemen()
        var result = ""
        for (item in values) {
            if (result != "") result = "$result,"
            result =
"$result${dept.indexOf(item)+1}"
        }
        return result
    }

    override fun onActivityResult(requestCode: Int,
resultCode: Int, data: Intent?) {
        when (requestCode) {
            FILTER_CODE -> {
                if (resultCode ==
Activity.RESULT_OK) {
```

```
                    list.clear()
                    prestasiAdapter.setData(list)
                    val resultDepartemen =
data?.getStringArrayListExtra(FILTER_DEPARTEMEN)
                    val resultKategoriPeserta =
data?.getStringArrayListExtra(FILTER_KATEGORI_PESERT
A)
                    val resultKategoriPrestasi =
data?.getStringArrayListExtra(FILTER_KATEGORI_PRESTA
SI)
                    val resultCapaian =
data?.getStringArrayListExtra(FILTER_CAPAIAN)
                    modelFilterPrestasi =
ModelFilterPrestasi(resultDepartemen as
ArrayList<String>, resultKategoriPeserta as
ArrayList<String>, resultKategoriPrestasi as
ArrayList<String>, resultCapaian as
ArrayList<String>)
                    vm.getFilteredPrestasiDosen(
                        departemen =
getDepartemenId(resultDepartemen),
                        kategoriPeserta =
getFilterValue(resultKategoriPeserta),
                        kategoriPrestasi =
getFilterValue(resultKategoriPrestasi),
                        capaian =
getFilterValue(resultCapaian),
                        year = null,
                        visibility = View.VISIBLE
                    )
                }
            }
```

```
            ADD_CODE -> {
                if (resultCode ==
Activity.RESULT_OK) {
                    vm.getPrestasi(View.VISIBLE,
false)
                }
            }
            DETAIL_CODE -> {
                if (resultCode ==
Activity.RESULT_OK) {
                    vm.getPrestasi(View.VISIBLE,
false)
                }
            }
        }
        super.onActivityResult(requestCode,
resultCode, data)
    }
}
```

Kode Sumber 4.10: Kode Sumber Halaman Prestasi Dosen

4.2.6. Menampilkan Halaman Training

Halaman training berisi daftar training yang tercatat. Field yang ditampilkan adalah nama, pelaku, tanggal, tempat, departemen dan tingkat. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.11 dan 4.12.

```
import android.app.Activity
import android.content.Context
import android.content.Intent
import android.os.Build
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.view.View
```

```
import android.view.inputmethod.InputMethodManager
import android.widget.SearchView
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.isVisible
import
androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.android.material.snackbar.Snackbar
import com.kpbois.fteicmobile.Helper
import com.kpbois.fteicmobile.LoginActivity
import com.kpbois.fteicmobile.R
import com.kpbois.fteicmobile.retrofit.ApiService
import
kotlinx.android.synthetic.main.activity_view2.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
```

Kode Sumber 4.11: Dependensi Halaman Training

```
class ViewTraining : AppCompatActivity(),
View.OnClickListener {

    companion object {
        const val FILTER_CODE = 100
        const val ADD_TRAINING = 101
        const val TRAINING_DETAIL = 102
    }

    private lateinit var trainingAdapter:
AdapterTraining
    private var modelFilter = ModelFilter()
    private val listJenis = arrayListOf<String>()

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_view2)
        setSupportActionBar(toolbar2)
        supportActionBar?.apply {
            setDisplayHomeAsUpEnabled(true)
            title = "Training"
        }
        fab_add.setOnClickListener(this)
        sw_list.setOnRefreshListener { filterData()
}
        prepareRecyclerView()
    }
```

```
    override fun onCreateOptionsMenu(menu: Menu?):
Boolean {
        menuInflater.inflate(R.menu.view_menu, menu)

        val searchItem =
menu?.findItem(R.id.action_search)
        val searchView = searchItem?.actionView as
SearchView
        searchView.apply {
            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.Q) {
                isIconifiedByDefault = false
                isIconified = false
            }
            setOnQueryTextListener(object :
SearchView.OnQueryTextListener {
                override fun onQueryTextSubmit(p0:
String?): Boolean {

trainingAdapter.filter.filter(p0)
                    return true
                }

                override fun onQueryTextChange(p0:
String?): Boolean {

trainingAdapter.filter.filter(p0)
                    return true
                }
            })
        }
```

```
        searchItem.setOnActionExpandListener(object
: MenuItem.OnActionExpandListener {
            override fun
onMenuItemActionCollapse(item: MenuItem): Boolean {
                val imm =
getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

imm.hideSoftInputFromWindow(currentFocus?.windowToke
n, 0)
                return true // Return true to
collapse action view
            }

            override fun
onMenuItemActionExpand(item: MenuItem): Boolean {
                searchView.requestFocusFromTouch()
                val imm =
getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

imm.toggleSoftInput(InputMethodManager.SHOW_IMPLICIT
, 0)
                return true // Return true to expand
action view
            }
        }
        )

        return super.onCreateOptionsMenu(menu)
    }
```

```
    override fun onOptionsItemSelected(item:
MenuItem): Boolean {
        return when (item.itemId) {
            R.id.action_filter -> {
                val intent = Intent(this,
FilterTraining::class.java)
                intent.putExtra("model_filter",
modelFilter)
                intent.putExtra("list_jenis",
listJenis)
                startActivityForResult(intent,
FILTER_CODE)
                true
            }
            else ->
super.onOptionsItemSelected(item)
        }
    }

    override fun onActivityResult(requestCode: Int,
resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode,
resultCode, data)
        if (resultCode == Activity.RESULT_OK) {
            when (requestCode) {
                FILTER_CODE -> {
                    if (data != null) {
                        modelFilter =
data.getParcelableExtra("model_filter")
                        filterData()
                    }
                }
                ADD_TRAINING, TRAINING_DETAIL ->
filterData()
            }
        }
    }
```

```
    override fun onClick(p0: View?) {
        when (p0?.id) {
            fab_add.id -> {
                val intent = Intent(this,
AddTraining::class.java)
                startActivityForResult(intent,
ADD_TRAINING)
            }
        }
    }

    private fun prepareRecyclerView() {
        trainingAdapter = AdapterTraining().apply {
            onClickListener = object :
AdapterTraining.OnClickListener {
                override fun onClick(data:
ModelTraining) {
                    val intent =
Intent(this@ViewTraining,
DetailTraining::class.java)

intent.putExtra(DetailTraining.TRAINING_DETAIL,
data)
                    startActivityForResult(intent,
TRAINING_DETAIL)
                }
            }
        }
        rv_list.apply {
            layoutManager =
LinearLayoutManager(this@ViewTraining,
RecyclerView.VERTICAL, false)
            adapter = trainingAdapter
        }
        filterData()
    }
```

```
private fun filterData() {
        sw_list.isRefreshing = false
        rv_list.isVisible = false
        progressBar.isVisible = true
        val token = "Token " + Helper.getToken(this)
        var jenis = ""
        modelFilter.jenis.forEach { jenis =
"$it,$jenis" }
        ApiService.endpoint.getTraining(
            token,
            if (jenis.isNotBlank())
jenis.removeSuffix(",") else null,
            if
(modelFilter.tanggalMulai.isNotBlank())
modelFilter.tanggalMulai else null,
            if
(modelFilter.tanggalSelesai.isNotBlank())
modelFilter.tanggalSelesai else null
        ).enqueue(object :
Callback<List<ModelTraining>> { override fun
onResponse(
                call: Call<List<ModelTraining>>,
                response:
Response<List<ModelTraining>>
            ) {
                progressBar.isVisible = false
                if (response.isSuccessful &&
response.body() != null) {
                    val list = response.body()!!

if(!Helper.getUserInfo()!!.isAdmin) {
                        list.filter { it.isValidated
}
                    }
```

```
trainingAdapter.setData(ArrayList(list))
                    list.forEach {
                        if
(!listJenis.contains(it.jenis))
listJenis.add(it.jenis)
                    }
                    if (list.isEmpty()) {
                        tv_notfound.isVisible = true
                        rv_list.isVisible = false
                    } else {
                        tv_notfound.isVisible =
false
                        rv_list.isVisible = true
                    }
                } else {
                    if (response.code() == 401) {

startActivity(Intent(this@ViewTraining,
LoginActivity::class.java))
                        finish()
                    }

Snackbar.make(findViewById(android.R.id.content),
"Something went wrong.",
Snackbar.LENGTH_SHORT).show()
                }
            }
            override fun onFailure(call:
Call<List<ModelTraining>>, t: Throwable) {
                progressBar.isVisible = false
                t.printStackTrace()

Snackbar.make(findViewById(android.R.id.content),
t.localizedMessage, Snackbar.LENGTH_SHORT).show()
            }
        })
    }
}
```

Kode Sumber 4.12: Kode Sumber Halaman Training

4.2.7. Menampilkan Halaman Sertifikasi

Halaman training berisi daftar sertifikasi yang tercatat. Field yang ditampilkan adalah nama, pelaku, tanggal, lembaga, departemen

dan tingkat. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.13 dan 4.14.

```
import android.content.Context
import android.content.Intent
import android.os.Build
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.view.View
import android.view.inputmethod.InputMethodManager
import android.widget.SearchView
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.isVisible
import
androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.android.material.snackbar.Snackbar
import com.kpbois.fteicmobile.Helper
import com.kpbois.fteicmobile.LoginActivity
import com.kpbois.fteicmobile.R
import com.kpbois.fteicmobile.retrofit.ApiService
import
kotlinx.android.synthetic.main.activity_view2.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
```

Kode Sumber 4.13: Dependensi Halaman Sertifikasi

```
class ViewSertifikasi : AppCompatActivity(),
View.OnClickListener  {

    companion object {
        const val FILTER_CODE = 100
        const val ADD_TRAINING = 101
        const val SERTIFIKASI_DETAIL = 102
    }

    private lateinit var sertifikasiAdapter:
AdapterSertifikasi
    private var modelFilter = ModelFilter()
    private val listDepartemen = arrayListOf<Int>()

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_view2)
        setSupportActionBar(toolbar2)
        supportActionBar?.apply {
            setDisplayHomeAsUpEnabled(true)
            title = "Sertifikasi"
        }
        fab_add.setOnClickListener(this)
        sw_list.setOnRefreshListener { filterData()
}
        prepareRecyclerView()
    }
```

```
    override fun onCreateOptionsMenu(menu: Menu?):
Boolean {
        menuInflater.inflate(R.menu.view_menu, menu)

        val searchItem =
menu?.findItem(R.id.action_search)
        val searchView = searchItem?.actionView as
SearchView
        searchView.apply {
            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.Q) {
                isIconifiedByDefault = false
                isIconified = false
            }
            setOnQueryTextListener(object :
SearchView.OnQueryTextListener {
                override fun onQueryTextSubmit(p0:
String?): Boolean {

sertifikasiAdapter.filter.filter(p0)
                    return true
                }

                override fun onQueryTextChange(p0:
String?): Boolean {

sertifikasiAdapter.filter.filter(p0)
                    return true
                }
            })
        }
```

```
        searchItem.setOnActionExpandListener(object
: MenuItem.OnActionExpandListener {
            override fun
onMenuItemActionCollapse(item: MenuItem): Boolean {
                val imm =
getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

imm.hideSoftInputFromWindow(currentFocus?.windowToke
n, 0)
                return true // Return true to
collapse action view
            }

            override fun
onMenuItemActionExpand(item: MenuItem): Boolean {
                searchView.requestFocusFromTouch()
                val imm =
getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

imm.toggleSoftInput(InputMethodManager.SHOW_IMPLICIT
, 0)
                return true // Return true to expand
action view
            }
        }
        )

        return super.onCreateOptionsMenu(menu)
    }
```

```
    override fun onOptionsItemSelected(item:
MenuItem): Boolean {
        return when (item.itemId) {
            R.id.action_filter -> {
                true
            }
            else ->
super.onOptionsItemSelected(item)
        }
    }

    override fun onClick(p0: View?) {
        when (p0?.id) {
            fab_add.id -> {

            }
        }
    }
    private fun prepareRecyclerView() {
        sertifikasiAdapter =
AdapterSertifikasi().apply {
            onClickListener = object :
AdapterSertifikasi.OnClickListener {
                override fun onClick(data:
ModelSertifikasi) {
                    val intent =
Intent(this@ViewSertifikasi,
DetailSertifikasi::class.java)

intent.putExtra(DetailSertifikasi.SERTIFIKASI_DETAIL
, data)
                    startActivityForResult(intent,
SERTIFIKASI_DETAIL)
                }
            }
        }
        rv_list.apply {
            layoutManager =
LinearLayoutManager(this@ViewSertifikasi,
RecyclerView.VERTICAL, false)
            adapter = sertifikasiAdapter
        }
        filterData()
    }
```

```
    private fun filterData() {
        sw_list.isRefreshing = false
        rv_list.isVisible = false
        progressBar.isVisible = true
        val token = "Token " + Helper.getToken(this)
        var depts = ""
        modelFilter.departemen.forEach { depts =
"$it.$depts" }
        ApiService.endpoint.getSertifikasi(
            token,
            if (depts.isNotBlank())
depts.removeSuffix(",") else null,
            if (modelFilter.tanggal.isNotBlank())
modelFilter.tanggal else null,
            if
(modelFilter.tanggalBerakhir.isNotBlank())
modelFilter.tanggalBerakhir else null
        ).enqueue(object :
Callback<List<ModelSertifikasi>>{
            override fun onResponse(
                call: Call<List<ModelSertifikasi>>,
                response:
Response<List<ModelSertifikasi>>
            ) {
                progressBar.isVisible = false
                if (response.isSuccessful &&
response.body() != null) {
                    val list = response.body()!!

if(!Helper.getUserInfo()!!.isAdmin) {
                        list.filter { it.isValidated
}
                    }

sertifikasiAdapter.setData(ArrayList(list))
                    list.forEach {
                        if
(!listDepartemen.contains(it.departemen))
it.departemen?.let { it1 ->
                            listDepartemen.add(
                                it1
                            )
                        }
                    }
                    if (list.isEmpty()) {
                        tv_notfound.isVisible = true
                        rv_list.isVisible = false
                    } else {
                        tv_notfound.isVisible =
false
                        rv_list.isVisible = true
                    }
```

```
                } else {
                    if (response.code() == 401) {

startActivity(Intent(this@ViewSertifikasi,
LoginActivity::class.java))
                        finish()
                    }

Snackbar.make(findViewById(android.R.id.content),
"Something went wrong.",
Snackbar.LENGTH_SHORT).show()
                }
            }

            override fun onFailure(call:
Call<List<ModelSertifikasi>>, t: Throwable) {
                progressBar.isVisible = false
                t.printStackTrace()

Snackbar.make(findViewById(android.R.id.content),
t.localizedMessage, Snackbar.LENGTH_SHORT).show()
            }
        })
    }
}
```

Kode Sumber 4.14: Kode Sumber Halaman Sertifikasi

4.2.8. Menampilkan Halaman Submission

Halaman submission berisi daftar kegiatan yang telah diajukan namun belum disetujui oleh admin. Field yang ditampilkan adalah nama, pelaku, dan jenis. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.15 dan 4.16.

```
import android.content.Context
import android.content.Intent
import android.os.Build
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.view.View
import android.view.inputmethod.InputMethodManager
import android.widget.SearchView
import androidx.appcompat.app.AppCompatActivity
import
androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.kpbois.fteicmobile.Helper
import com.kpbois.fteicmobile.R
import com.kpbois.fteicmobile.retrofit.ApiService
import
kotlinx.android.synthetic.main.activity_view2.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
```

Kode Sumber 4.15: Dependensi Halaman Submission

```
class ViewSubmission: AppCompatActivity() {

    private lateinit var recyclerView: RecyclerView
    private lateinit var adapterSubmission:
AdapterSubmission
    private val items = ArrayList<ModelSubmission>()
    private var filter: MenuItem? = null
    companion object{
        const val DETAIL_CODE = 101
    }

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_view2)
        setSupportActionBar(toolbar2)
        supportActionBar?.apply {
            title = "Submission"
            setDisplayHomeAsUpEnabled(true)
        }
        config()
    }

    override fun onStart() {
        getSubmissionList(View.VISIBLE)
        super.onStart()
    }

    private fun config() {
        fab_add.visibility = View.GONE

        adapterSubmission = AdapterSubmission()
        adapterSubmission.setOnClickListener(object
: AdapterSubmission.OnClickListener{
```

```
            override fun onItemClick(data:
ModelSubmission) {
                val intent =
Intent(this@ViewSubmission,
DetailSubmission::class.java)

intent.putExtra(DetailSubmission.MODUL, data.modul)
                intent.putExtra(DetailSubmission.ID,
data.id)
                startActivityForResult(intent,
DETAIL_CODE)
            }
        })

        recyclerView = rv_list
        recyclerView.apply {
            layoutManager =
LinearLayoutManager(this@ViewSubmission)
            setHasFixedSize(true)
        }

        sw_list.setOnRefreshListener {
            getSubmissionList(View.GONE)
            sw_list.isRefreshing = false
        }
    }

    private fun getSubmissionList(progressBarShow:
Int) {
        progressBar.visibility = progressBarShow
        items.clear()
        val token = "Token " + Helper.getToken(this)
```

```
ApiService.endpoint.getSubmission(token).enqueue(obj
ect : Callback<List<ModelSubmission>>{
            override fun onResponse(call:
Call<List<ModelSubmission>>, response:
Response<List<ModelSubmission>>) {
                progressBar.visibility = View.GONE
                val resp = response.body()!!
                if (!resp.isNullOrEmpty()) {
                    items.addAll(resp)
                    setRecyclerViewContent()
                }
            }

            override fun onFailure(call:
Call<List<ModelSubmission>>, t: Throwable) {
                progressBar.visibility = View.GONE
            }
        })
    }

    private fun setRecyclerViewContent() {
        adapterSubmission.setData(items)
        recyclerView.adapter = adapterSubmission
    }

    override fun onPrepareOptionsMenu(menu: Menu?):
Boolean {
        filter = menu?.findItem(R.id.action_filter)
        filter?.isVisible = false
        return super.onPrepareOptionsMenu(menu)
    }
```

```
    override fun onCreateOptionsMenu(menu: Menu?):
Boolean {
        menuInflater.inflate(R.menu.view_menu, menu)

        val searchItem =
menu?.findItem(R.id.action_search)
        val searchView = searchItem?.actionView as
SearchView
        searchView.apply {
            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.Q) {
                searchView.isIconifiedByDefault =
false
                searchView.isIconified = false
            }
            setOnQueryTextListener(object :
SearchView.OnQueryTextListener{
                override fun
onQueryTextSubmit(query: String?): Boolean {

adapterSubmission.filter.filter(query)
                    return true
                }

                override fun
onQueryTextChange(newText: String?): Boolean {

adapterSubmission.filter.filter(query)
                    return true
                }
            })
        }
```

```
        searchItem.setOnActionExpandListener(object
: MenuItem.OnActionExpandListener {
            override fun
onMenuItemActionExpand(item: MenuItem?): Boolean {
                searchView.requestFocusFromTouch()
                val imm =
getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

imm.toggleSoftInput(InputMethodManager.SHOW_IMPLICIT
, 0)
                return true
            }

            override fun
onMenuItemActionCollapse(item: MenuItem?): Boolean {
                val imm =
getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

imm.hideSoftInputFromWindow(currentFocus?.windowToke
n, 0)
                return true
            }
        })
        return super.onCreateOptionsMenu(menu)
    }
    override fun onActivityResult(requestCode: Int,
resultCode: Int, data: Intent?) {
        if (resultCode == RESULT_OK) {
            when (requestCode) {
                DETAIL_CODE ->
getSubmissionList(View.VISIBLE)
            }
        }
        super.onActivityResult(requestCode,
resultCode, data)
    }
}
```

Kode Sumber 4.16: Kode Sumber Halaman Submission

4.2.9. Menampilkan Halaman Ubah password

Halaman ini berupa fasilitas mengganti kata sandi. Field yang ditampilkan adalah nama, depertemen, password lama, password baru,

dan konfirmasi password. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.17 dan 4.18.

```
import android.os.Bundle
import android.view.View
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.kpbois.fteicmobile.retrofit.ApiService
import
kotlinx.android.synthetic.main.activity_change_passw
ord.*
import okhttp3.ResponseBody
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
```

Kode Sumber 4.17: Dependensi Halaman Ubah Password

```
class ChangePasswordActivity :AppCompatActivity(),
View.OnClickListener {

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)

setContentView(R.layout.activity_change_password)
        bt_save.setOnClickListener(this)
        bt_cancel.setOnClickListener(this)
    }

    override fun onClick(v: View?) {
        when(v?.id){
            bt_save.id->changePass()
            bt_cancel.id->finish()
        }
    }

    private fun changePass(){
        pb_loading.visibility = View.VISIBLE

ApiService.endpoint.changePassword(Helper.getToken(a
pplicationContext)!!,
et_old.text.toString(),et_new.text.toString())
            .enqueue(object : Callback<ResponseBody>
{
                override fun onResponse(
                    call: Call<ResponseBody>,
                    response: Response<ResponseBody>
                ) {
                    pb_loading.visibility =
View.GONE
                    if(response.isSuccessful())
                        Toast.makeText(
                            applicationContext,
                            "Success:
${response.message()}",
                            Toast.LENGTH_SHORT
                        ).show()
                    else
                        Toast.makeText(
                            applicationContext,
                            "Failed:
${response.message()}",
                            Toast.LENGTH_SHORT
                        ).show()
                }
```

```
                override fun onFailure(call:
Call<ResponseBody>, t: Throwable) {
                    pb_loading.visibility =
View.GONE
                    Toast.makeText(
                        applicationContext,
                        "Failed: Something went
wrong",
                        Toast.LENGTH_SHORT
                    ).show()
                }
            })
    }
}
```

Kode Sumber 4.18: Kode Sumber Halaman Ubah Password

4.2.10. Menampilkan Halaman Download

Halaman download berisi form untuk mengunggah berkas sesuai field yang ditampilkan. Field yang ditampilkan adalah jenis, departemen, dan tahun. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.19 dan 4.20.

```
import android.Manifest
import android.app.DownloadManager
import android.content.Context
import android.content.pm.PackageManager
import android.net.Uri
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.os.Environment
import android.view.View
import android.widget.ArrayAdapter
import android.widget.Toast
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import com.kpbois.fteicmobile.Helper
import com.kpbois.fteicmobile.R
import
kotlinx.android.synthetic.main.activity_download.*
import java.util.*
import kotlin.collections.ArrayList
```

Kode Sumber 4.19: Dependensi Halaman Download

```
class DownloadActivity : AppCompatActivity(),
View.OnClickListener {

    companion object{
        const val PERMISSION_CODE = 1
    }
    private var base_url =
"http://206.189.159.64:8000/api"

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_download)
        supportActionBar?.apply {
            title = "Download"
            setDisplayHomeAsUpEnabled(true)
        }
        config()
    }

    private fun config() {
        ArrayAdapter.createFromResource(this,
R.array.modul, R.layout.spinner_item)
            .also { arrayAdapter ->

arrayAdapter.setDropDownViewResource(android.R.layou
t.simple_spinner_dropdown_item)
                spinner_download_modul.adapter =
arrayAdapter
            }
        val dept = ArrayList<String>()
        dept.add("All")
```

```
        dept.addAll(Helper.getAllDepartemen())
        val departemenAdapter = ArrayAdapter(this,
R.layout.spinner_item, dept)

departemenAdapter.setDropDownViewResource(android.R.
layout.simple_spinner_dropdown_item)
        spinner_download_departemen.adapter =
departemenAdapter

        val tahunAdapter = ArrayAdapter(this,
R.layout.spinner_item, Helper.getYear())

tahunAdapter.setDropDownViewResource(android.R.layou
t.simple_spinner_dropdown_item)
        spinner_download_tahun.adapter =
tahunAdapter

        btn_download.setOnClickListener(this)
    }

    override fun onSupportNavigateUp(): Boolean {
        this.onBackPressed()
        finish()
        return true
    }

    private fun checkPerm(): Boolean {
        val result =
ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE)
        return result ==
PackageManager.PERMISSION_GRANTED
    }
```

```
    private fun requestPerm() {
        ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.WRITE_EXTERNAL_STORAGE),
PERMISSION_CODE)

    }

    override fun onClick(v: View?) {
        if (v?.id == R.id.btn_download) {
            if (checkPerm()) {
                var modul =
spinner_download_modul.selectedItem.toString().toLow
erCase(Locale.ROOT)
                modul = if (modul.equals("prestasi
dosen", true)) "prestasi/dosen" else modul
                val departemen =
spinner_download_departemen.selectedItem.toString().
toLowerCase(Locale.ROOT)
                var url =
"$base_url/$modul/export?tahun=${spinner_download_ta
hun.selectedItem}"
                if (departemen != "all") url =
"$url&departemen=${Helper.getIdDepartment(spinner_do
wnload_departemen.selectedItem.toString())}"
                val token = "Token " +
Helper.getToken(this)
                val request =
DownloadManager.Request(Uri.parse(url))

.setNotificationVisibility(DownloadManager.Request.V
ISIBILITY_VISIBLE_NOTIFY_COMPLETED)

.setTitle(getString(R.string.app_name))
                    .setDescription("Downloading")
                    .setAllowedOverMetered(true)
                    .setAllowedOverRoaming(true)
```

```
.setMimeType("application/vnd.ms-excel")

.setDestinationInExternalPublicDir(Environment.DIREC
TORY_DOWNLOADS, "" + System.currentTimeMillis())

.addRequestHeader("Authorization", token)
                val downloadManager =
getSystemService(Context.DOWNLOAD_SERVICE) as
DownloadManager
                downloadManager.enqueue(request)
            }
            else requestPerm()
        }
    }

    override fun
onRequestPermissionsResult(requestCode: Int,
permissions: Array<out String>, grantResults:
IntArray) {
        if (requestCode == PERMISSION_CODE) {
            if (grantResults.isNotEmpty() &&
grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                Toast.makeText(this, "Downloading
File", Toast.LENGTH_SHORT).show()
            }
            else Toast.makeText(this, "Permission
Denied", Toast.LENGTH_SHORT).show()
        }
    }
}
```

Kode Sumber 4.20: Kode Sumber Halaman Download

4.2.11. Menampilkan Halaman Detail Training

Halaman detail berisi detil tiap training. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.21 dan 4.22.

```
import android.content.Context
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.view.View
import android.widget.ProgressBar
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import
androidx.swiperefreshlayout.widget.CircularProgressD
rawable
import com.bumptech.glide.Glide
import com.kpbois.fteicmobile.Helper
import com.kpbois.fteicmobile.R
import com.kpbois.fteicmobile.retrofit.ApiService
import
kotlinx.android.synthetic.main.activity_detail_train
ing.*
import okhttp3.ResponseBody
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
```

Kode Sumber 4.21: Dependensi Halaman Detail Training

```
class DetailTraining : AppCompatActivity() {

    companion object {
        const val TRAINING_DETAIL =
"training_detail"
    }

    private var data: ModelTraining? = null

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)

setContentView(R.layout.activity_detail_training)
        supportActionBar?.apply {
            title = "Detail Training"
            setDisplayHomeAsUpEnabled(true)
        }
        data = intent.getParcelableExtra(
            TRAINING_DETAIL)
        showData()
    }

    override fun onCreateOptionsMenu(menu: Menu?):
Boolean {
        menuInflater.inflate(R.menu.kultam_menu,
menu)
        return super.onCreateOptionsMenu(menu)
    }
```

```
    private fun showData() {
        val circularProgressDrawable =
CircularProgressDrawable(this)
        circularProgressDrawable.strokeWidth = 5f
        circularProgressDrawable.centerRadius = 30f
        circularProgressDrawable.start()
        if (data != null) {
            tv_training.text = data!!.judul
            tv_jenis.text = data!!.jenis
            tv_peserta.text = data!!.peserta
            tv_tanggal_mulai.text =
data!!.date_start
            tv_tanggal_selesai.text =
data!!.date_end
            tv_tempat.text = data!!.tempat
            tv_dept.text =
Helper.getDepartemen(data!!.departemen)
            Glide.with(this)
                .load(data!!.filepath)

.placeholder(circularProgressDrawable)
                .into(iv_bukti)
        }
    }
```

```
    fun deleteData(progressBar: ProgressBar?, token:
String, id: Int, context: Context) {
        progressBar?.visibility = View.VISIBLE
        ApiService.endpoint.deleteTraining(
            token, id
        ).enqueue(object : Callback<ResponseBody> {
            override fun onResponse(
                call: Call<ResponseBody>,
                response: Response<ResponseBody>
            ) {
                if (response.isSuccessful) {
                    Toast.makeText(
                        context,
                        "Success:
${response.message()}",
                        Toast.LENGTH_SHORT
                    ).show()
                }
                else Toast.makeText(
                    context,
                    "Failed: ${response.message()}",
                    Toast.LENGTH_SHORT
                ).show()
                progressBar?.visibility = View.GONE
            }
            override fun onFailure(call:
Call<ResponseBody>, t: Throwable) {
                Toast.makeText(
                    context,
                    "Failed: ${t.localizedMessage}",
                    Toast.LENGTH_SHORT
                ).show()
                progressBar?.visibility = View.GONE
            }
        })
    }
```

```
    fun validateData(progressBar: ProgressBar?,
token: String, id: Int, context: Context) {
        progressBar?.visibility = View.VISIBLE
        ApiService.endpoint.validateTraining(
            token, id
        ).enqueue(object : Callback<ResponseBody> {
            override fun onResponse(
                call: Call<ResponseBody>,
                response: Response<ResponseBody>
            ) {
                if (response.isSuccessful) {
                    Toast.makeText(
                        context,
                        "Success:
${response.message()}",
                        Toast.LENGTH_SHORT
                    ).show()
                }
                else Toast.makeText(
                    context,
                    "Failed: ${response.message()}",
                    Toast.LENGTH_SHORT
                ).show()
                progressBar?.visibility = View.GONE
            }
            override fun onFailure(call:
Call<ResponseBody>, t: Throwable) {
                Toast.makeText(
                    context,
                    "Failed: ${t.localizedMessage}",
                    Toast.LENGTH_SHORT
                ).show()
                progressBar?.visibility = View.GONE
            }
        })
    }
}
```

Kode Sumber 4.22: Kode Sumber Halaman Detail Training

4.2.12. Menampilkan Halaman Detail Konferensi / Jurnal

Halaman detail berisi detil tiap kegiatan. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.23 dan 4.24.

```
import android.app.Activity
import android.content.Context
import android.content.Intent
import android.net.Uri
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.view.View
import android.widget.ProgressBar
import android.widget.Toast
import
androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.kpbois.fteicmobile.Helper
import com.kpbois.fteicmobile.R
import com.kpbois.fteicmobile.retrofit.ApiService
import
kotlinx.android.synthetic.main.activity_detail_view.
*
import okhttp3.ResponseBody
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
```

Kode Sumber 4.23: Dependensi Halaman Detail Konferensi / Jurnal

```
class DetailKonjur : AppCompatActivity() {
    private lateinit var recyclerView: RecyclerView
    private var list: ArrayList<ModelKonjurDetail> =
arrayListOf()
    private var name = ""
    private var id = 0
    private var delete: MenuItem? = null
    private var validate: MenuItem? = null
    private var isValidated = false
    private var isAdmin = false
    companion object {
        const val KONJUR_DETAIL = "konjur_detail"
        const val STATE = "state"
    }
    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)

setContentView(R.layout.activity_detail_view)
        setSupportActionBar(toolbar_detail)
        name = intent.getStringExtra(STATE)!!
```

```
        supportActionBar?.apply {
            title = "Detail $name"
            setDisplayHomeAsUpEnabled(true)
        }
        if (name.equals("konferensi", true)) {
            getItemKonferensi(intent!!)
        }
        else if (name.equals("jurnal", true))
getItemJurnal(intent)

        configuration()
        checkIsAdmin()
    }

    private fun checkIsAdmin() {
        val userInfo = Helper.getUserInfo()
        isAdmin = userInfo?.isAdmin as Boolean
    }

    private fun getItemJurnal(intent: Intent) {
        val detail =
intent.getParcelableExtra<ModelJurnal>(KONJUR_DETAIL
)
        if (detail != null) {
            id = detail.id
            var modelJurnalDetail =
ModelKonjurDetail("Judul", detail.judul)
            list.add(modelJurnalDetail)
            modelJurnalDetail =
ModelKonjurDetail("Author", detail.author)
            list.add(modelJurnalDetail)
            modelJurnalDetail =
ModelKonjurDetail("Departemen",
Helper.getDepartemen(detail.departemen?.toInt()))
            list.add(modelJurnalDetail)
```

```
            modelJurnalDetail =
ModelKonjurDetail("Published At",
detail.published_at)
            list.add(modelJurnalDetail)
            modelJurnalDetail =
ModelKonjurDetail("Jurnal Vol", detail.jurnal_vol)
            list.add(modelJurnalDetail)
            modelJurnalDetail =
ModelKonjurDetail("Jurnal No", detail.jurnal_no)
            list.add(modelJurnalDetail)
            modelJurnalDetail =
ModelKonjurDetail("Url", detail.url)
            list.add(modelJurnalDetail)
            modelJurnalDetail =
ModelKonjurDetail("Tahun", detail.tahun)
            list.add(modelJurnalDetail)
            modelJurnalDetail =
ModelKonjurDetail("Tingkat", detail.tingkat)
            list.add(modelJurnalDetail)
            modelJurnalDetail =
ModelKonjurDetail("Scopus", detail.scopus)
            list.add(modelJurnalDetail)
            isValidated =
detail.is_validated.toBoolean()
        }
    }

private fun getItemKonferensi(intent: Intent) {
        val detail =
intent.getParcelableExtra<ModelKonferensi>(KONJUR_DE
TAIL)
        if (detail != null) {
            id = detail.id
```

```
var modelKonferensiDetail =
ModelKonjurDetail("Judul", detail.judul)
            list.add(modelKonferensiDetail)
            modelKonferensiDetail =
ModelKonjurDetail("Author", detail.author)
            list.add(modelKonferensiDetail)
            modelKonferensiDetail =
ModelKonjurDetail("Departemen",
Helper.getDepartemen(detail.departemen?.toInt()))
            list.add(modelKonferensiDetail)
            modelKonferensiDetail =
ModelKonjurDetail("Published At",
detail.published_at)
            list.add(modelKonferensiDetail)
            modelKonferensiDetail =
ModelKonjurDetail("Url", detail.url)
            list.add(modelKonferensiDetail)
            modelKonferensiDetail =
ModelKonjurDetail("Tahun", detail.tahun)
            list.add(modelKonferensiDetail)
            modelKonferensiDetail =
ModelKonjurDetail("Tingkat", detail.tingkat)
            list.add(modelKonferensiDetail)
            modelKonferensiDetail =
ModelKonjurDetail("Konf Hal", detail.konf_hal)
            list.add(modelKonferensiDetail)
            modelKonferensiDetail =
ModelKonjurDetail("Tempat", detail.tempat)
list.add(modelKonferensiDetail)
```

```
modelKonferensiDetail = ModelKonjurDetail("Tanggal
Mulai", detail.tanggal_mulai)
            list.add(modelKonferensiDetail)
            modelKonferensiDetail =
ModelKonjurDetail("Tanggal Selesai",
detail.tanggal_selesai)
            list.add(modelKonferensiDetail)
            modelKonferensiDetail =
ModelKonjurDetail("Scopus", detail.scopus)
            list.add(modelKonferensiDetail)
            isValidated =
detail.isValidated.toBoolean()
        }
    }

    private fun configuration() {
        val adapterDetail =
AdapterDetailKonjur(list)
        adapterDetail.setOnClickUrl(object :
AdapterDetailKonjur.OnClickUrl{
            override fun onClick(data: String) {
                val browserIntent =
Intent(Intent.ACTION_VIEW, Uri.parse(data))
                startActivity(browserIntent)
            }
        })
        recyclerView = rv_detail
        recyclerView.apply {
            setHasFixedSize(true)
            layoutManager =
LinearLayoutManager(this@DetailKonjur)
            adapter = adapterDetail
        }
    }
```

```
override fun onPrepareOptionsMenu(menu: Menu?):
Boolean {
        delete = menu?.findItem(R.id.action_delete)
        validate =
menu?.findItem(R.id.action_validate)
        if (isAdmin) {
            validate?.isVisible = !isValidated
            delete?.isVisible = true
        }
        else {
            validate?.isVisible = false
            delete?.isVisible = false
        }
        return super.onPrepareOptionsMenu(menu)
    }

    override fun onCreateOptionsMenu(menu: Menu?):
Boolean {
        menuInflater.inflate(R.menu.detail_menu,
menu)
        return super.onCreateOptionsMenu(menu)
    }

override fun onOptionsItemSelected(item: MenuItem):
Boolean {
        return when (item.itemId) {
            R.id.action_delete -> {
                pb_detail.visibility = View.VISIBLE
                val token = "Token " +
Helper.getToken(this)
```

```
if (name.equals("konferensi", true))
deleteKonferensi(pb_detail, token, id, this)
                else if (name.equals("jurnal",
true)) deleteJurnal(pb_detail, token, id, this)
                true
            }
            R.id.action_validate -> {
                pb_detail.visibility = View.VISIBLE
                val token = "Token " +
Helper.getToken(this)
                if (name.equals("konferensi", true))
validateKonferensi(pb_detail, token, id, this)
                else if (name.equals("jurnal",
true)) validateJurnal(pb_detail, token, id, this)
                true
            }
            else -> return
super.onOptionsItemSelected(item)
        }
    }

fun validateJurnal(progressBar: ProgressBar, token:
String, id: Int, context: Context) {
        ApiService.endpoint.validateJurnal(token,
id).enqueue(object : Callback<ResponseBody>{
            override fun onResponse(call:
Call<ResponseBody>, response:
Response<ResponseBody>) {
```

```
                progressBar.visibility = View.GONE
                val returnIntent = Intent()
                returnIntent.putExtra("result",
name)
                setResult(Activity.RESULT_OK,
returnIntent)
                finish()
            }

            override fun onFailure(call:
Call<ResponseBody>, t: Throwable) {
                progressBar.visibility = View.GONE
                Toast.makeText(context, "Gagal
validasi Jurnal", Toast.LENGTH_SHORT).show()
            }
        })
    }

    fun validateKonferensi(progressBar: ProgressBar,
token: String, id: Int, context: Context) {

ApiService.endpoint.validateKonferensi(token,
id).enqueue(object : Callback<ResponseBody>{
            override fun onResponse(call:
Call<ResponseBody>, response:
Response<ResponseBody>) {
                progressBar.visibility = View.GONE
                val returnIntent = Intent()
```

```
                returnIntent.putExtra("result",
name)
                setResult(Activity.RESULT_OK,
returnIntent)
                finish()
            }

            override fun onFailure(call:
Call<ResponseBody>, t: Throwable) {
                progressBar.visibility = View.GONE
                Toast.makeText(context, "Gagal
validasi Konferensi", Toast.LENGTH_SHORT).show()
            }
        })
    }

    fun deleteJurnal(progressBar: ProgressBar,
token: String, id: Int, context: Context) {
        ApiService.endpoint.deleteJurnal(
            token = token,
            id = id
        ).enqueue(object: Callback<ResponseBody>{
            override fun onFailure(call:
Call<ResponseBody>, t: Throwable) {
                progressBar.visibility = View.GONE
                Toast.makeText(context, "Gagal
menghapus Jurnal", Toast.LENGTH_SHORT).show()
            }

            override fun onResponse(call:
Call<ResponseBody>, response:
Response<ResponseBody>) {
                progressBar.visibility = View.GONE
```

```
                val returnIntent = Intent()
                returnIntent.putExtra("result",
name)
                setResult(Activity.RESULT_OK,
returnIntent)
                finish()
            }
        })
    }

    fun deleteKonferensi(progressBar: ProgressBar,
token: String, id: Int, context: Context) {
        ApiService.endpoint.deleteKonferensi(
            token = token,
            id = id
        ).enqueue(object: Callback<ResponseBody>{
            override fun onFailure(call:
Call<ResponseBody>, t: Throwable) {
                progressBar.visibility = View.GONE
                Toast.makeText(context, "Gagal
menghapus Konferensi", Toast.LENGTH_SHORT).show()
            }

            override fun onResponse(call:
Call<ResponseBody>, response:
Response<ResponseBody>) {
                progressBar.visibility = View.GONE
                val returnIntent = Intent()
                returnIntent.putExtra("result",
name)
                setResult(Activity.RESULT_OK,
returnIntent)
                finish()
            }
        })
    }

    override fun onSupportNavigateUp(): Boolean {
        this.onBackPressed()
        return true
    }
}
```

Kode Sumber 4.24: Kode Sumber Halaman Detail Konferensi atau Jurnal

4.2.13. Menampilkan Halaman Detail Prestasi

Halaman detail berisi detil tiap kegiatan. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.25 dan 4.26.

```
import android.app.Activity
import android.content.Context
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.view.View
import android.widget.ProgressBar
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import
androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.kpbois.fteicmobile.Helper
import com.kpbois.fteicmobile.R
import com.kpbois.fteicmobile.retrofit.ApiService
import
com.kpbois.fteicmobile.retrofit.PrestasiDosenModel
import
kotlinx.android.synthetic.main.activity_detail_view.
*
import okhttp3.ResponseBody
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
```

Kode Sumber 4.25: Dependensi Halaman Detail Prestasi

```
class DetailPrestasi : AppCompatActivity() {

    private val list : ArrayList<ModelPretasiDetail>
= arrayListOf()
    private var delete: MenuItem? = null
    private var validate: MenuItem? = null
    private lateinit var recyclerView: RecyclerView
    private var id = 0
    private var isValidated = false
    private var isAdmin = false
    companion object{
        const val PRESTASI_DETAIL =
"prestasi_detail"
    }

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)

setContentView(R.layout.activity_detail_view)
        setSupportActionBar(toolbar_detail)
        supportActionBar?.apply {
            title = "Detail Prestasi"
            setDisplayHomeAsUpEnabled(true)
        }
        val detail =
intent.getParcelableExtra<PrestasiDosenModel>(PRESTA
SI_DETAIL)

        if (detail != null) {
            val items = getItemDetail(detail)
            list.addAll(items)
        }

        setRecyclerView()
        checkIsAdmin()
    }
```

```
private fun checkIsAdmin() {
        val userInfo = Helper.getUserInfo()
        isAdmin = userInfo?.isAdmin as Boolean
    }

    private fun setRecyclerView() {
        val adapterDetailPrestasi =
AdapterDetailPrestasi(list)
        adapterDetailPrestasi.setOnClickUrl(object :
AdapterDetailPrestasi.OnClickUrl{
            override fun onClick(data: String) {
                val browserIntent =
Intent(Intent.ACTION_VIEW, Uri.parse(data))
                startActivity(browserIntent)
            }
        })
        recyclerView = rv_detail
        recyclerView.apply {
            layoutManager =
LinearLayoutManager(this@DetailPrestasi)
            setHasFixedSize(true)
            adapter = adapterDetailPrestasi
        }
    }

    fun getItemDetail(data: PrestasiDosenModel):
ArrayList<ModelPretasiDetail>{
        id = data.id as Int
        val list = ArrayList<ModelPretasiDetail>()
        var modelPretasiDetail =
ModelPretasiDetail("Nama
Penghargaan",data.nama_penghargaan)
```

```
        list.add(modelPretasiDetail)
        modelPretasiDetail =
ModelPretasiDetail("Jenis
Penghargaan",data.jenis_penghargaan)
        list.add(modelPretasiDetail)
        modelPretasiDetail =
ModelPretasiDetail("Nama Dosen", data.nama)
        list.add(modelPretasiDetail)
        modelPretasiDetail =
ModelPretasiDetail("Capaian", data.capaian)
        list.add(modelPretasiDetail)
        modelPretasiDetail =
ModelPretasiDetail("Departemen",
Helper.getDepartemen(data.departemen?.toInt()))
        list.add(modelPretasiDetail)
        modelPretasiDetail =
ModelPretasiDetail("Kategori Peserta",
data.kategori_peserta)
        list.add(modelPretasiDetail)
        modelPretasiDetail =
ModelPretasiDetail("Kategori Prestasi",
data.kategori_prestasi)
        list.add(modelPretasiDetail)
        modelPretasiDetail =
ModelPretasiDetail("Lembaga Penyelenggara",
data.lembaga_penyelenggara)
        list.add(modelPretasiDetail)
        modelPretasiDetail = ModelPretasiDetail("Web
Berita", data.web_berita as String)
        list.add(modelPretasiDetail)
        if (!data.filepath.isNullOrEmpty()) {
            modelPretasiDetail =
ModelPretasiDetail("Bukti", image = data.filepath)
```

```
            list.add(modelPretasiDetail)
        }
        isValidated = data.is_validated.toBoolean()
        return list
    }

    override fun onPrepareOptionsMenu(menu: Menu?):
Boolean {
        delete = menu?.findItem(R.id.action_delete)
        validate =
menu?.findItem(R.id.action_validate)
        if (isAdmin) {
            validate?.isVisible = !isValidated
            delete?.isVisible = true
        }
        else {
            validate?.isVisible = false
            delete?.isVisible = false
        }
        return super.onPrepareOptionsMenu(menu)
    }

    override fun onCreateOptionsMenu(menu: Menu?):
Boolean {
        menuInflater.inflate(R.menu.detail_menu,
menu)
        return super.onCreateOptionsMenu(menu)
    }
```

```
    override fun onOptionsItemSelected(item:
MenuItem): Boolean {
        return when (item.itemId) {
            R.id.action_delete -> {
                val token = "Token " +
Helper.getToken(this)
                AlertDialog.Builder(this,
R.style.AppCompatAlertDialogStyle)
                    .setTitle("Yakin ingin hapus?")
                    .setMessage("Data tidak dapat
dikembalikan setelah dihapus.")
                    .setPositiveButton("Ya") {
dialogInterface, _ ->
                        deletePrestasi(pb_detail,
token, id, this)
                        dialogInterface.dismiss()
                    }
                    .setNegativeButton("Tidak",
null)
                    .create().show()
                true
            }
            R.id.action_validate -> {
                val token = "Token " +
Helper.getToken(this)
                validatePrestasi(pb_detail, token,
id, this)
                true
            }
            else -> return
super.onOptionsItemSelected(item)
        }
    }
```

```
    fun deletePrestasi(progressBar: ProgressBar,
token: String, id: Int, context: Context){
        progressBar.visibility = View.VISIBLE

ApiService.endpoint.deletePrestasiDosen(token =
token, id = id).enqueue(object :
Callback<ResponseBody>{
            override fun onFailure(call:
Call<ResponseBody>, t: Throwable) {
                progressBar.visibility = View.GONE
                Toast.makeText(context, "Gagal
menghapus Prestasi", Toast.LENGTH_SHORT).show()
            }

            override fun onResponse(call:
Call<ResponseBody>, response:
Response<ResponseBody>) {
                progressBar.visibility = View.GONE
                val returnIntent = Intent()
                setResult(Activity.RESULT_OK,
returnIntent)
                finish()
            }
        })
    }

    fun validatePrestasi(progressBar: ProgressBar,
token: String, id: Int, context: Context) {
        progressBar.visibility = View.VISIBLE
```

```
ApiService.endpoint.validatePrestasiDosen(token,
id).enqueue(object : Callback<ResponseBody>{
            override fun onResponse(call:
Call<ResponseBody>, response:
Response<ResponseBody>) {
                progressBar.visibility = View.GONE
                val returnIntent = Intent()
                setResult(Activity.RESULT_OK,
returnIntent)
                finish()
            }

            override fun onFailure(call:
Call<ResponseBody>, t: Throwable) {
                progressBar.visibility = View.GONE
                Toast.makeText(context, "Gagal
validasi Prestasi", Toast.LENGTH_SHORT).show()
            }

        })
    }

    override fun onSupportNavigateUp(): Boolean {
        this.onBackPressed()
        return true
    }

    override fun onBackPressed() {
        super.onBackPressed()
        val returnIntent = Intent()
        setResult(Activity.RESULT_CANCELED,
returnIntent)
        finish()
    }
}
```

Kode Sumber 4.26: Kode Sumber Halaman Detail

4.2.14. Menampilkan Halaman Detail Sertifikasi

Halaman detail berisi detil tiap sertifikasi. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.27 dan 4.28.

```
import android.content.Context
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.view.View
import android.widget.ProgressBar
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import
androidx.swiperefreshlayout.widget.CircularProgressD
rawable
import com.bumptech.glide.Glide
import com.kpbois.fteicmobile.Helper
import com.kpbois.fteicmobile.R
import com.kpbois.fteicmobile.retrofit.ApiService
import
kotlinx.android.synthetic.main.activity_detail_serti
fikasi.*
import okhttp3.ResponseBody
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
```

Kode Sumber 4.27: Dependensi Halaman Detail Sertifikasi

```
class DetailSertifikasi:AppCompatActivity() {
    companion object {
        const val SERTIFIKASI_DETAIL =
"sertifikasi_detail"
    }

    private var data: ModelSertifikasi? = null

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)

setContentView(R.layout.activity_detail_sertifikasi)
        supportActionBar?.apply {
            title = "Detail Sertifikasi"
            setDisplayHomeAsUpEnabled(true)
        }
        data =
intent.getParcelableExtra(SERTIFIKASI_DETAIL)
        showData()
    }
```

```
    override fun onCreateOptionsMenu(menu: Menu?):
Boolean {
        menuInflater.inflate(R.menu.detail_menu,
menu)
        return super.onCreateOptionsMenu(menu)
    }

    override fun onOptionsItemSelected(item:
MenuItem): Boolean {
        return when (item.itemId) {
            R.id.action_delete -> {
                AlertDialog.Builder(this,
R.style.AppCompatAlertDialogStyle)
                    .setTitle("Yakin ingin hapus?")
                    .setMessage("Data tidak dapat
dikembalikan setelah dihapus.")
                    .setPositiveButton("Ya") {
dialogInterface, _ ->

deleteData(null,Helper.getToken(applicationContext)!
!,data!!.id!!,applicationContext)
                        dialogInterface.dismiss()
                    }
                    .setNegativeButton("Tidak",
null)
                    .create().show()
                true
            }
            else ->
super.onOptionsItemSelected(item)
        }
    }
```

```
    private fun showData() {
        val circularProgressDrawable =
CircularProgressDrawable(this)
        circularProgressDrawable.strokeWidth = 5f
        circularProgressDrawable.centerRadius = 30f
        circularProgressDrawable.start()
        if (data != null) {
            tv_nama_sertifikasi.text =
data!!.nama_sertifikasi
            tv_nomor.text = data!!.nomor
            tv_nama.text = data!!.nama
            tv_tanggal.text = data!!.tanggal
            tv_tanggal_selesai.text =
data!!.tanggal_berakhir
            tv_dept.text =
Helper.getDepartemen(data!!.departemen)
            Glide.with(this)
                .load(data!!.filepath)

.placeholder(circularProgressDrawable)
                .into(iv_bukti)
        }
    }
```

```
    fun deleteData(progressBar: ProgressBar?, token:
String, id: Int, context: Context) {
        progressBar?.visibility = View.VISIBLE
        ApiService.endpoint.deleteSertifikasi(
            token, id
        ).enqueue(object : Callback<ResponseBody> {
            override fun onResponse(
                call: Call<ResponseBody>,
                response: Response<ResponseBody>
            ) {
                if (response.isSuccessful) {
                    Toast.makeText(
                        context,
                        "Success:
${response.message()}",
                        Toast.LENGTH_SHORT
                    ).show()
                }
                else Toast.makeText(
                    context,
                    "Failed: ${response.message()}",
                    Toast.LENGTH_SHORT
                ).show()

                progressBar?.visibility = View.GONE
            }
            override fun onFailure(call:
Call<ResponseBody>, t: Throwable) {
                Toast.makeText(
                    context,
                    "Failed: ${t.localizedMessage}",
                    Toast.LENGTH_SHORT
                ).show()
                progressBar?.visibility = View.GONE
            }
        })
    }
```

```
    fun validateData(progressBar: ProgressBar?,
token: String, id: Int, context: Context) {
        progressBar?.visibility = View.VISIBLE
        ApiService.endpoint.validateSertifikasi(
            token, id
        ).enqueue(object : Callback<ResponseBody> {
            override fun onResponse(
                call: Call<ResponseBody>,
                response: Response<ResponseBody>
            ) {
                if (response.isSuccessful) {
                    Toast.makeText(
                        context,
                        "Success:
${response.message()}",
                        Toast.LENGTH_SHORT
                    ).show()
                }
                else Toast.makeText(
                    context,
                    "Failed: ${response.message()}",
                    Toast.LENGTH_SHORT
                ).show()
                progressBar?.visibility = View.GONE
            }
            override fun onFailure(call:
Call<ResponseBody>, t: Throwable) {
                Toast.makeText(
                    context,
                    "Failed: ${t.localizedMessage}",
                    Toast.LENGTH_SHORT
                ).show()
                progressBar?.visibility = View.GONE
            }
        })
    }
}
```

Kode Sumber 4.28: Kode Sumber Halaman Detail Sertifikasi

4.3 Implementasi Android Component

4.3.1. Komponen Adapter Prestasi

Merupakan class yang menjembatani antara *UI component* (halaman prestasi) dengan *data source*. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.29 dan 4.30.

```
import android.annotation.SuppressLint
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Filter
import android.widget.Filterable
import androidx.recyclerview.widget.RecyclerView
import com.kpbois.fteicmobile.R
import
com.kpbois.fteicmobile.retrofit.PrestasiDosenModel
import
kotlinx.android.synthetic.main.item_prestasi.view.*
import java.util.*
import kotlin.collections.ArrayList
```

Kode Sumber 4.29: Dependensi Komponen Adapter Prestasi

```
class AdapterPrestasi () :
RecyclerView.Adapter<AdapterPrestasi.PrestasiHolder>
(), Filterable {

    private lateinit var onClickListener:
OnClickListener
    private var prestasi =
ArrayList<PrestasiDosenModel>()
    private var filterPrestasi =
ArrayList<PrestasiDosenModel>()

    inner class PrestasiHolder(itemView: View)
:RecyclerView.ViewHolder(itemView) {
        @SuppressLint("SetTextI18n")
        fun bind(item: PrestasiDosenModel) {
            with(itemView) {
                tv_tahun.text =
getYear(item.tanggal)
                tv_judul.text =
"${item.nama_penghargaan} ${item.jenis_penghargaan}"
                tv_nama.text = item.nama
                tv_peringkat_prestasi.text =
item.capaian

                cv_item_prestasi.setOnClickListener
{
onClickListener.onClick(filterPrestasi[absoluteAdapt
erPosition]) }
            }
        }
    }

    fun setOnClickListener(
onClickListener: OnClickListener) {
        this.onClickListener = onClickListener
    }
```

```
    override fun onCreateViewHolder(parent:
ViewGroup, viewType: Int): PrestasiHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout
.item_prestasi, parent, false)
        return PrestasiHolder(view)
    }

    fun setData(items: List<PrestasiDosenModel>) {
        prestasi.clear()
        prestasi.addAll(items)
        filterPrestasi.clear()
        filterPrestasi.addAll(items)
        notifyDataSetChanged()

    }

    override fun getItemCount(): Int {
        return filterPrestasi.size
    }

    @SuppressLint("SetTextI18n")
    override fun onBindViewHolder(holder:
PrestasiHolder, position: Int) {
        holder.bind(filterPrestasi[position])
    }

    interface OnClickListener {
        fun onClick(data: PrestasiDosenModel)
    }
```

```
    override fun getFilter(): Filter {
        return object : Filter() {
            override fun
performFiltering(constraint: CharSequence?):
FilterResults {
                val query =
constraint.toString().toLowerCase(Locale.ROOT)
                val result = FilterResults()
                if (query.isEmpty()) result.values =
prestasi
                else {
                    val list =
ArrayList<PrestasiDosenModel>()
                    for (item in prestasi) {
                        if
(item.nama_penghargaan.toLowerCase(Locale.ROOT).cont
ains(query) ||
item.nama.toLowerCase(Locale.ROOT).contains(query)
||
item.jenis_penghargaan.toLowerCase(Locale.ROOT).cont
ains(query)) {
                            list.add(item)
                        }
                    }
                    result.values = list
                }
                return result
            }
            override fun publishResults(constraint:
CharSequence?, results: FilterResults?) {
                filterPrestasi.clear()

filterPrestasi.addAll(results?.values as
List<PrestasiDosenModel>)
                notifyDataSetChanged()
            }

        }
    }

    private fun getYear(tanggal: String): String {
        return tanggal.substring(0..3)
    }
}
```

Kode Sumber 4.30: Kode Sumber Komponen Adapter Prestasi

4.3.2. Komponen Adapter Konferensi

Merupakan class yang menjembatani antara *UI component* (halaman konferensi) dengan *data source*. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.31 dan 4.32.

```
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Filter
import android.widget.Filterable
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.kpbois.fteicmobile.R
import java.util.*
import kotlin.collections.ArrayList
```

Kode Sumber 4.31: Dependensi Komponen Adapter Konferensi

```
class AdapterKonferensi() :
RecyclerView.Adapter<AdapterKonferensi.KonferensiHol
der>(), Filterable {

    private lateinit var onClickListener:
OnClickListener
    private var konferensi =
ArrayList<ModelKonferensi>()
    private var filterKonferensi =
ArrayList<ModelKonferensi>()

    inner class KonferensiHolder (itemView : View) :
RecyclerView.ViewHolder(itemView) {
        var tahun: TextView =
itemView.findViewById(R.id.tv_tahun)
        var judul: TextView =
itemView.findViewById(R.id.tv_judul)
        var author: TextView =
itemView.findViewById(R.id.tv_nama)
    }

    fun setOnClickListener(onClickListener:
OnClickListener) {
        this.onClickListener = onClickListener
    }

    fun setData(items: List<ModelKonferensi>) {
        konferensi.clear()
        konferensi.addAll(items)
        filterKonferensi.clear()
        filterKonferensi.addAll(items)
        notifyDataSetChanged()
    }
```

```
    override fun onCreateViewHolder(parent:
ViewGroup, viewType: Int): KonferensiHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout
.item,parent,false)
        return KonferensiHolder(view)
    }

    override fun getItemCount(): Int {
        return filterKonferensi.size
    }

    override fun onBindViewHolder(holder:
KonferensiHolder, position: Int) {
        val item = filterKonferensi[position]
        holder.tahun.text = item.tahun
        holder.judul.text = item.judul
        holder.author.text = item.author

        holder.itemView.setOnClickListener {

onClickListener.onClick(filterKonferensi[holder.abso
luteAdapterPosition])
        }
    }

    interface OnClickListener{
        fun onClick(data: ModelKonferensi)
    }
```

```
    override fun getFilter(): Filter {
        return object: Filter() {
            override fun
performFiltering(constraint: CharSequence?):
FilterResults {
                val query =
constraint.toString().toLowerCase(Locale.ROOT)
                val result = FilterResults()
                if (query.isEmpty()) result.values =
konferensi
                else {
                    val list =
ArrayList<ModelKonferensi>()
                    for (item in  konferensi) {
                        if
(item.judul.toLowerCase(Locale.ROOT).contains(query)
||
item.author.toLowerCase(Locale.ROOT).contains(query)
) {
                            list.add(item)
                        }
                    }
                    result.values = list
                }
                return result
            }

            override fun publishResults(constraint:
CharSequence?, results: FilterResults?) {
                filterKonferensi.clear()
```

Kode Sumber 4.32: Kode Sumber Komponen Adapter Konferensi

4.3.3. Komponen Adapter Jurnal

Merupakan class yang menjembatani antara *UI component* (halaman jurnal) dengan *data source*. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.33 dan 4.34.

```
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Filter
import android.widget.Filterable
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.kpbois.fteicmobile.R
import java.util.*
import kotlin.collections.ArrayList
```

Kode Sumber 4.33: Dependensi Komponen Adapter Jurnal

```
class AdapterJurnal () :
RecyclerView.Adapter<AdapterJurnal.JurnalHolder>(),
Filterable {

    private lateinit var onClickListener:
OnClickListener
    private var jurnal = ArrayList<ModelJurnal>()
    private val filterJurnal =
ArrayList<ModelJurnal>()

    inner class JurnalHolder(itemView: View):
RecyclerView.ViewHolder(itemView) {
        var tahun: TextView =
itemView.findViewById(R.id.tv_tahun)
        var judul: TextView =
itemView.findViewById(R.id.tv_judul)
        var author: TextView =
itemView.findViewById(R.id.tv_nama)
    }

    fun setOnClickListener(onClickListener:
OnClickListener){
        this.onClickListener = onClickListener
    }

    fun setData(items: List<ModelJurnal>) {
        jurnal.clear()
        jurnal.addAll(items)
        filterJurnal.clear()
        filterJurnal.addAll(items)
        notifyDataSetChanged()
    }
```

```
    override fun onCreateViewHolder(parent:
ViewGroup, viewType: Int): JurnalHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout
.item, parent, false)
        return JurnalHolder(view)
    }

    override fun getItemCount(): Int {
        return filterJurnal.size
    }

    override fun onBindViewHolder(holder:
JurnalHolder, position: Int) {
        val item = filterJurnal[position]
        holder.tahun.text = item.tahun
        holder.judul.text = item.judul
        holder.author.text = item.author

        holder.itemView.setOnClickListener {

onClickListener.onClick(filterJurnal[holder.absolute
AdapterPosition])
        }
    }

    interface OnClickListener{
        fun onClick(data: ModelJurnal)
    }
```

```
    override fun getFilter(): Filter {
        return object : Filter() {
            override fun
performFiltering(constraint: CharSequence?):
FilterResults {
                val query =
constraint.toString().toLowerCase(Locale.ROOT)
                val result = FilterResults()
                if (query.isEmpty()) result.values =
jurnal
                else{
                    val list =
ArrayList<ModelJurnal>()
                    for (item in jurnal) {
                        if
(item.judul.toLowerCase(Locale.ROOT).contains(query)
||
item.author.toLowerCase(Locale.ROOT).contains(query)
) {
                            list.add(item)
                        }
                    }
                    result.values = list
                }
                return result
            }

            override fun publishResults(constraint:
CharSequence?, results: FilterResults?) {
                filterJurnal.clear()
                filterJurnal.addAll(results?.values
as List<ModelJurnal>)
                notifyDataSetChanged()
            }

        }
    }
}
```

Kode Sumber 4.34: Kode Sumber Komponen Adapter Jurnal

4.3.4. Komponen Adapter Submission

Komponen DatePickerForm berisi form untuk melakukan render pada pemilihan tanggal. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.35 dan 4.36.

```
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Filter
import android.widget.Filterable
import androidx.recyclerview.widget.RecyclerView
import com.kpbois.fteicmobile.R
import kotlinx.android.synthetic.main.item.view.*
import java.util.*
import kotlin.collections.ArrayList
```

Kode Sumber 4.35: Dependensi Komponen Adapter Submission

```
class AdapterSubmission() :
RecyclerView.Adapter<AdapterSubmission.ViewHolder>()
, Filterable {

    private var submission =
ArrayList<ModelSubmission>()
    private var filterSubmission =
ArrayList<ModelSubmission>()
    private lateinit var onClickListener:
OnClickListener

    fun setOnClickListener(onClickListener:
OnClickListener) {
        this.onClickListener = onClickListener
    }

    inner class ViewHolder(itemView: View):
RecyclerView.ViewHolder(itemView) {
        fun bind(item: ModelSubmission) {
            with(itemView) {
                tv_tahun.text = item.modul
                tv_judul.text = item.judul
                tv_nama.text = item.nama

                cv_item.setOnClickListener {

onClickListener.onItemClick(item)
                }
            }
        }
    }
```

```
    fun setData(items: List<ModelSubmission>) {
        submission.clear()
        filterSubmission.clear()
        submission.addAll(items)
        filterSubmission.addAll(items)
        notifyDataSetChanged()
    }

    override fun onCreateViewHolder(parent:
ViewGroup, viewType: Int): ViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout
.item, parent, false)
        return ViewHolder(view)
    }

    override fun onBindViewHolder(holder:
ViewHolder, position: Int) {
        holder.bind(submission[position])
    }

    override fun getItemCount(): Int {
        return filterSubmission.size
    }

    interface OnClickListener {
        fun onItemClick(data: ModelSubmission)
    }
```

```
    override fun getFilter(): Filter {
        return object : Filter() {
            override fun
performFiltering(constraint: CharSequence?):
FilterResults {
                val query =
constraint.toString().toLowerCase(Locale.ROOT)
                val result = FilterResults()
                if (query.isEmpty()) result.values =
submission
                else {
                    val list =
ArrayList<ModelSubmission>()
                    for (item in submission) {
                        if
(item.judul.toLowerCase(Locale.ROOT).contains(query)
||
item.nama.toLowerCase(Locale.ROOT).contains(query)
||
item.modul.toLowerCase(Locale.ROOT).contains(query))
{
                            list.add(item)
                        }
                    }
                    result.values = list
                }
                return result
            }

            override fun publishResults(constraint:
CharSequence?, results: FilterResults?) {
                filterSubmission.clear()

filterSubmission.addAll(results?.values as
List<ModelSubmission>)
                notifyDataSetChanged()
            }

        }
    }
}
```

Kode Sumber 4.36: Kode Sumber Komponen Adapter Submission

4.3.5. Komponen Adapter Kuliah Tamu

Merupakan class yang menjembatani antara *UI component* (halaman kuliah tamu) dengan *data source*. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.37 dan 4.38.

```
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Filter
import android.widget.Filterable
import androidx.recyclerview.widget.RecyclerView
import com.kpbois.fteicmobile.R
import kotlinx.android.synthetic.main.item.view.*
```

Kode Sumber 4.37: Dependensi Komponen Adapter Kuliah Tamu

```
class AdapterKultam() :
RecyclerView.Adapter<AdapterKultam.ViewHolder>(),
Filterable {

    lateinit var onClickListener: OnClickListener
    private val kultamList =
ArrayList<ModelKultam>()
    private val filteredList =
ArrayList<ModelKultam>()

    override fun onCreateViewHolder(parent:
ViewGroup, viewType: Int): ViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout
.item, parent, false)
        return ViewHolder(view)
    }

    override fun getItemCount(): Int =
filteredList.size

    override fun onBindViewHolder(holder:
ViewHolder, position: Int) {
        holder.bind(filteredList[position])
    }

    inner class ViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        fun bind(data: ModelKultam) {
            with (itemView) {
                tv_judul.text = data.topik
                tv_nama.text = data.institusi
                tv_tahun.text = data.tanggal.take(4)
                cv_item.setOnClickListener {
onClickListener.onClick(data) }
            }
        }
    }

    interface OnClickListener {
        fun onClick(data: ModelKultam)
    }
```

```
fun setData(list: ArrayList<ModelKultam>) {
        kultamList.clear()
        kultamList.addAll(list)
        filteredList.clear()
        filteredList.addAll(list)
        notifyDataSetChanged()
    }

    fun getData(): ArrayList<ModelKultam> =
kultamList

    override fun getFilter(): Filter {
        return object : Filter() {
            override fun performFiltering(p0:
CharSequence?): FilterResults {
                val query =
p0.toString().toLowerCase()
                val result = FilterResults()
                if (query.isNullOrEmpty())
result.values = kultamList
                else {
                    val list =
ArrayList<ModelKultam>()
                    for (kultam in kultamList) {
                        if
(kultam.topik.toLowerCase().contains(query))
                            list.add(kultam)
                    }
                    result.values = list
                }
                return result
            }

            override fun publishResults(p0:
CharSequence?, p1: FilterResults?) {
                filteredList.clear()
                filteredList.addAll(p1?.values as
ArrayList<ModelKultam>)
                notifyDataSetChanged()
            }
        }
    }
}
```

Kode Sumber 4.38: Kode Sumber Komponen Adapter Kuliah Tamu

4.3.6. Komponen Adapter Training

Merupakan class yang menjembatani antara *UI component* (halaman training) dengan *data source*. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.39 dan 4.40.

```
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Filter
import android.widget.Filterable
import androidx.recyclerview.widget.RecyclerView
import com.kpbois.fteicmobile.R
import kotlinx.android.synthetic.main.item.view.*
```

Kode Sumber 4.39: Dependensi Komponen Adapter Training

```
class AdapterTraining() :
RecyclerView.Adapter<AdapterTraining.ViewHolder>(),
Filterable {

    lateinit var onClickListener: OnClickListener
    private val trainingList =
ArrayList<ModelTraining>()
    private val filteredList =
ArrayList<ModelTraining>()

    override fun onCreateViewHolder(parent:
ViewGroup, viewType: Int): ViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout
.item, parent, false)
        return ViewHolder(view)
    }

    override fun getItemCount(): Int =
filteredList.size

    override fun onBindViewHolder(holder:
ViewHolder, position: Int) {
        holder.bind(filteredList[position])
    }
```

```
inner class ViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        fun bind(data: ModelTraining) {
            with (itemView) {
                tv_judul.text = data.judul
                tv_nama.text = data.peserta
                tv_tahun.text =
data.date_start.take(4)
                cv_item.setOnClickListener {
onClickListener.onClick(data) }
            }
        }
    }

    interface OnClickListener {
        fun onClick(data: ModelTraining)
    }

    fun setData(list: ArrayList<ModelTraining>) {
        trainingList.clear()
        trainingList.addAll(list)
        filteredList.clear()
        filteredList.addAll(list)
        notifyDataSetChanged()
    }

    fun getData(): ArrayList<ModelTraining> =
trainingList
```

```
override fun getFilter(): Filter {
        return object : Filter() {
            override fun performFiltering(p0:
CharSequence?): FilterResults {
                val query =
p0.toString().toLowerCase()
                val result = FilterResults()
                if (query.isNullOrEmpty())
result.values = trainingList
                else {
                    val list =
ArrayList<ModelTraining>()
                    for (training in trainingList) {
                        if
(training.judul.toLowerCase().contains(query))
                            list.add(training)
                    }
                    result.values = list
                }
                return result
            }

            override fun publishResults(p0:
CharSequence?, p1: FilterResults?) {
                filteredList.clear()
                filteredList.addAll(p1?.values as
ArrayList<ModelTraining>)
                notifyDataSetChanged()
            }
        }
    }
}
```

Kode Sumber 4.40: Kode Sumber Komponen Adapter Training

4.3.7. Komponen Adapter Sertifikasi

Merupakan class yang menjembatani antara *UI component* (halaman sertifikasi) dengan *data source*. Dependensi dan Kode Sumber dapat dilihat pada Kode Sumber 4.41 dan 4.42.

```
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Filter
import android.widget.Filterable
import androidx.recyclerview.widget.RecyclerView
import com.kpbois.fteicmobile.R
import kotlinx.android.synthetic.main.item.view.*
```

Kode Sumber 4.41: Dependensi Komponen Adapter Sertifikasi

```
class AdapterSertifikasi:
RecyclerView.Adapter<AdapterSertifikasi.ViewHolder>(
), Filterable {

    lateinit var onClickListener: OnClickListener
    private val sertifikasiList =
ArrayList<ModelSertifikasi>()
    private val filteredList =
ArrayList<ModelSertifikasi>()

    override fun onCreateViewHolder(parent:
ViewGroup, viewType: Int): ViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout
.item, parent, false)
        return ViewHolder(view)
    }

    override fun getItemCount(): Int =
filteredList.size

    override fun onBindViewHolder(holder:
ViewHolder, position: Int) {
        holder.bind(filteredList[position])
    }
```

```
inner class ViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        fun bind(data: ModelSertifikasi) {
            with(itemView) {
                tv_judul.text =
data.nama_sertifikasi
                tv_nama.text = data.nama
                tv_tahun.text = data.tanggal.take(4)
                cv_item.setOnClickListener {
onClickListener.onClick(data) }
            }
        }
    }

    interface OnClickListener {
        fun onClick(data: ModelSertifikasi)
    }

    fun setData(list: ArrayList<ModelSertifikasi>) {
        sertifikasiList.clear()
        sertifikasiList.addAll(list)
        filteredList.clear()
        filteredList.addAll(list)
        notifyDataSetChanged()
    }

fun getData(): ArrayList<ModelSertifikasi> =
sertifikasiList
```

```
override fun getFilter(): Filter {
        return object : Filter() {
            override fun performFiltering(p0:
CharSequence?): FilterResults {
                val query =
p0.toString().toLowerCase()
                val result = FilterResults()
                if (query.isNullOrEmpty())
result.values = sertifikasiList
                else {
                    val list =
ArrayList<ModelSertifikasi>()
                    for (sertifikasi in
sertifikasiList) {
                        if
(sertifikasi.nama.toLowerCase().contains(query) ||
sertifikasi.nama_sertifikasi.toLowerCase().contains(
query))
                            list.add(sertifikasi)
                    }
                    result.values = list
                }
                return result
            }

            override fun publishResults(p0:
CharSequence?, p1: FilterResults?) {
                filteredList.clear()
                filteredList.addAll(p1?.values as
ArrayList<ModelSertifikasi>)
                notifyDataSetChanged()
            }
        }
    }
}
```

Kode Sumber 4.42: Kode Sumber Komponen Adapter Sertifikasi

BAB V
PENGUJIAN DAN EVALUASI

5.1. Skenario Pengujian

5.1.1. Login Pengguna

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka aplikasi pada smartphone.
- Memasukkan *username* dan *password* lalu klik tombol LOGIN
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman dashboard yang berisi tombol kuliah tamu, konferensi / jurnal, prestasi dosen, training, dan sertifikasi.

5.1.2. Logout Pengguna

- Skenario pengujian aplikasi adalah sebagai berikut:
- Membuka halaman dashboard pada aplikasi.
- Menekan tombol dengan symbol titik tiga di bagian kanan atas.
- Menekan tombol *logout* di *dropdown*.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman login.

5.1.3. Membuka Halaman Kuliah Tamu

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada aplikasi.
- Menekan tombol Kuliah Tamu.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman kuliah tamu yang berisi daftar kuliah tamu, *input* untuk pencarian, *filter* pencarian, dan tombol tambah.

5.1.4. Membuka Halaman Detail Kuliah Tamu

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada aplikasi.

- Menekan tombol Kuliah Tamu.
- Menekan salah satu daftar kuliah tamu.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman detail kuliah tamu yang berisi judul kegiatan, pemateri, institusi, departemen, tingkat, tanggal, URL, gambar, tombol hapus, dan tombol validasi jika kuliah tamu belum divalidasi.

5.1.5. Menghapus Kuliah Tamu

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada aplikasi.
- Menekan tombol Kuliah Tamu.
- Menekan salah satu daftar kuliah tamu.
- Menekan tombol hapus.
- Menekan tombol YA pada *modal* yang muncul.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu muncul pemberitahuan bahwa data sudah terhapus, dan data sudah tidak ada di halaman kuliah tamu.

5.1.6. Memvalidasi Kuliah Tamu

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada aplikasi.
- Menekan tombol Kuliah Tamu.
- Menekan salah satu daftar kuliah tamu.
- Menekan tombol validasi.
- Menekan tombol YA pada *modal* yang muncul.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu muncul pemberitahuan bahwa data sudah tervalidasi, dan tombol validasi hilang dari halaman detail kuliah tamu

5.1.7. Membuka Halaman Konferensi / Jurnal

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada *smartphone*.
- Menekan tombol Konferensi / Jurnal.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman konferensi / jurnal yang berisi daftar konferensi / jurnal, *input* untuk pencarian, dan *filter* pencarian.

5.1.8. Membuka Halaman Detail Konferensi / Jurnal

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada *smartphone*.
- Menekan tombol Konferensi / Jurnal.
- Menekan salah satu daftar konferensi / jurnal.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman detail konferensi / jurnal yang berisi judul kegiatan, departemen, URL, konf hal, *check box* scopus, tanggal selesai, author, dipublish di, tingkat, tempat, tanggal mulai, tombol hapus, dan tombol validasi jika konferensi / jurnal belum divalidasi.

5.1.9. Menghapus Konferensi / Jurnal

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada *smartphone*.
- Menekan tombol konferensi / jurnal.
- Menekan salah satu daftar konferensi / jurnal.
- Menekan tombol hapus.
- Menekan tombol YA pada *dialog* yang muncul.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu muncul pemberitahuan bahwa data sudah terhapus, dan data sudah tidak ada di halaman konferensi / jurnal.

5.1.10. Memvalidasi Konferensi / Jurnal

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada *smartphone*.

- Menekan tombol konferensi / jurnal.
- Menekan salah satu daftar konferensi / jurnal.
- Menekan tombol validasi.
- Menekan tombol YA pada *dialog* yang muncul.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu muncul pemberitahuan bahwa data sudah tervalidasi, dan tombol validasi hilang dari halaman detail konferensi / jurnal.

5.1.11. Membuka Halaman Prestasi

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada *smartphone*.
- Menekan tombol prestasi.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman prestasi yang berisi daftar prestasi, *input* untuk pencarian, dan *filter* pencarian.

5.1.12. Membuka Halaman Detail Prestasi

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada *smartphone*.
- Menekan tombol prestasi.
- Menekan salah satu daftar prestasi.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman detail prestasi yang berisi nama penghargaan, pelaku, kategori peserta. lembaga penyelenggara, capaian, URL, NIP, departemen, jenis prestasi, tingkat, tanggal, tombol hapus, dan tombol validasi jika prestasi belum divalidasi.

5.1.13. Menghapus Prestasi

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada *smartphone*.
- Menekan tombol prestasi.

- Menekan salah satu daftar prestasi.
- Menekan tombol hapus.
- Menekan tombol YA pada *dialog* yang muncul.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu muncul pemberitahuan bahwa data sudah terhapus, dan data sudah tidak ada di halaman prestasi.

5.1.14. Memvalidasi Prestasi

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada *smartphone*.
- Menekan tombol prestasi.
- Menekan salah satu daftar prestasi.
- Menekan tombol validasi.
- Menekan tombol YA pada *dialog* yang muncul.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu muncul pemberitahuan bahwa data sudah tervalidasi, dan tombol validasi hilang dari halaman detail prestasi.

5.1.15. Membuka Halaman Training

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka dashboard pada smartphone.
- Menekan tombol training.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman training yang berisi daftar training, *input* untuk pencarian, *filter* pencarian, dan tombol tambah.

5.1.16. Membuka Halaman Detail Training

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka dashboard pada smartphone.
- Menekan tombol training.
- Menekan salah satu daftar prestasi training.

- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman detail training yang berisi judul kegiatan, pelaku, tempat, tanggal selesai, jenis, departemen, tanggal mulai, gambar, tombol hapus, dan tombol validasi jika training belum divalidasi.

5.1.17. Menghapus Training

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka dashboard pada smartphone.
- Menekan tombol training.
- Menekan salah satu daftar training.
- Menekan tombol hapus.
- Menekan tombol YA pada *modal* yang muncul.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu muncul pemberitahuan bahwa data sudah terhapus, dan data sudah tidak ada di halaman training.

5.1.18. Memvalidasi Training

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka dashboard pada smartphone.
- Menekan tombol training.
- Menekan salah satu daftar training.
- Menekan tombol validasi.
- Menekan tombol YA pada *modal* yang muncul.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu muncul pemberitahuan bahwa data sudah tervalidasi, dan tombol validasi hilang dari halaman detail training.

5.1.19. Menambah Daftar Training

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka /dashboard pada browser.
- Menekan salah satu tombol training.

- Menekan tombol TAMBAH.
- Mengisi *form*.
- Menekan tombol SUBMIT.
- Menekan tombol YA pada *modal* yang muncul.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat pemberitahuan data berhasil di *entry*, dan data training terdapat pada halaman prestasi dosen.

5.1.20. Membuka Halaman Sertifikasi

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka dashboard pada smartphone.
- Menekan tombol sertifikasi.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman training yang berisi daftar training, *input* untuk pencarian, *filter* pencarian, dan tombol tambah.

5.1.21. Membuka Halaman Detail Sertifikasi

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka dashboard pada smartphone.
- Menekan tombol sertifikasi.
- Menekan salah satu daftar prestasi sertifikasi.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman detail sertifikasi yang berisi nama sertifikasi, nama peserta, nomor, valid sampai, lembaga sertifikasi, departemen, tanggal dikeluarkan, gambar, tombol hapus, dan tombol validasi jika sertifikasi belum divalidasi.

5.1.22. Menghapus Sertifikasi

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka dashboard pada smartphone.
- Menekan tombol sertifikasi.

- Menekan salah satu daftar sertifikasi.
- Menekan tombol hapus.
- Menekan tombol YA pada *modal* yang muncul.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu muncul pemberitahuan bahwa data sudah terhapus, dan data sudah tidak ada di halaman sertifikasi.

5.1.23. Memvalidasi Sertifikasi

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka dashboard pada smartphone.
- Menekan tombol sertifikasi.
- Menekan salah satu daftar sertifikasi.
- Menekan tombol validasi.
- Menekan tombol YA pada *modal* yang muncul.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu muncul pemberitahuan bahwa data sudah tervalidasi, dan tombol validasi hilang dari halaman detail sertifikasi.

5.1.24. Menambah Daftar Sertifikasi

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka dashboard pada smartphone.
- Menekan salah satu tombol sertifikasi.
- Menekan tombol TAMBAH.
- Mengisi *form*.
- Menekan tombol SUBMIT.
- Menekan tombol YA pada *modal* yang muncul.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat pemberitahuan data berhasil di *entry*, dan data sertifikasi terdapat pada halaman sertifikasi.

5.1.25. Mengubah Kata Sandi

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka dashboard pada smartphone.
- Menekan tombol titik tiga di kanan atas.
- Menekan tombol Change Password.
- Mengisi password lama, password baru, dan konfirmasi password.
- Menekan tombol simpan.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat pemberitahuan data berhasil disimpan.

5.1.26. Membuka Halaman Submission

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada *smartphone*.
- Menekan tombol menu di kanan atas.
- Menekan tombol Submission.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman submission yang berisi data-data submission.

5.1.27. Membuka Halaman Detail Submission

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada *smartphone*.
- Menekan tombol menu di kanan atas.
- Menekan tombol Submission.
- Menekan salah satu daftar prestasi sertifikasi.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman detail submission yang berisi data-data yang sesuai dengan jenis submission

5.1.28. Membuka Halaman Download

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada *smartphone*.
- Menekan tombol menu di kanan atas.
- Menekan tombol Download.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu melihat halaman download yang berisi dropdown jenis, departemen, tahun, dan tombol download.

5.1.29. Download Data

Skenario pengujian aplikasi adalah sebagai berikut:

- Membuka halaman dashboard pada *smartphone*.
- Menekan tombol menu di kanan atas.
- Menekan tombol Download.
- Memilih jenis, departemen, dan tahun.
- Menekan tombol DOWNLOAD.
- Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu data yang dipilih terunduh dalam format file xlsx.

5.2. Evaluasi Pengujian

Pada subbab ini akan diberikan hasil evaluasi dari pengujian-pengujian yang telah dilakukan. Hasil evaluasi pengujian dapat dilihat pada Tabel 5.1.

Tabel 5.1: Tabel evaluasi pengujian aplikasi sesuai kebutuhan

No.	Kebutuhan	Uji Coba	Status
UC001	Login Pengguna	Pengguna memasukkan username dan password pada halaman login. Pengguna dapat melihat halaman dashboard	Berhasil

UC002	Logout Pengguna	Pengguna menekan tombol profile kemudian menekan logout. Pengguna melihat halaman login dan tidak bisa mengakses dashboard.	Berhasil
UC003	Membuka Halaman Kuliah Tamu	Pengguna membuka halaman dashboard. Pengguna menekan tombol Kuliah Tamu. Pengguna melihat halaman kuliah tamu.	Berhasil
UC004	Membuka Halaman Detail Kuliah Tamu	Pengguna membuka halaman dashboard. Pengguna menekan tombol Kuliah Tamu. Pengguna menekan salah satu data kuliah tamu. Pengguna melihat halaman detail kuliah tamu.	Berhasil
UC005	Menghapus Kuliah Tamu	Pengguna membuka halaman dashboard. Pengguna menekan tombol Kuliah Tamu. Pengguna menekan salah satu data kuliah tamu. Pengguna menekan tombol hapus. Pengguna menekan tombol ya pada modal. Data kuliah tamu terhapus.	Berhasil
UC006	Memvalidasi Kuliah Tamu	Pengguna membuka halaman dashboard. Pengguna menekan tombol Kuliah Tamu. Pengguna menekan salah satu data kuliah tamu. Pengguna menekan tombol validasi. Pengguna menekan tombol ya pada modal. Data kuliah tamu tervalidasi, tombol validasi hilang.	Berhasil
UC007	Membuka Halaman	Pengguna membuka halaman dashboard. Pengguna menekan	Berhasil

	Konferensi / Jurnal	tombol Konferensi / Jurnal. Pengguna melihat halaman konferensi / jurnal.	
UC008	Membuka Halaman Detail Konferensi / Jurnal	Pengguna membuka halaman dashboard. Pengguna menekan tombol Konferensi / Jurnal Pengguna menekan salah satu data konferensi / jurnal. Pengguna melihat halaman detail konferensi / jurnal.	Berhasil
UC009	Menghapus Konferensi / Jurnal	Pengguna membuka halaman dashboard. Pengguna menekan tombol Konferensi / Jurnal. Pengguna menekan salah satu data konferensi / jurnal. Pengguna menekan tombol hapus. Pengguna menekan tombol ya pada dialog. Data konferensi / jurnal terhapus.	Berhasil
UC010	Memvalidasi Konferensi / Jurnal	Pengguna membuka halaman dashboard. Pengguna menekan tombol Konferensi / Jurnal. Pengguna menekan salah satu data konferensi / jurnal. Pengguna menekan tombol validasi. Pengguna menekan tombol ya pada dialog. Data konferensi / jurnal tervalidasi, tombol validasi hilang.	Berhasil
UC011	Membuka Halaman Prestasi	Pengguna membuka halaman dashboard. Pengguna menekan tombol Prestasi Dosen. Pengguna melihat halaman prestasi dosen.	Berhasil

UC012	Membuka Halaman Detail Prestasi	Pengguna membuka halaman dashboard. Pengguna menekan tombol Prestasi. Pengguna menekan salah satu data prestasi. Pengguna melihat halaman detail prestasi.	Berhasil
UC013	Menghapus Prestasi	Pengguna membuka halaman dashboard. Pengguna menekan tombol Prestasi. Pengguna menekan salah satu data prestasi. Pengguna menekan tombol hapus. Pengguna menekan tombol ya pada dialog. Data prestasi terhapus.	Berhasil
UC014	Memvalidasi Prestasi	Pengguna membuka halaman dashboard. Pengguna menekan tombol Prestasi. Pengguna menekan salah satu data prestasi. Pengguna menekan tombol validasi. Pengguna menekan tombol ya pada dialog. Data prestasi tervalidasi, tombol validasi hilang.	Berhasil
UC015	Membuka Halaman Training	Pengguna membuka halaman dashboard. Pengguna menekan tombol Training. Pengguna melihat halaman training.	Berhasil
UC016	Membuka Halaman Detail Training	Pengguna membuka halaman dashboard. Pengguna menekan tombol Training. Pengguna menekan salah satu data sertifikasi Pengguna melihat halaman detail training.	Berhasil
UC017	Menghapus Training	Pengguna membuka halaman dashboard. Pengguna menekan tombol Training. Pengguna	Berhasil

		menekan salah satu data training. Pengguna menekan tombol hapus. Pengguna menekan tombol ya pada modal. Data training terhapus.	
UC018	Memvalidasi Training	Pengguna membuka halaman dashboard. Pengguna menekan tombol Training. Pengguna menekan salah satu data training. Pengguna menekan tombol validasi. Pengguna menekan tombol ya pada modal. Data training tervalidasi, tombol validasi hilang.	Berhasil
UC019	Membuka Halaman Sertifikasi	Pengguna membuka halaman dashboard. Pengguna menekan tombol Sertifikasi. Pengguna melihat halaman sertifikasi.	Berhasil
UC020	Membuka Halaman Detail Sertifikasi	Pengguna membuka halaman dashboard. Pengguna menekan tombol Sertifikasi. Pengguna menekan salah satu sertifikasi. Pengguna melihat halaman detail sertifikasi.	Berhasil
UC020	Menghapus Sertifikasi	Pengguna membuka halaman dashboard. Pengguna menekan tombol Sertifikasi. Pengguna menekan salah satu data sertifikasi. Pengguna menekan tombol hapus. Pengguna menekan tombol ya pada modal. Data sertifikasi terhapus.	Berhasil
UC021	Memvalidasi Sertifikasi	Pengguna membuka halaman dashboard. Pengguna menekan tombol Sertifikasi. Pengguna	Berhasil

		menekan salah satu data sertifikasi. Pengguna menekan tombol validasi. Pengguna menekan tombol ya pada modal. Data sertifikasi tervalidasi, tombol validasi hilang.	
UC022	Membuka Halaman Account	Pengguna membuka halaman dashboard. Pengguna menekan tombol profile. Pengguna menekan tombol Account. Pengguna melihat halaman Account.	Berhasil
UC023	Mengubah Kata Sandi	Pengguna membuka halaman dashboard. Pengguna menekan tombol titik tiga. Pengguna menekan tombol Change Password. Pengguna mengisi password lama dan password baru. Pengguna menekan tombol simpan. Pengguna melihat pemberitahuan data berhasil disimpan.	Berhasil
UC024	Membuka Halaman Submission	Pengguna membuka halaman dashboard. Pengguna menekan tombol menu. Pengguna menekan tombol Submission. Pengguna melihat halaman Submission.	Berhasil
UC025	Membuka Halaman Detail Submission	Pengguna membuka halaman dashboard. Pengguna menekan tombol menu. Pengguna menekan tombol Submission. Pengguna menekan salah satu submission. Pengguna melihat halaman detail submission.	Berhasil

UC026	Membuka Halaman Download	Pengguna membuka halaman dashboard. Pengguna menekan tombol menu. Pengguna menekan tombol Download. Pengguna melihat halaman Download.	Berhasil
UC027	Download Data	Pengguna membuka halaman dashboard. Pengguna menekan tombol menu. Pengguna menekan tombol Download. Pengguna memilih jenis, departemen, dan tahun data. Data terunduh dalam format xlsx.	Berhasil

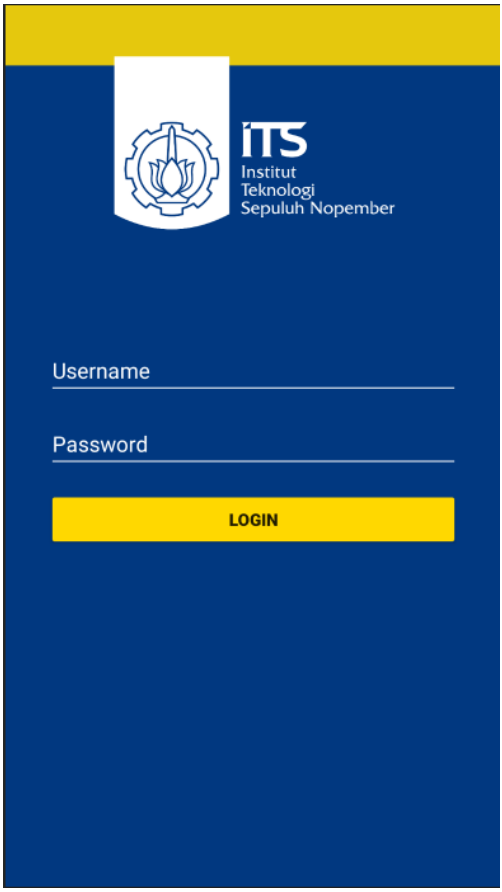
5.2.1. Login Pengguna

Pengujian dilakukan dengan cara membuka aplikasi pada smartphone. Kemudian memasukkan username dan password. Lalu muncullah halaman dashboard. Halaman dashboard berisi tombol kuliah tamu, konferensi / jurnal, prestasi dosen, training, sertifikasi seperti pada Gambar 5.1.

Gambar 5.1: Tampilan hasil uji coba login pengguna

5.2.2. Logout Pengguna

Pengujian dilakukan dengan cara membuka halaman dashboard pada aplikasi lalu menekan tombol profile di kanan atas kemudian menekan tombol logout. Lalu muncullah halaman login seperti pada Gambar 5.2.



Gambar 5.2: Tampilan hasil uji coba logout pengguna

5.2.3. Membuka Halaman Kuliah Tamu

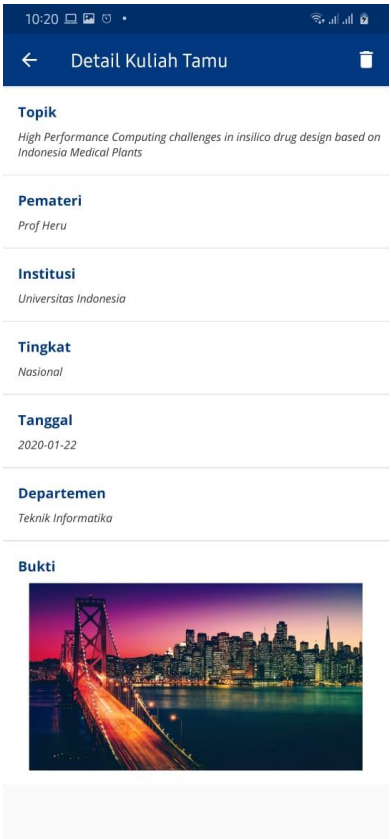
Pengujian dilakukan dengan cara membuka halaman dashboard pada aplikasi. Kemudian menekan tombol kuliah tamu. Lalu muncullah halaman kuliah tamu. Halaman kuliah tamu berisi daftar kuliah tamu, *input* untuk pencarian, *filter* pencarian, dan tombol tambah seperti pada Gambar 5.3.

Gambar 5.3: Tampilan hasil uji coba membuka halaman kuliah tamu

5.2.4. Membuka Halaman Detail Kuliah Tamu

Pengujian dilakukan dengan cara membuka halaman dashboard pada aplikasi. Kemudian menekan tombol kuliah tamu. Kemudian menekan salah satu data. Lalu muncullah halaman detail kuliah tamu. Halaman detail kuliah tamu berisi judul kegiatan,

pemateri, institusi, departemen, tingkat, tanggal, URL, gambar, tombol hapus, dan tombol validasi jika kuliah tamu belum divalidasi seperti pada Gambar 5.4.

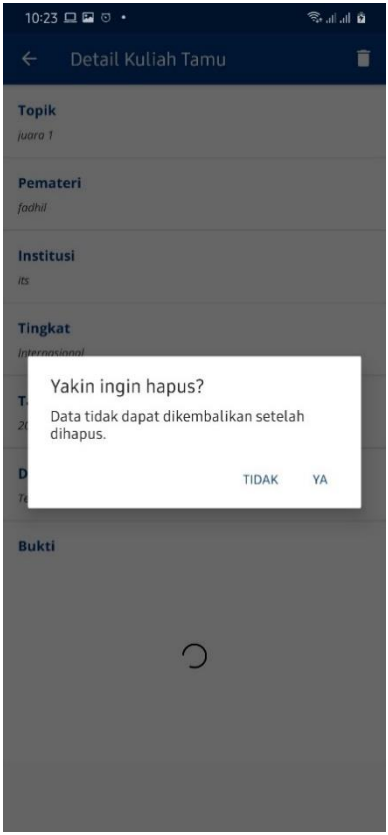


Gambar 5.4: Tampilan hasil uji coba melihat halaman detail kuliah tamu

5.2.5. Menghapus Kuliah Tamu

Pengujian dilakukan dengan cara membuka halaman dashboard pada aplikasi. Kemudian menekan tombol kuliah tamu. Kemudian menekan salah satu data. Kemudian menekan tombol

hapus. Kemudian menekan tombol ya pada modal. Lalu muncul pemberitahuan bahwa data sudah terhapus seperti Gambar 5.5.

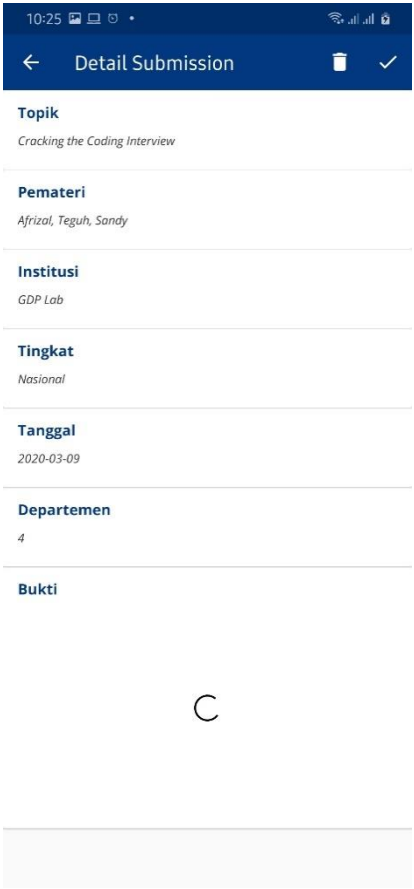


Gambar 5.5: Tampilan hasil uji coba menghapus kuliah tamu

5.2.6. Memvalidasi Kuliah Tamu

Pengujian dilakukan dengan cara membuka halaman dashboard pada aplikasi. Kemudian menekan tombol kuliah tamu. Kemudian menekan salah satu data. Kemudian menekan tombol

validasi. Kemudian menekan tombol ya pada modal. Lalu muncul pemberitahuan bahwa data sudah tervalidasi seperti Gambar 5.6.

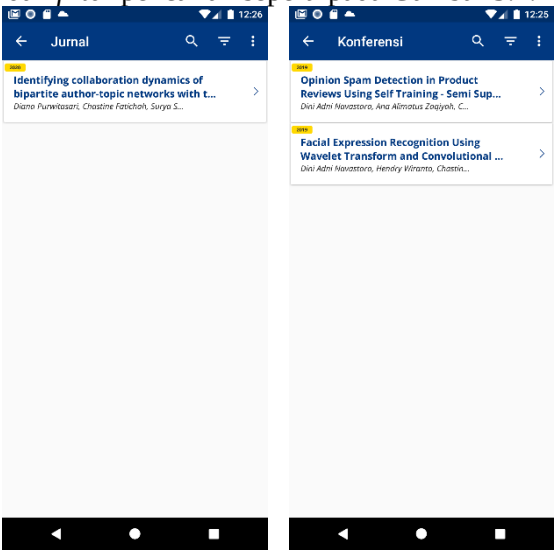


Gambar 5.6: Tampilan hasil uji coba memvalidasi kuliah tamu

5.2.7. Membuka Halaman Konferensi / Jurnal

Pengujian dilakukan dengan cara membuka halaman dashboard pada *smartphone*. Kemudian menekan tombol konferensi /

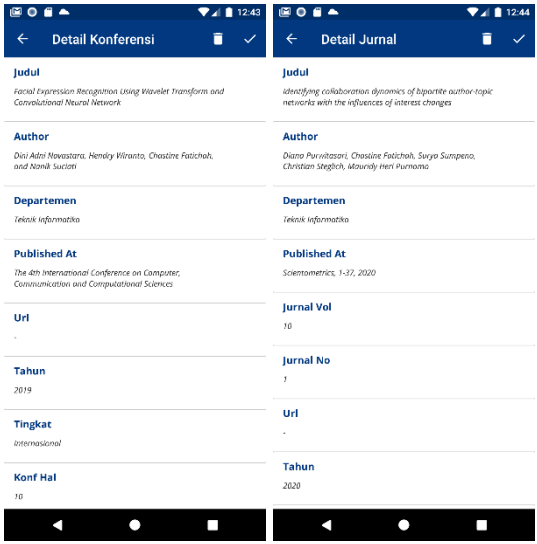
jurnal. Lalu muncullah halaman konferensi / jurnal. Halaman konferensi / jurnal berisi daftar konferensi / jurnal, *input* untuk pencarian, dan *filter* pencarian seperti pada Gambar 5.7.



Gambar 5.7: Tampilan hasil uji coba membuka halaman konferensi / jurnal

5.2.8. Membuka Halaman Detail Konferensi / Jurnal

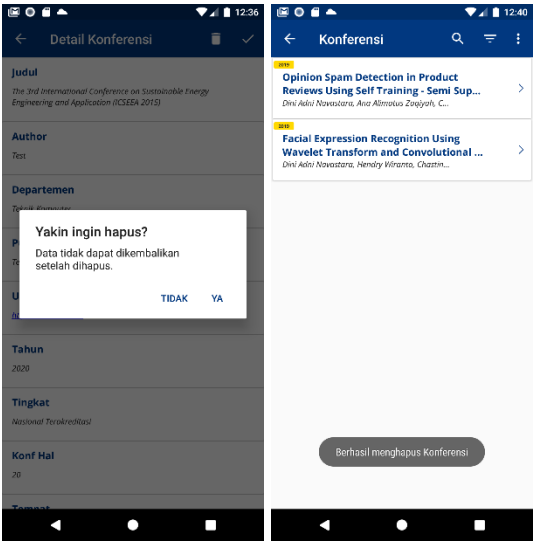
Pengujian dilakukan dengan cara membuka halaman dashboard pada *smartphone*. Kemudian menekan tombol konferensi / jurnal. Kemudian menekan salah satu data. Lalu muncullah halaman detail konferensi / jurnal. Halaman detail konferensi / jurnal berisi judul kegiatan, departemen, URL, konf hal, *check box* scopus, tanggal selesai, author, dipublish di, tingkat, tempat, tanggal mulai, tombol hapus, dan tombol validasi jika konferensi / jurnal belum divalidasi seperti pada Gambar 5.8.



Gambar 5.8: Tampilan hasil uji coba membuka halaman detail konferensi / jurnal

5.2.9. Menghapus Konferensi / Jurnal

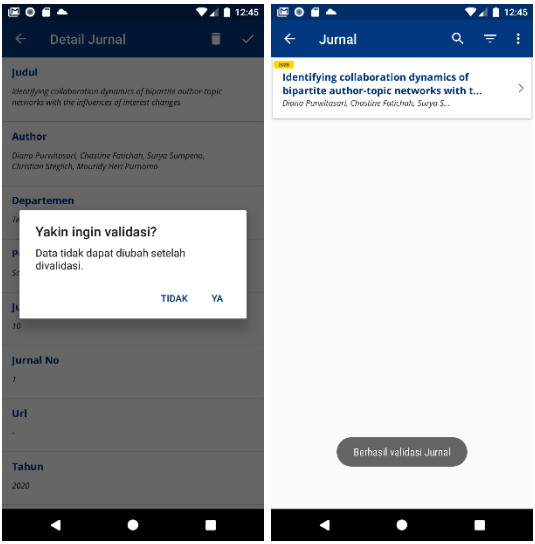
Pengujian dilakukan dengan cara membuka halaman dashboard pada *smartphone*. Kemudian menekan tombol konferensi / jurnal. Kemudian menekan salah satu data. Kemudian menekan tombol hapus. Kemudian menekan tombol ya pada dialog. Lalu muncul pemberitahuan bahwa data sudah terhapus seperti Gambar 5.9.



Gambar 5.9: Tampilan hasil uji coba menghapus konferensi / jurnal

5.2.10. Memvalidasi Konferensi / Jurnal

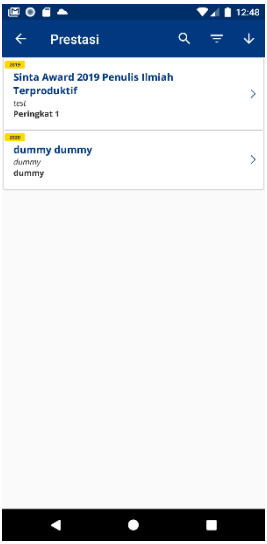
Pengujian dilakukan dengan cara membuka /dashboard pada browser. Kemudian menekan tombol konferensi / jurnal. Kemudian menekan salah satu data. Kemudian menekan tombol validasi. Kemudian menekan tombol ya pada dialog. Lalu muncul pemberitahuan bahwa data sudah tervalidasi seperti Gambar 5.10.



Gambar 5.10 Tampilan hasil uji coba memvalidasi konferensi / jurnal

5.2.11. Membuka Halaman Prestasi

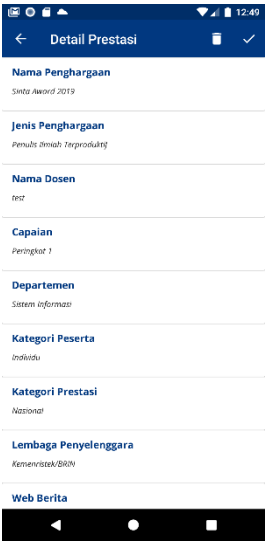
Pengujian dilakukan dengan cara membuka halaman dashboard pada *smartphone*. Kemudian menekan tombol prestasi. Lalu muncullah halaman prestasi. Halaman prestasi dosen berisi daftar prestasi dosen, *input* untuk pencarian, dan *filter* pencarian seperti pada Gambar 5.11.



Gambar 5.11: Tampilan hasil uji coba membuka halaman prestasi dosen

5.2.12. Membuka Halaman Detail Prestasi

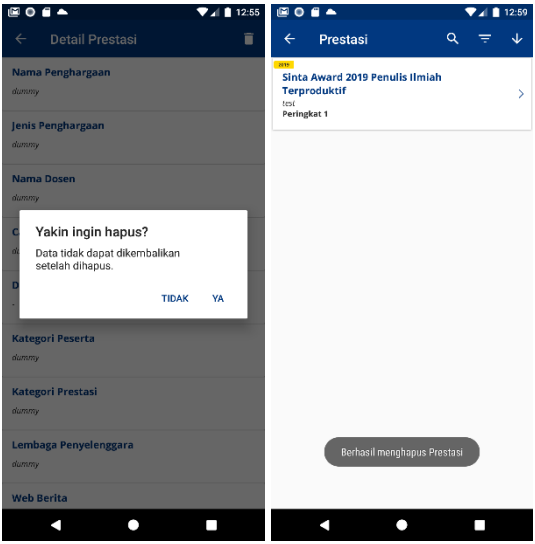
Pengujian dilakukan dengan cara membuka halaman dashboard pada *smartphone*. Kemudian menekan tombol prestasi. Kemudian menekan salah satu data. Lalu muncullah halaman detail prestasi. Halaman detail prestasi dosen berisi nama penghargaan, pelaku, kategori peserta. lembaga penyelenggara, capaian, URL, NIP, departemen, jenis prestasi, tingkat, tanggal, tombol hapus, dan tombol validasi jika prestasi belum divalidasi seperti pada Gambar 5.12.



Gambar 5.12: Tampilan hasil uji coba membuka halaman detail prestasi dosen

5.2.13. Menghapus Prestasi Dosen

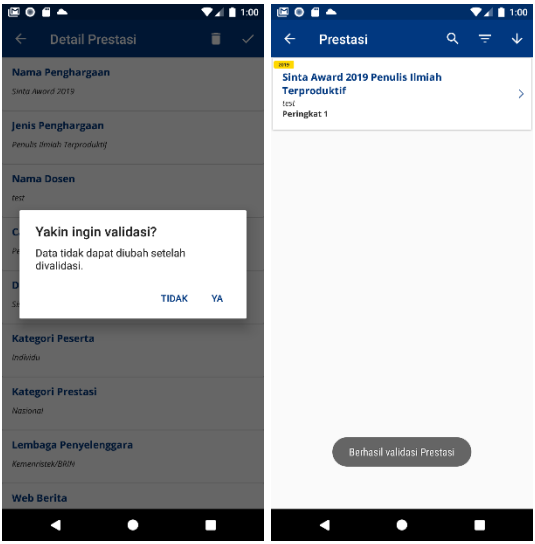
Pengujian dilakukan dengan cara membuka halaman dashboard pada *smartphone*. Kemudian menekan tombol prestasi. Kemudian menekan salah satu data. Kemudian menekan tombol hapus. Kemudian menekan tombol ya pada dialog. Lalu muncul pemberitahuan bahwa data sudah terhapus seperti Gambar 5.13.



Gambar 5.13: Tampilan hasil uji coba menghapus prestasi dosen

5.2.14. Memvalidasi Prestasi Dosen

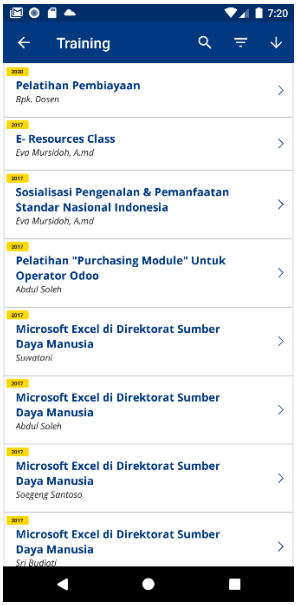
Pengujian dilakukan dengan cara membuka halaman dashboard pada *smartphone*. Kemudian menekan tombol prestasi. Kemudian menekan salah satu data. Kemudian menekan tombol validasi. Kemudian menekan tombol ya pada dialog. Lalu muncul pemberitahuan bahwa data sudah tervalidasi seperti Gambar 5.14.



Gambar 5.14: Tampilan hasil uji coba memvalidasi prestasi dosen

5.2.15. Membuka Halaman Training

Pengujian dilakukan dengan cara membuka home pada smartphone. Kemudian menekan tombol training. Lalu muncullah halaman training. Halaman training berisi daftar prestasi dosen, *input* untuk pencarian, *filter* pencarian, dan tombol tambah seperti pada Gambar 5.15.



Gambar 5.15: Tampilan hasil uji coba membuka halaman training

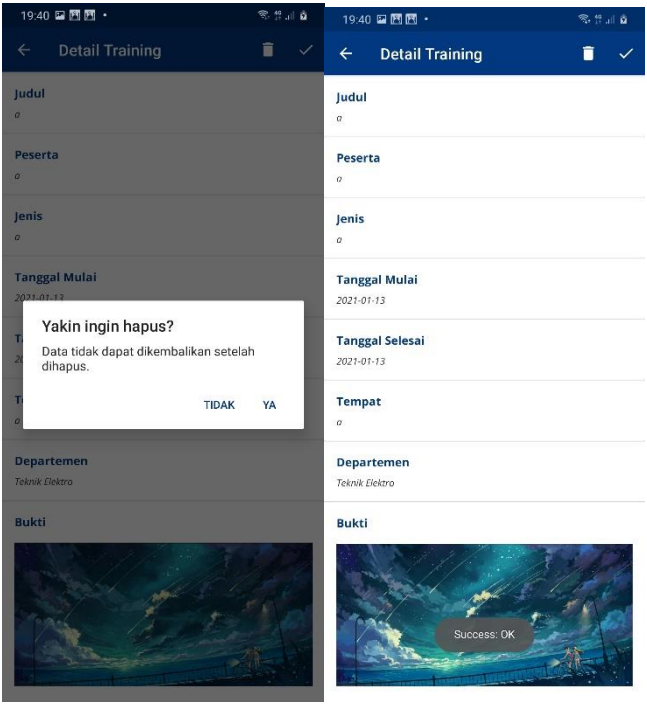
5.2.16. Membuka Halaman Detail Training

Pengujian dilakukan dengan cara membuka home pada smartphone Kemudian menekan tombol training. Kemudian menekan salah satu data. Lalu muncullah halaman detail training. Halaman detail training berisi nama penghargaan, pelaku, kategori peserta. lembaga penyelenggara, capaian, URL, NIP, departemen, jenis prestasi, tingkat, tanggal, tombol hapus, dan tombol validasi jika training belum divalidasi seperti pada Gambar 5.16.

Gambar 5.16: Tampilan hasil uji coba membuka halaman detail training

5.2.17. Menghapus Training

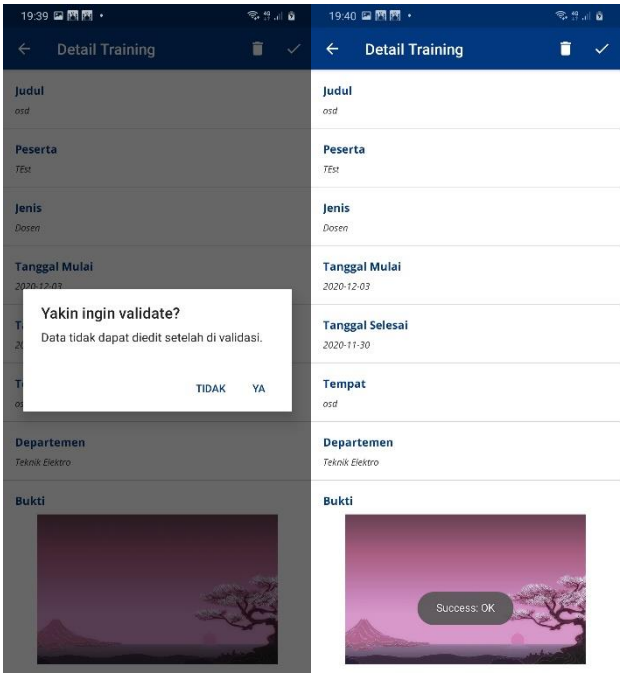
Pengujian dilakukan dengan cara membuka home pada smartphone. Kemudian menekan tombol training. Kemudian menekan salah satu data. Kemudian menekan tombol hapus. Kemudian menekan tombol ya pada modal. Lalu muncul pemberitahuan bahwa data sudah terhapus seperti Gambar 5.17.



Gambar 5.17: Tampilan hasil uji coba menghapus training

5.2.18. Memvalidasi Training

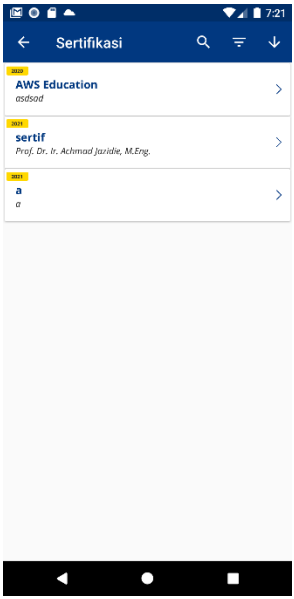
Pengujian dilakukan dengan cara membuka hone pada smartphone. Kemudian menekan tombol training n. Kemudian menekan salah satu data. Kemudian menekan tombol validasi. Kemudian menekan tombol ya pada modal. Lalu muncul pemberitahuan bahwa data sudah tervalidasi seperti Gambar 5.18.



Gambar 5.18: Tampilan hasil uji coba memvalidasi training

5.2.19. Membuka Halaman Sertifikasi

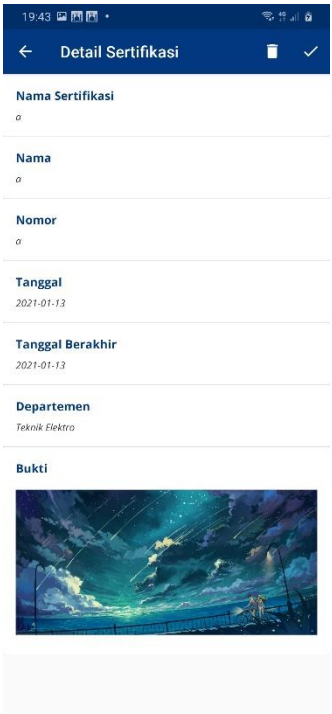
Pengujian dilakukan dengan cara membuka home pada smartphone. Kemudian menekan tombol sertifikasi. Lalu muncullah halaman sertifikasi. Halaman sertifikasi berisi daftar prestasi dosen, *input* untuk pencarian, *filter* pencarian, dan tombol tambah seperti pada Gambar 5.19.



Gambar 5.19: Tampilan hasil uji coba membuka halaman sertifikasi

5.2.20. Membuka Halaman Detail Sertifikasi

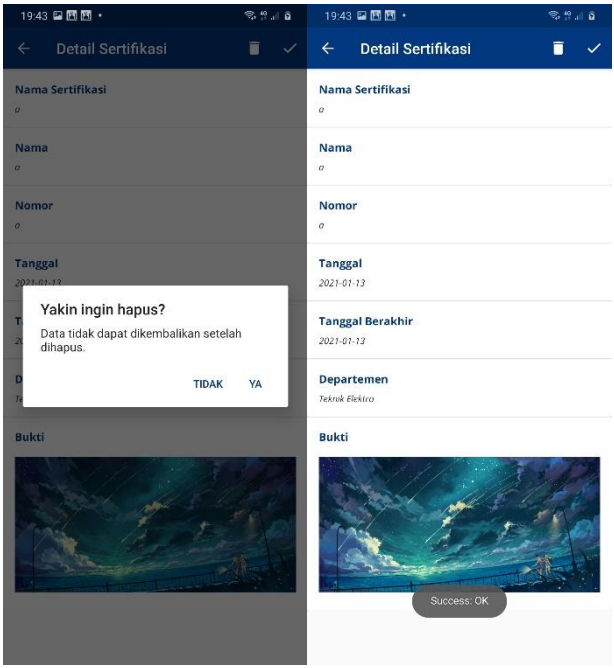
Pengujian dilakukan dengan cara membuka home pada smartphone. Kemudian menekan tombol sertifikasi. Kemudian menekan salah satu data. Lalu muncullah halaman detail sertifikasi. Halaman detail sertifikasi berisi nama sertifikasi, nama peserta, nomor, valid sampai, lembaga sertifikasi, departemen, tanggal dikeluarkan, gambar, tombol hapus, dan tombol validasi jika sertifikasi belum divalidasi seperti pada Gambar 5.21.



Gambar 5.20: Tampilan hasil uji coba membuka halaman detail sertifikasi

5.2.21. Menghapus Sertifikasi

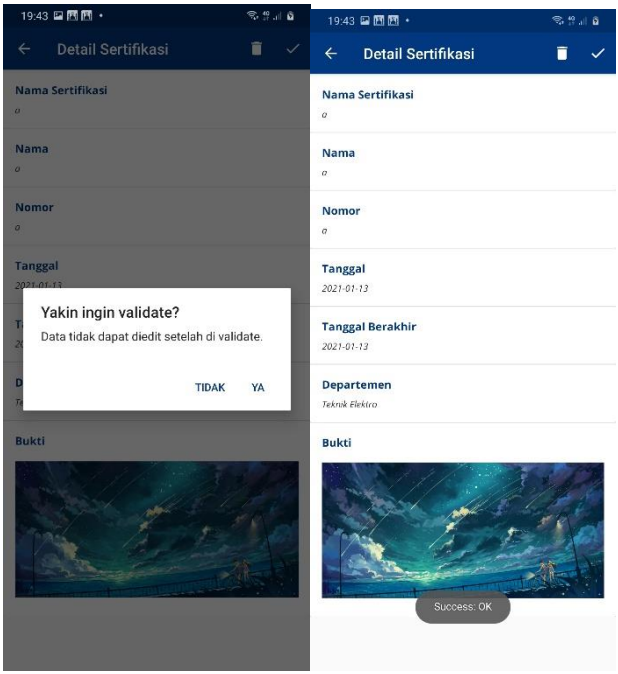
Pengujian dilakukan dengan cara membuka /dashboard pada browser. Kemudian menekan tombol sertifikasi. Kemudian menekan salah satu data. Kemudian menekan tombol hapus. Kemudian menekan tombol ya pada modal. Lalu muncul pemberitahuan bahwa data sudah terhapus seperti Gambar 5.21.



Gambar 5.21: Tampilan hasil uji coba menghapus sertifikasi

5.2.22. Memvalidasi Sertifikasi

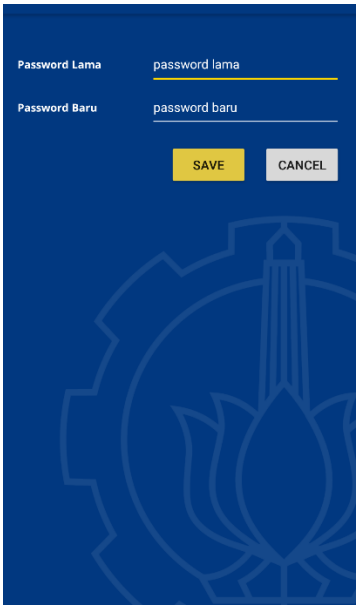
Pengujian dilakukan dengan cara membuka home pada smartphone. Kemudian menekan tombol sertifikasi. Kemudian menekan salah satu data. Kemudian menekan tombol validasi. Kemudian menekan tombol ya pada modal. Lalu muncul pemberitahuan bahwa data sudah tervalidasi seperti Gambar 5.22.



Gambar 5.22: Tampilan hasil uji coba memvalidasi sertifikasi

5.2.23. Mengubah Kata Sandi

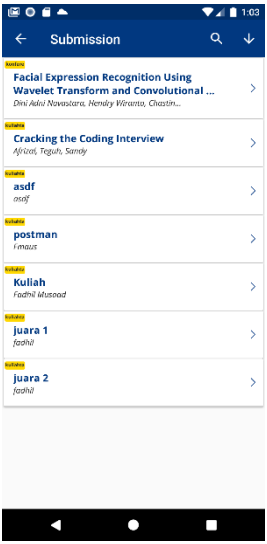
Pengujian dilakukan dengan cara membuka home pada smartphone Kemudian menekan tombol menu Kemudian menekan tombol Change Password. Kemudian mengisi password lama, password baru, dan konfirmasi password. Kemudian menekan simpan. Lalu muncul pemberitahuan data berhasil disimpan seperti pada Gambar 5.23.



Gambar 5.23: Tampilan hasil uji coba mengubah kata sandi

5.2.24. Membuka Halaman Submission

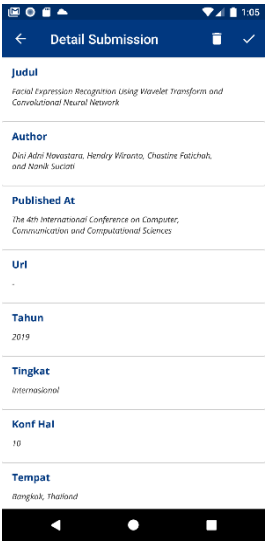
Pengujian dilakukan dengan cara membuka halaman dashboard pada *smartphone*. Kemudian menekan tombol menu. Kemudian menekan tombol submission. Lalu muncullah halaman submission seperti pada Gambar 5.24.



Gambar 5.24: Tampilan hasil uji coba membuka halaman submission

5.2.25. Membuka Halaman Detail Submission

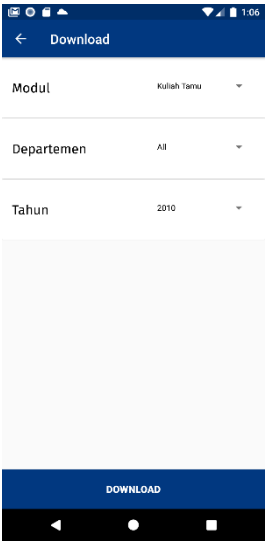
Pengujian dilakukan dengan cara membuka halaman dashboard pada *smartphone*. Kemudian menekan tombol menu. Kemudian menekan tombol submission. Kemudian menekan salah satu data. Lalu muncullah halaman detail submission seperti pada Gambar 5.25.



Gambar 5.25: Tampilan hasil uji coba membuka halaman submission

5.2.26. Membuka Halaman Download

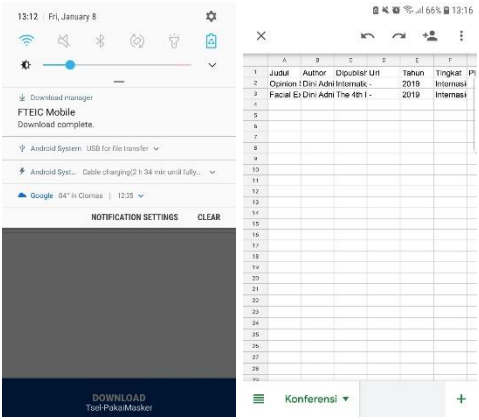
Pengujian dilakukan dengan cara membuka halaman dashboard pada *smartphone*. Kemudian menekan tombol menu. Kemudian menekan tombol download. Lalu muncullah halaman download seperti pada Gambar 5.26.



Gambar 5.26: Tampilan hasil uji coba membuka halaman download

5.2.27. Download Data

Pengujian dilakukan dengan cara membuka /dashboard pada browser. Kemudian menekan tombol profile. Kemudian menekan tombol download. Kemudian memilih jenis, departemen, dan tahun data. Kemudian menekan tombol download. Lalu data terdownload dengan format xlsx seperti pada Gambar 5.27.



Gambar 5.27: Tampilan hasil uji coba download data

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Setelah melakukan pengujian, didapatkan beberapa kesimpulan dan saran yang didapatkan untuk *Pengembangan Aplikasi Monitoring Kinerja FTEIC ITS berbasis Android.*

6.1. Kesimpulan

- Aplikasi membantu melakukan pengawasan pencatatan kegiatan di Fakultas Teknlologi Elektro dan Informatika Cerdas.

6.2. Saran

- Perlu adanya *reusability* yang lebih baik untuk tiap komponen pada *class*
- Algoritma pengurutan perlu diperbaiki agar tidak terdapat komputasi ganda.
- Tampilan aplikasi dapat lebih diperhalus menggunakan animasi agar perpindahan antar tampilan terlihat lebih responsif.

DAFTAR PUSTAKA

[1] Wikipedia (2021). Python (bahasa pemrograman) - Wikipedia bahasa Indonesia, ensiklopedia bebas. [online] Available at: https://id.wikipedia.org/wiki/Python_(bahasa_pemrograman) [Accessed 7 Jan. 2021]

[2] Mozilla Developer Network (2021). What is JavaScript? | MDN [online] Availabel at: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript [Accessed 7 Jan. 2021].

[3] Wikipedia (2021). HTML - Wikipedia bahasa Indonesia, ensiklopedia bebas [online] Available at: https://id.wikipedia.org/wiki/HTML [Accessed 7 Jan. 2021].

[4] Mozilla Developer Network (2021). What is a web server? | MDN [online] Available at: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server [Accessed 21 Jan. 2021].

[5] Codepolitan (2021) Situs-situs Populer yang Dibuat dengan Django [online] Available at: https://www.codepolitan.com/situs-situs-populer-yang-dibuat-dengan-django [Accessed at 7 Jan. 2021]. situs-populer-yang-dibuat-dengan-django [Accessed at 7 Jan. 2021].

[6] Wikipedia (2021). Android (sistem operasi) - Wikipedia bahasa Indonesia, ensiklopedia bebas. [online] Available at: https://id.wikipedia.org/wiki/Android_(sistem_operasi) [Accessed 9 Jan. 2021]

[6] Wikipedia (2021). Android Studio - Wikipedia bahasa Indonesia, ensiklopedia bebas. [online] Available at: https://id.wikipedia.org/wiki/Android_Studio [Accessed 9 Jan. 2021]

[6] Dicoding (2021). Apa Itu Kotlin? Kenapa Kita Harus Mempelajari Kotlin? [online] Available at: https://www.dicoding.com/blog/apa-itu-kotlin-kenapa-kita-harus-mempelajari-kotlin [Accessed 9 Jan. 2021]

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Muhammad Rafi Fadhilah, lahir pada 26 September 1999 di Surabaya. Penulis merupakan mahasiswa yang sedang menempuh studi di Departemen Informatika Institut Teknologi Sepuluh Nopember (ITS). Penulis pernah aktif berorganisasi di dalam kampus dengan menjadi Staff Departemen Teknologi pada Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) tahun 2019 dan juga beberapa kali menjadi asisten dosen pada beberapa mata kuliah seperti dasar pemrograman dan struktur data.