Pengembangan Aplikasi Perangkat Bergerak

Modul Mini Project

Modul Mini Project kali ini akan membahas mengenai pengembangan aplikasi Android untuk rekomendasi tempat wisata di Jawa Timur bernama “Travelupa”. Kalian akan diajak untuk mengembangkan aplikasi Android sederhana yang memiliki beberapa fitur seperti card, button, text, dan image yang terintegrasi dengan Firebase untuk autentikasinya. Selain itu, kalian diajak untuk menerapkan kotlin coroutines, room database, cameraX, dan testLab Firebase.

Bab 1 Android Studio

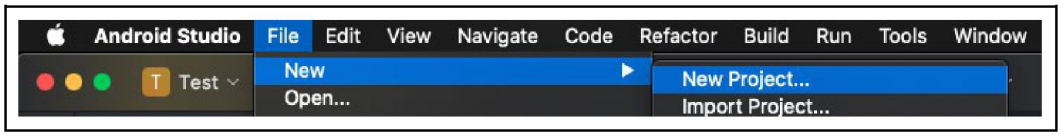
Tujuan	Mahasiswa memahami cara menggunakan Android Studio sebagai alat pengembangan aplikasi Android, termasuk cara membuat proyek baru, mengenal antarmuka, dan menjalankan aplikasi dasar.

Android Studio merupakan salah satu tools yang digunakan untuk mengembangkan aplikasi Android. Kali ini, kita akan menggunakan Android Studio untuk mengembangkan aplikasi Travelupa.

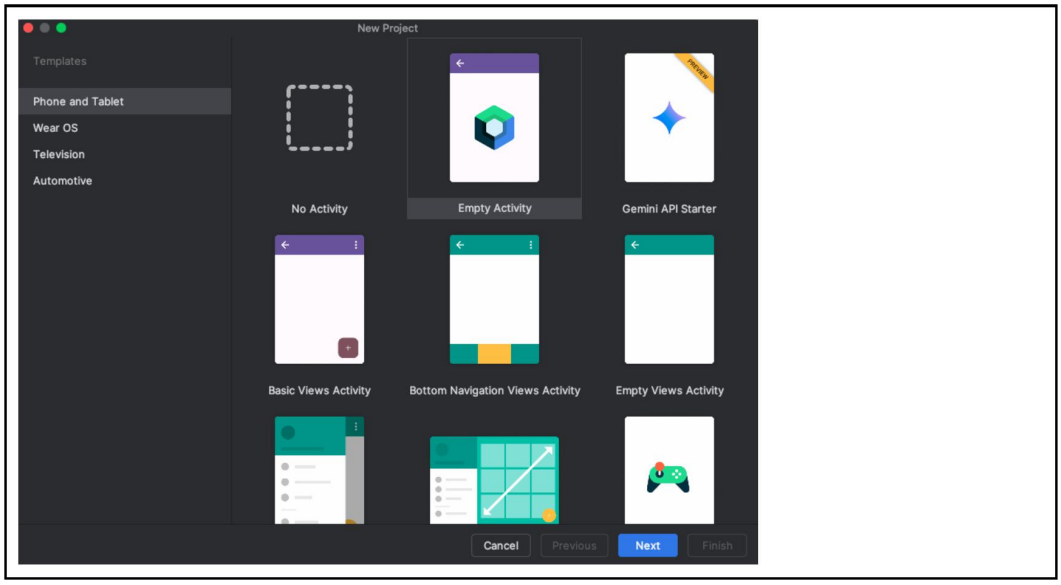
Langkah-langkah :

1. Buka Android Studio.

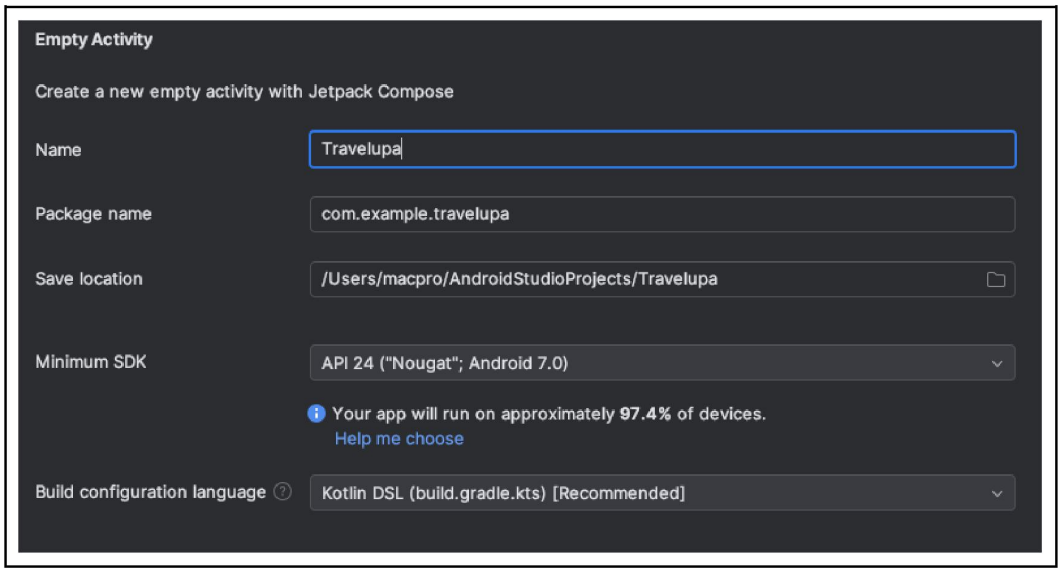
2. Buat project baru **File > New > New Project.**



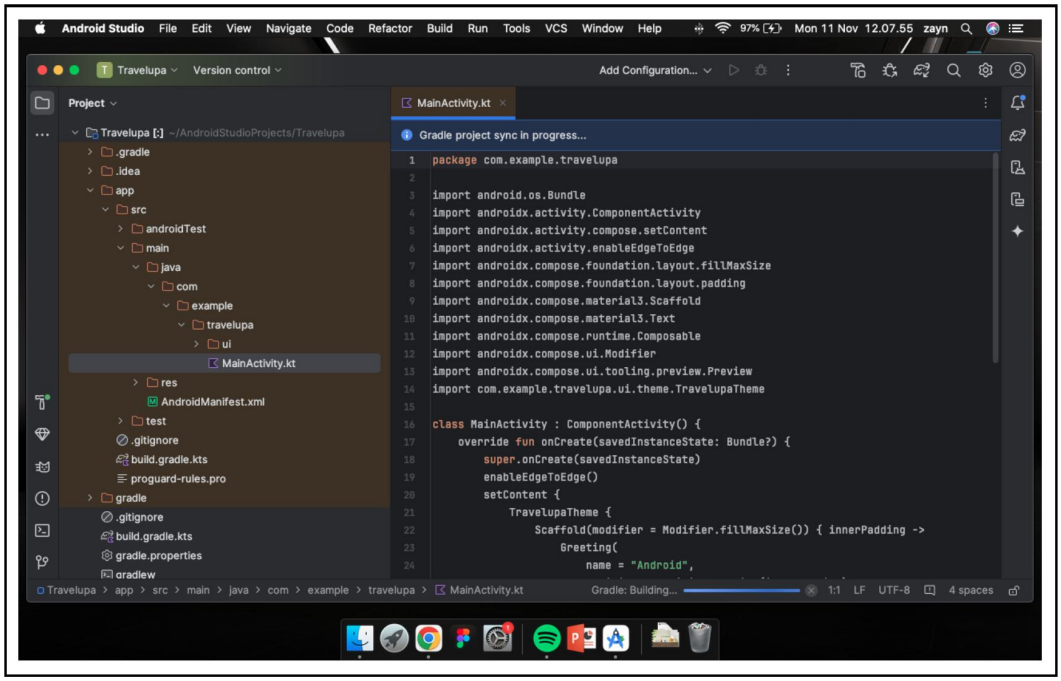
3. Pilih Empty Activity, karena kita akan mengembangkan dengan Jetpack Compose.



4. Buat nama project bebas, misal Travelupa. Lakukan konfigurasi untuk lokasi penyimpanan, minimum SDK (Android 7 disarankan), dan build menggunakan gradle.



5. Tunggu sampai semua proses selesai. Disarankan menggunakan Wi-Fi karena membutuhkan banyak dependensi yang diunduh.



6. Jika seluruh proses telah selesai, maka project telah siap.

Bab 2 Jetpack Compose (Bagian 1)

Tujuan	Mahasiswa memahami dasar-dasar Jetpack Compose dan dapat membuat elemen UI sederhana seperti teks, tombol, gambar, dan layout dasar.

Apa itu Jetpack Compose?
Jetpack Compose adalah framework yang digunakan untuk membuat antarmuka pengguna (UI) di Android menggunakan pendekatan deklaratif.

Keunggulan:
1. Mengurangi boilerplate code.
2. Responsif terhadap perubahan state.
3. Mudah untuk debugging dan pengujian.

Langkah-langkah :

A. Persiapan Project

1. Pastikan dependensi compose sudah ada di dalam gradle

```
dependencies {
    implementation 'androidx.compose.ui:ui:1.6.0'
    implementation 'androidx.compose.material:material:1.6.0'
    implementation 'androidx.compose.ui:ui-tooling-preview:1.6.0'
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.6.1'
    implementation 'androidx.activity:activity-compose:1.7.2'
}
```

2. Sinkronisasi gradle

B. Membuat Elemen UI

3. Buka file `MainActivity.kt` dan ubah isi fungsi `setContent` seperti berikut

```
package com.example.travelupa

import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.activity.ComponentActivity
import android.os.Bundle
import androidx.activity.compose.setContent
import androidx.compose.ui.text.style.TextAlign
```

```
import com.example.travelupa.ui.theme.TravelupaTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            TravelupaTheme {
                GreetingScreen()
            }
        }
    }
}

@Composable
fun GreetingScreen() {
    Box(
        modifier = Modifier.fillMaxSize(),
        contentAlignment = Alignment.Center,
    ) {
        Column(
            horizontalAlignment = Alignment.CenterHorizontally,
            modifier = Modifier
                .fillMaxWidth()
                .padding(16.dp)
        ) {
            Text(
                text = "Selamat Datang di Travelupa!",
                style = MaterialTheme.typography.h4,
                textAlign = TextAlign.Center
            )

            Spacer(modifier = Modifier.height(16.dp))

            Text(
                text = "Solusi buat kamu yang lupa kemana-mana",
                style = MaterialTheme.typography.h6
            )
        }

        Button(
            onClick = { /*TODO*/ },
            modifier = Modifier
                .width(360.dp)
                .align(Alignment.BottomCenter)
                .padding(bottom = 16.dp)
        ) {
            Text(text = "Mulai")
        }
    }
}
```

4. Jalankan dengan emulator atau ponsel android dalam mode USB debugging
5. Hasil akhirnya akan seperti ini

6. UI bersifat bebas, ini hanya tampilan sederhana yang bisa kalian coba

Bab 3 Jetpack Compose (Bagian 2)

Tujuan	Mahasiswa dapat membuat antarmuka pengguna (UI) yang lebih kompleks dengan Jetpack Compose, termasuk daftar dinamis menggunakan LazyColumn, dan menggunakan modifier untuk styling UI agar lebih responsif.

Apa itu LazyColumn?
LazyColumn adalah komponen Compose untuk menampilkan daftar data yang besar secara efisien. Data hanya akan dimuat ketika elemen-elemen tersebut akan ditampilkan di layar, yang mengoptimalkan penggunaan memori dan performa.

Langkah-langkah :

A. Membuat Daftar Menggunakan `LazyColumn`

1. Pada file MainActivity.kt, buat fungsi baru `RekomendasiTempatScreen` untuk menampilkan daftar tempat wisata.
2. Buat daftar tempat wisata dalam bentuk data sederhana

```
data class TempatWisata(val nama: String, val deskripsi: String)

val daftarTempatWisata = listOf(
    TempatWisata("Tumpak Sewu", "Air terjun tercantik di Jawa
Timur."),
    TempatWisata("Gunung Bromo", "Matahari terbitnya bagus
banget."),
    // Tambahkan sesuai keinginanmu
)
```

3. Import LazyColumn

```
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.background
```

4. Gunakan LazyColumn untuk menampilkan daftar tempat wisata

```
@Composable
fun RekomendasiTempatScreen() {
    LazyColumn(modifier = Modifier.padding(16.dp)) {
        items(daftarTempatWisata) { tempat ->
            TempatItem(tempat)
        }
    }
```

```
}

@Composable
fun TempatItem(tempat: TempatWisata) {
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(vertical = 8.dp)
            .background(MaterialTheme.colors.surface),
        elevation = 4.dp
    ) {
        Column(modifier = Modifier.padding(16.dp)) {
            Text(
                text = tempat.nama,
                style = MaterialTheme.typography.h6,
                modifier = Modifier.padding(bottom = 8.dp)
            )
            Text(
                text = tempat.deskripsi,
                style = MaterialTheme.typography.body2,
                modifier = Modifier.padding(top = 4.dp)
            )
        }
    }
}
```

5. Ketika di run belum akan menampilkan apa-apa karena setContent belum diterapkan ke halaman ini, namun jika ingin melihat preview pada halaman ini, maka setContent perlu diubah.

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            TravelupaTheme {
                RekomendasiTempatScreen()
            }
        }
    }
}
```

6. Tambahkan gambar dari Tumpak Sewu dan Bromo, masukkan ke dalam folder res/drawable. Gunakan penamaan yang benar seperti gunung_bromo.png. Jika sudah maka kode akhirnya akan seperti berikut

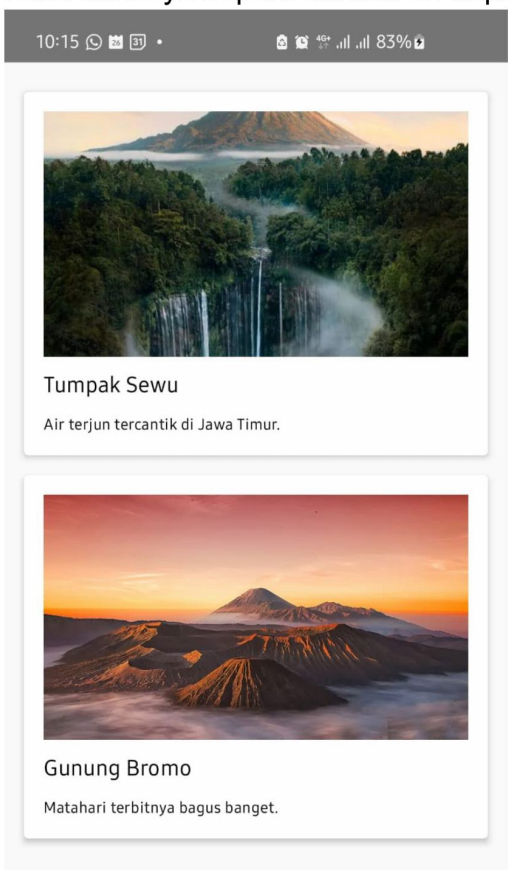
```
package com.example.travelupa

import androidx.compose.foundation.layout.*
```

```
import androidx.compose.material.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.activity.ComponentActivity
import android.os.Bundle
import androidx.activity.compose.setContent
import androidx.compose.ui.text.style.TextAlign
import com.example.travelupa.ui.theme.TravelupaTheme
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.background
import androidx.compose.runtime.*
import androidx.compose.ui.res.painterResource
import androidx.compose.foundation.Image
import androidx.compose.ui.layout.ContentScale


class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            TravelupaTheme {
                RekomendasiTempatScreen()
            }
        }
    }
}

@Composable
fun GreetingScreen() {
    Box(
        modifier = Modifier.fillMaxSize(),
        contentAlignment = Alignment.Center,
    ) {
        Column(
            horizontalAlignment = Alignment.CenterHorizontally,
            modifier = Modifier
                .fillMaxWidth()
                .padding(16.dp)
        ) {
            Text(
                text = "Selamat Datang di Travelupa!",
                style = MaterialTheme.typography.h4,
                textAlign = TextAlign.Center
            )

            Spacer(modifier = Modifier.height(16.dp))

            Text(
                text = "Solusi buat kamu yang lupa kemana-mana",
```

```
                style = MaterialTheme.typography.h6
            )

        }

        Button(
            onClick = {/*TODO*/},
            modifier = Modifier
                .width(360.dp)
                .align(Alignment.BottomCenter)
                .padding(bottom = 16.dp)
        ) {
            Text(text = "Mulai")
        }
    }
}

data class TempatWisata(val nama: String, val deskripsi: String,
val gambar: Int)

val daftarTempatWisata = listOf(
    TempatWisata(
        "Tumpak Sewu",
        "Air terjun tercantik di Jawa Timur.",
        R.drawable.tumpak_sewu),
    TempatWisata(
        "Gunung Bromo",
        "Matahari terbitnya bagus banget.",
        R.drawable.gunung_bromo)
)

@Composable
fun RekomendasiTempatScreen() {
    LazyColumn(modifier = Modifier.padding(16.dp)) {
        items(daftarTempatWisata) { tempat ->
            TempatItem(tempat)
        }
    }
}

@Composable
fun TempatItem(tempat: TempatWisata) {
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(vertical = 8.dp)
            .background(MaterialTheme.colors.surface),
        elevation = 4.dp
    ) {
        Column(modifier = Modifier.padding(16.dp)) {
            Image(
                painter = painterResource(id = tempat.gambar),
```

```
                contentDescription = tempat.nama,
                modifier = Modifier
                    .fillMaxWidth()
                    .height(200.dp),
                contentScale = ContentScale.Crop
            )
            Text(
                text = tempat.nama,
                style = MaterialTheme.typography.h6,
                modifier = Modifier.padding(bottom = 8.dp, top =
12.dp)
            )
            Text(
                text = tempat.deskripsi,
                style = MaterialTheme.typography.body2,
                modifier = Modifier.padding(top = 4.dp)
            )
        }
    }
}
```

7. Hasil akhirnya seperti berikut. UI dapat kalian improvisasi.



Bab 4 UI State

Tujuan	Mahasiswa dapat mengelola dan menampilkan perubahan data di UI menggunakan konsep state di Jetpack Compose, serta memahami cara menangani keadaan UI yang berubah berdasarkan interaksi pengguna atau data yang diterima.

Apa itu State di Jetpack Compose?
State adalah data yang dapat berubah seiring waktu dan mempengaruhi tampilan UI. Di Jetpack Compose, state dapat digunakan untuk mengubah tampilan elemen UI berdasarkan kondisi tertentu.

Langkah-langkah :

A. Menambah Data Baru

1. Kita akan membuat tampilan untuk menambah data baru bagi pengguna. Pada bagian function RekomendasiTempatScreen(), tambahkan kode berikut

```
@Composable
fun RekomendasiTempatScreen() {
    var daftarTempatWisata by remember { mutableStateOf(listOf(
        TempatWisata(t"Tumpak Sewu", "Air terjun tercantik di Jawa
Timur.", R.drawable.tumpak_sewu),
        TempatWisata("Gunung Bromo", "Matahari terbitnya bagus
banget.", R.drawable.gunung_bromo)
    )) }

    var showTambahDialog by remember { mutableStateOf(false) }

    Scaffold(
        floatingActionButton = {
            FloatingActionButton(
                onClick = { showTambahDialog = true },
                backgroundColor = MaterialTheme.colors.primary
            ) {
                Icon(Icons.Filled.Add, contentDescription = "Tambah
Tempat Wisata")
            }
        }
    ) { paddingValues ->
        Column(
            modifier = Modifier
                .padding(paddingValues)
                .padding(16.dp)
        ) {
            LazyColumn {
                items(daftarTempatWisata) { tempat ->
                    TempatItemEditable(
```

```
                        tempat = tempat,
                        onDelete = {
                            daftarTempatWisata =
daftarTempatWisata.filter { it != tempat }
                        }
                    )
                }
            }
        }

        if (showTambahDialog) {
            TambahTempatWisataDialog(
                onDismiss = { showTambahDialog = false },
                onTambah = { nama, deskripsi, gambar ->
                    val nuevoTempat = TempatWisata(nama, deskripsi,
gambar)
                    daftarTempatWisata = daftarTempatWisata +
nuevoTempat
                    showTambahDialog = false
                }
            )
        }
    }
}
```

2. Tambahkan function untuk mengedit tampilan, kali ini kita dapat membuat tampilan untuk menghapus data

```
@Composable
fun TempatItemEditable(
    tempat: TempatWisata,
    onDelete: () -> Unit
) {
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(vertical = 8.dp)
            .background(MaterialTheme.colors.surface),
        elevation = 4.dp
    ) {
        Column(modifier = Modifier.padding(16.dp)) {
            Row(
                modifier = Modifier.fillMaxWidth(),
                horizontalArrangement = Arrangement.SpaceBetween,
                verticalAlignment = Alignment.CenterVertically
            ) {
                Column(modifier = Modifier.weight(1f)) {
                    Image(
                        painter = painterResource(id =
tempat.gambar),
```

```
                        contentDescription = tempat.nama,
                        modifier = Modifier
                            .fillMaxWidth()
                            .height(200.dp),
                        contentScale = ContentScale.Crop
                    )
                    Text(
                        text = tempat.nama,
                        style = MaterialTheme.typography.h6,
                        modifier = Modifier.padding(bottom = 8.dp,
top = 12.dp)
                    )
                    Text(
                        text = tempat.deskripsi,
                        style = MaterialTheme.typography.body2,
                        modifier = Modifier.padding(top = 4.dp)
                    )
                }
                IconButton(onClick = onDelete) {
                    Icon(
                        Icons.Filled.Delete,
                        contentDescription = "Hapus Tempat Wisata",
                        tint = MaterialTheme.colors.error
                    )
                }
            }
        }
    }
}
```

3. Tambahkan dialog untuk menambah data dan gambar. Keseluruhan kode dapat dilihat berikut ini

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            TravelupaTheme {
                RekomendasiTempatScreen()
            }
        }
    }
}

@Composable
fun GreetingScreen() {
    Box(
        modifier = Modifier.fillMaxSize(),
        contentAlignment = Alignment.Center,
    ) {
```

```
        Column(
            horizontalAlignment = Alignment.CenterHorizontally,
            modifier = Modifier
                .fillMaxWidth()
                .padding(16.dp)
        ) {
            Text(
                text = "Selamat Datang di Travelupa!",
                style = MaterialTheme.typography.h4,
                textAlign = TextAlign.Center
            )

            Spacer(modifier = Modifier.height(16.dp))

            Text(
                text = "Solusi buat kamu yang lupa kemana-mana",
                style = MaterialTheme.typography.h6
            )

        }

        Button(
            onClick = {/*TODO*/},
            modifier = Modifier
                .width(360.dp)
                .align(Alignment.BottomCenter)
                .padding(bottom = 16.dp)
        ) {
            Text(text = "Mulai")
        }
    }
}

data class TempatWisata(
    val nama: String,
    val deskripsi: String,
    val gambarUriString: String? = null,
    val gambarResId: Int? = null
)

@Composable
fun RekomendasiTempatScreen() {
    var daftarTempatWisata by remember { mutableStateOf(listOf(
        TempatWisata(
            "Tumpak Sewu",
            "Air terjun tercantik di Jawa Timur.",
            gambarResId = R.drawable.tumpak_sewu
        ),
        TempatWisata(
            "Gunung Bromo",
            "Matahari terbitnya bagus banget.",
            gambarResId = R.drawable.gunung_bromo
```

```
        )
    )) }

    var showTambahDialog by remember { mutableStateOf(false) }

    Scaffold(
        floatingActionButton = {
            FloatingActionButton(
                onClick = { showTambahDialog = true },
                backgroundColor = MaterialTheme.colors.primary
            ) {
                Icon(Icons.Filled.Add, contentDescription = "Tambah
Tempat Wisata")
            }
        }
    ) { paddingValues ->
        Column(
            modifier = Modifier
                .padding(paddingValues)
                .padding(16.dp)
        ) {
            LazyColumn {
                items(daftarTempatWisata) { tempat ->
                    TempatItemEditable(
                        tempat = tempat,
                        onDelete = {
                            daftarTempatWisata =
daftarTempatWisata.filter { it != tempat }
                        }
                    )
                }
            }
        }

        // Dialog Tambah Tempat Wisata
        if (showTambahDialog) {
            TambahTempatWisataDialog(
                onDismiss = { showTambahDialog = false },
                onTambah = { nama, deskripsi, gambarUri ->
                    val uriString = gambarUri?.toString() ?: ""
                    val nuevoTempat = TempatWisata(nama, deskripsi,
uriString)
                    daftarTempatWisata = daftarTempatWisata +
nuevoTempat
                    showTambahDialog = false
                }
            )
        }
    }
}

@Composable
```

```
fun TempatItemEditable(
    tempat: TempatWisata,
    onDelete: () -> Unit
) {
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(vertical = 8.dp)
            .background(MaterialTheme.colors.surface),
        elevation = 4.dp
    ) {
        Column(modifier = Modifier.padding(16.dp)) {
            Row(
                modifier = Modifier.fillMaxWidth(),
                horizontalArrangement = Arrangement.SpaceBetween,
                verticalAlignment = Alignment.CenterVertically
            ) {
                Column(modifier = Modifier.weight(1f)) {
                    // Tampilkan gambar dari URI string atau
resource ID
                    Image(
                        painter = tempat.gambarUriString?.let {
uriString ->
                            rememberAsyncImagePainter(

ImageRequest.Builder(LocalContext.current)
                                    .data(Uri.parse(uriString))
                                    .build()
                            )
                        } ?: tempat.gambarResId?.let {
                            painterResource(id = it)
                        } ?: painterResource(id =
R.drawable.default_image),
                        contentDescription = tempat.nama,
                        modifier = Modifier
                            .fillMaxWidth()
                            .height(200.dp),
                        contentScale = ContentScale.Crop
                    )
                    Text(
                        text = tempat.nama,
                        style = MaterialTheme.typography.h6,
                        modifier = Modifier.padding(bottom = 8.dp,
top = 12.dp)
                    )
                    Text(
                        text = tempat.deskripsi,
                        style = MaterialTheme.typography.body2,
                        modifier = Modifier.padding(top = 4.dp)
                    )
                }
                IconButton(onClick = onDelete) {
```

```
                    Icon(
                        Icons.Filled.Delete,
                        contentDescription = "Hapus Tempat Wisata",
                        tint = MaterialTheme.colors.error
                    )
                }
            }
        }
    }
}

@Composable
fun TambahTempatWisataDialog(
    onDismiss: () -> Unit,
    onTambah: (String, String, String?) -> Unit
) {
    var nama by remember { mutableStateOf("") }
    var deskripsi by remember { mutableStateOf("") }
    var gambarUri by remember { mutableStateOf<Uri?>(null) }

    val gambarLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.GetContent()
    ) { uri: Uri? ->
        gambarUri = uri
    }

    AlertDialog(
        onDismissRequest = onDismiss,
        title = { Text("Tambah Tempat Wisata Baru") },
        text = {
            Column {
                TextField(
                    value = nama,
                    onValueChange = { nama = it },
                    label = { Text("Nama Tempat") },
                    modifier = Modifier.fillMaxWidth()
                )
                Spacer(modifier = Modifier.height(8.dp))
                TextField(
                    value = deskripsi,
                    onValueChange = { deskripsi = it },
                    label = { Text("Deskripsi") },
                    modifier = Modifier.fillMaxWidth()
                )
                Spacer(modifier = Modifier.height(8.dp))
                gambarUri?.let { uri ->
                    Image(
                        painter = rememberAsyncImagePainter(model =
uri),
                        contentDescription = "Gambar yang dipilih",
                        modifier = Modifier
                            .fillMaxWidth()
```

```
                            .height(200.dp),
                        contentScale = ContentScale.Crop
                    )
                }
                Spacer(modifier = Modifier.height(8.dp))
                Button(
                    onClick = { gambarLauncher.launch("image/*") },
                    modifier = Modifier.fillMaxWidth()
                ) {
                    Text("Pilih Gambar")
                }
            }
        },
        confirmButton = {
            Button(
                onClick = {
                    if (nama.isNotBlank() &&
deskripsi.isNotBlank()) {
                        onTambah(nama, deskripsi,
gambarUri?.toString())
                    }
                }
            ) {
                Text("Tambah")
            }
        },
        dismissButton = {
            Button(
                onClick = onDismiss,
                colors =
ButtonDefaults.buttonColors(backgroundColor =
MaterialTheme.colors.surface)
            ) {
                Text("Batal")
            }
        }
    )
}


@Composable
fun GambarPicker(
    gambarUri: Uri?,
    onPilihGambar: () -> Unit
) {
    Column(
        horizontalAlignment = Alignment.CenterHorizontally,
        modifier = Modifier.fillMaxWidth()
    ) {
        // Tampilkan gambar jika sudah dipilih
        gambarUri?.let { uri ->
            Image(
```

```
                painter = rememberAsyncImagePainter(
                    ImageRequest.Builder(LocalContext.current)
                        .data(uri)
                        .build()
                ),
                contentDescription = "Gambar Tempat Wisata",
                modifier = Modifier
                    .size(200.dp)
                    .clickable { onPilihGambar() },
                contentScale = ContentScale.Crop
            )
        } ?: run {
            OutlinedButton(
                onClick = onPilihGambar,
                modifier = Modifier.fillMaxWidth()
            ) {
                Icon(Icons.Filled.Add, contentDescription = "Pilih
Gambar")
                Spacer(modifier = Modifier.width(8.dp))
                Text("Pilih Gambar")
            }
        }
    }
}
```

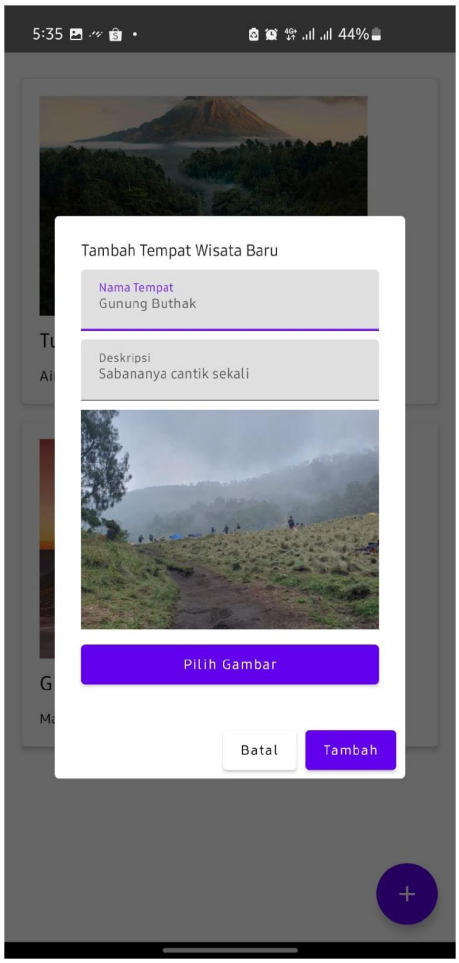
4. Jika terdapat error, pastikan import beberapa elemen berikut

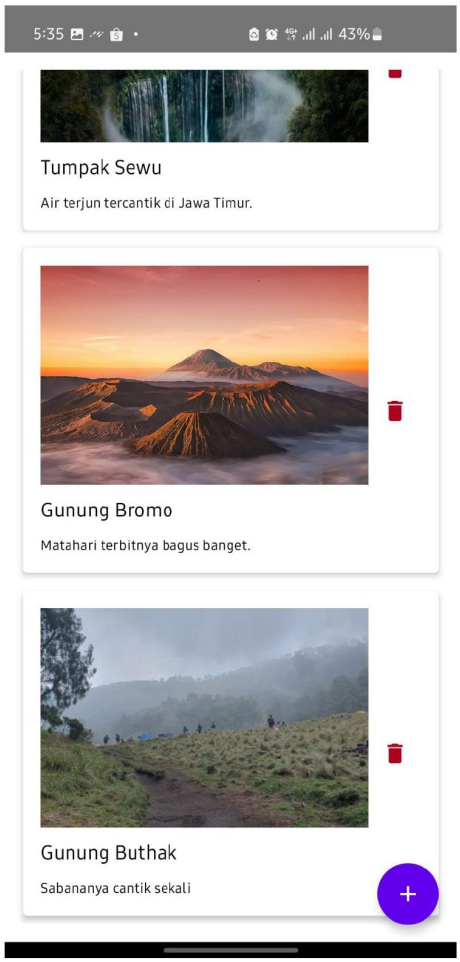
```
import androidx.compose.ui.res.painterResource
import androidx.compose.foundation.Image
import androidx.compose.ui.layout.ContentScale
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Add
import androidx.compose.material.icons.filled.Delete
import android.net.Uri
import androidx.compose.ui.platform.LocalContext
import coil.compose.rememberAsyncImagePainter
import coil.request.ImageRequest
import androidx.compose.foundation.clickable
import androidx.activity.compose.rememberLauncherForActivityResult
import androidx.activity.result.contract.ActivityResultContracts
```

5. Pastikan gradle juga sudah memiliki dependensi berikut

```
implementation("io.coil-kt:coil-compose:2.4.0")
```

6. Jalankan dan berikut merupakan hasilnya


5:35
44%
Tambah Tempat Wisata Baru
Nama Tempat
Gunung Buthak
Deskripsi
Sabananya cantik sekali
Pilih Gambar
Batal
Tambah


5:35
43%
Tumpak Sewu
Air terjun tercantik di Jawa Timur.
Gunung Bromo
Matahari terbitnya bagus banget.
Gunung Buthak
Sabananya cantik sekali

Bab 5 Google Firebase & REST API

Tujuan	Mahasiswa dapat mengintegrasikan aplikasi Android dengan Firebase untuk autentikasi pengguna dan menyimpan data menggunakan Firebase Firestore, serta menggunakan REST API untuk mengambil dan mengirim data dari dan ke server.

Pengenalan Firebase

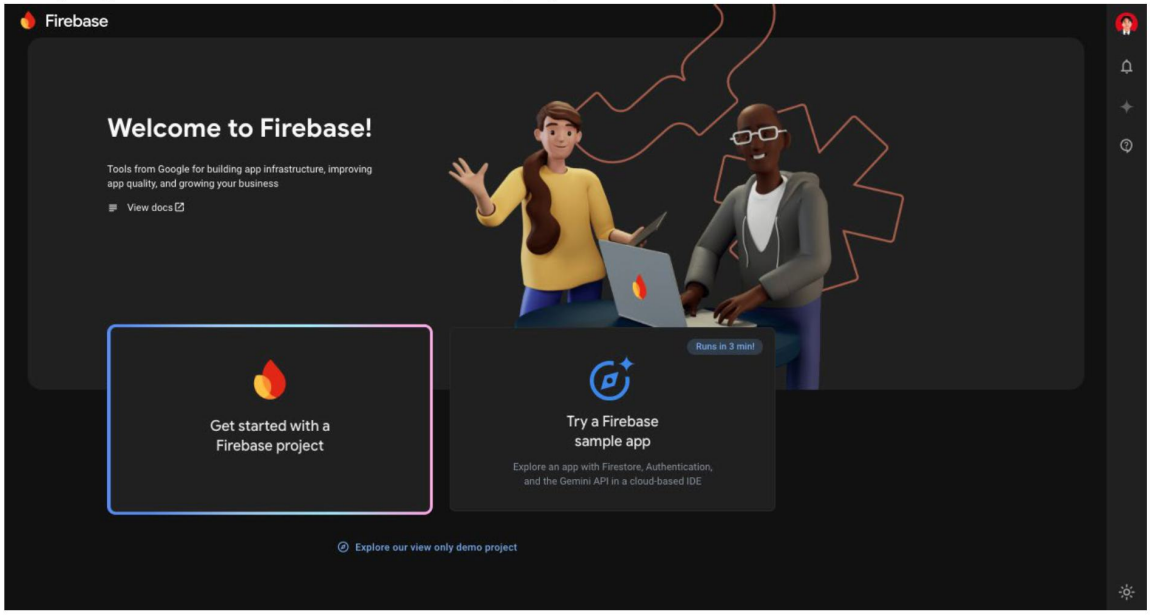
Firebase adalah platform yang dikembangkan oleh Google untuk membangun aplikasi mobile dan web. Firebase menyediakan berbagai layanan seperti autentikasi pengguna, database real-time, pengelolaan file, dan pengiriman pemberitahuan. Dalam modul ini, kita akan fokus pada Firebase Authentication dan Firebase Firestore.

- Firebase Authentication digunakan untuk menangani autentikasi pengguna, baik menggunakan email/password, akun media sosial, atau metode lainnya.
- Firebase Firestore adalah database NoSQL dari Firebase yang memungkinkan penyimpanan dan pengambilan data secara real-time.

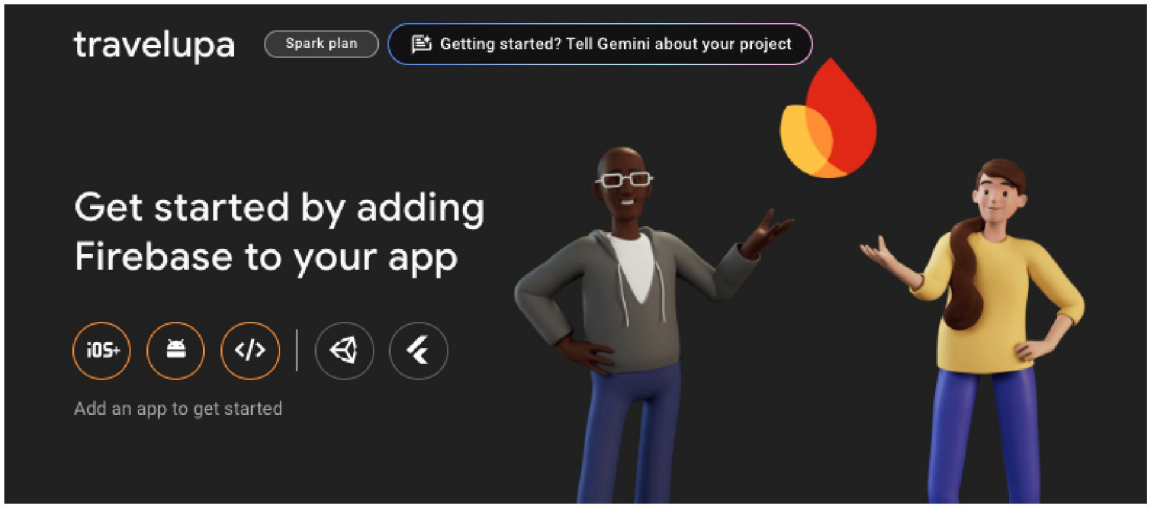
Langkah-langkah :

A. Mengintegrasikan Firebase ke dalam Proyek Android

1. Kunjungi Firebase Console dan buat proyek baru.



2. Pilih platform Android dan ikuti langkah-langkah untuk menambahkan aplikasi Android ke proyek Firebase.



3. Unduh file `google-services.json` dan letakkan di folder `app/` proyek Android.
4. Tambahkan dependensi Firebase di file `build.gradle`:

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.jetbrains.kotlin.android)
    id("com.google.gms.google-services") version "4.4.2" apply
false
}


dependencies{
       implementation("org.jetbrains.kotlinx:kotlinx-coroutines-
      android:1.7.3")
          implementation("io.coil-kt:coil-compose:2.4.0")
          implementation(platform("com.google.firebase:firebase-
      bom:33.6.0"))
          implementation("com.google.firebase:firebase-analytics")
          implementation("com.google.firebase:firebase-auth")
          implementation("androidx.navigation:navigation-
      compose:2.5.3")

}

apply(plugin = "com.google.gms.google-services")
```

5. Sinkronisasi gradle

B. Firebase Auth untuk Login Pengguna

6. Tambahkan UI untuk login pengguna

```
@Composable
```

```
fun LoginScreen(
    onLoginSuccess: () -> Unit
) {
    var email by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var errorMessage by remember { mutableStateOf<String?>(null) }
    var isLoading by remember { mutableStateOf(false) }

    val coroutineScope = rememberCoroutineScope()

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center
    ) {
        OutlinedTextField(
            value = email,
            onValueChange = {
                email = it
                errorMessage = null
            },
            label = { Text("Email") },
            singleLine = true,
            modifier = Modifier.fillMaxWidth()
        )

        Spacer(modifier = Modifier.height(8.dp))

        OutlinedTextField(
            value = password,
            onValueChange = {
                password = it
                errorMessage = null
            },
            label = { Text("Password") },
            singleLine = true,
            modifier = Modifier.fillMaxWidth()
        )

        Spacer(modifier = Modifier.height(16.dp))

        Button(
            onClick = {
                if (email.isBlank() || password.isBlank()) {
                    errorMessage = "Please enter email and
password"
                    return@Button
                }

                isLoading = true
                errorMessage = null
```

```
                coroutineScope.launch {
                    try {
                        // Firebase Authentication
                        val authResult =
withContext(Dispatchers.IO) {

FirebaseAuth.getInstance().signInWithEmailAndPassword(email,
password).await()
                        }
                        isLoading = false
                        onLoginSuccess()
                    } catch (e: Exception) {
                        isLoading = false
                        errorMessage = "Login failed:
${e.localizedMessage}"
                    }
                }
            },
            modifier = Modifier.fillMaxWidth(),
            enabled = !isLoading
        ) {
            if (isLoading) {
                CircularProgressIndicator(
                    modifier = Modifier.size(20.dp),
                    color = Color.White
                )
            } else {
                Text("Login")
            }
        }

        errorMessage?.let {
            Text(
                text = it,
                color = Color.Red,
                modifier = Modifier.padding(top = 8.dp)
            )
        }
    }
}
```

7. Buat navigasi agar setelah login dapat masuk ke dalam screen rekomendasi tempat wisata

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        FirebaseApp.initializeApp(this)
        setContent {
```

```
            TravelupaTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = Color.White
                ) {
                    AppNavigation()
                }
            }
        }
    }
}

@Composable
fun AppNavigation() {
    val navController = rememberNavController()

    NavHost(
        navController = navController,
        startDestination = Screen.Login.route
    ) {
        composable(Screen.Login.route) {
            LoginScreen(
                onLoginSuccess = {

navController.navigate(Screen.RekomendasiTempat.route) {
                        popUpTo(Screen.Login.route) { inclusive =
true }
                    }
                }
            )
        }
        composable(Screen.RekomendasiTempat.route) {
            RekomendasiTempatScreen(
                onBackToLogin = {
                    navController.navigateUp()
                }
            )
        }
    }
}
```

8. Jalankan dan lihat hasilnya

C. Firebase Firestore untuk Menyimpan Data

9. Buat koleksi `tempat_wisata` di Firestore
10. Simpan data tempat wisata menggunakan Firestore
11. Pada function TambahTempatWisataDialog ubah agar data nama dan deskripsi dapat disimpan ke dalam Firestore, sedangkan gambar akan disimpan local (modul Room).

```
@Composable
fun TambahTempatWisataDialog(
    firestore: FirebaseFirestore,
    context: Context,
    onDismiss: () -> Unit,
    onTambah: (String, String, String?) -> Unit
) {
    var nama by remember { mutableStateOf("") }
    var deskripsi by remember { mutableStateOf("") }
    var gambarUri by remember { mutableStateOf<Uri?>(null) }
    var isUploading by remember { mutableStateOf(false) }

    val gambarLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.GetContent()
    ) { uri: Uri? ->
        gambarUri = uri
    }

    AlertDialog(
        onDismissRequest = onDismiss,
        title = { Text("Tambah Tempat Wisata Baru") },
        text = {
            Column {
                TextField(
                    value = nama,
                    onValueChange = { nama = it },
                    label = { Text("Nama Tempat") },
                    modifier = Modifier.fillMaxWidth(),
                    enabled = !isUploading
                )
                Spacer(modifier = Modifier.height(8.dp))
                TextField(
                    value = deskripsi,
                    onValueChange = { deskripsi = it },
                    label = { Text("Deskripsi") },
                    modifier = Modifier.fillMaxWidth(),
                    enabled = !isUploading
                )
                Spacer(modifier = Modifier.height(8.dp))
                gambarUri?.let { uri ->
                    Image(
                        painter = rememberAsyncImagePainter(model =
uri),
                        contentDescription = "Gambar yang dipilih",
                        modifier = Modifier
                            .fillMaxWidth()
                            .height(200.dp),
                        contentScale = ContentScale.Crop
                    )
                }
```

```
                Spacer(modifier = Modifier.height(8.dp))
                Button(
                    onClick = { gambarLauncher.launch("image/*") },
                    modifier = Modifier.fillMaxWidth(),
                    enabled = !isUploading
                ) {
                    Text("Pilih Gambar")
                }
            }
        },
        confirmButton = {
            Button(
                onClick = {
                    if (nama.isNotBlank() && deskripsi.isNotBlank()
&& gambarUri != null) {
                        isUploading = true
                        val tempatWisata = TempatWisata(nama,
deskripsi)

                        uploadImageToFirestore(
                            firestore,
                            context,
                            gambarUri!!,
                            tempatWisata,
                            onSuccess = { uploadedTempat ->
                                isUploading = false
                                onTambah(nama, deskripsi,
uploadedTempat.gambarUriString)
                                onDismiss()
                            },
                            onFailure = { e ->
                                isUploading = false
                            }
                        )
                    }
                },
                enabled = !isUploading
            ) {
                if (isUploading) {
                    CircularProgressIndicator(
                        modifier = Modifier.size(20.dp),
                        color = Color.White
                    )
                } else {
                    Text("Tambah")
                }
            }
        },
        dismissButton = {
            Button(
                onClick = onDismiss,
                colors =
```

```
ButtonDefaults.buttonColors(backgroundColor =
MaterialTheme.colors.surface),
                enabled = !isUploading
            ) {
                Text("Batal")
            }
        }
    )
}
```

12. Update data class

```
data class TempatWisata(
    val nama: String = "",
    val deskripsi: String = "",
    val gambarUriString: String? = null,
    val gambarResId: Int? = null
)
```

13. Agar data yang sudah disimpan ke firestore dapat ditampilkan ke halaman, tambahkan kode berikut. Function ini juga bisa digunakan untuk menambah atau menghapus data

```
@Composable
fun TempatItemEditable(
    tempat: TempatWisata,
    onDelete: () -> Unit
) {
    val firestore = FirebaseFirestore.getInstance()
    var expanded by remember { mutableStateOf(false) }

    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(vertical = 8.dp)
            .background(MaterialTheme.colors.surface),
        elevation = 4.dp
    ) {
        Column(modifier = Modifier.padding(16.dp)) {
            Image(
                painter = tempat.gambarUriString?.let { uriString -
>
                    rememberAsyncImagePainter(
                        ImageRequest.Builder(LocalContext.current)
                            .data(Uri.parse(uriString))
                            .build()
                    )
                } ?: tempat.gambarResId?.let {
```

```
                    painterResource(id = it)
                } ?: painterResource(id =
R.drawable.default_image),
                contentDescription = tempat.nama,
                modifier = Modifier
                    .fillMaxWidth()
                    .height(200.dp),
                contentScale = ContentScale.Crop
            )
            Box(modifier = Modifier.fillMaxWidth()) {
                Column(modifier = Modifier
                    .align(Alignment.CenterStart)
                ) {
                    Text(
                        text = tempat.nama,
                        style = MaterialTheme.typography.h6,
                        modifier = Modifier.padding(bottom = 8.dp,
top = 12.dp)
                    )
                    Text(
                        text = tempat.deskripsi,
                        style = MaterialTheme.typography.body2,
                        modifier = Modifier.padding(top = 4.dp)
                    )
                }
                IconButton(
                    onClick = { expanded = true },
                    modifier = Modifier.align(Alignment.TopEnd)
                ) {
                    Icon(Icons.Default.MoreVert, contentDescription
= "More options")
                }
                DropdownMenu(
                    expanded = expanded,
                    onDismissRequest = { expanded = false },
                    offset = DpOffset(250.dp, 0.dp),
                ) {
                    DropdownMenuItem(onClick = {
                        expanded = false

firestore.collection("tempat_wisata").document(tempat.nama)
                            .delete()
                            .addOnSuccessListener {
                                onDelete()
                            }
                            .addOnFailureListener { e ->
                                Log.w("TempatItemEditable", "Error
deleting document", e)
                            }
                    }) {
                        Text("Delete")
                    }
```

```
                }
            }
        }
    }
}
```

14. Function TambahTempatWisataDialog untuk menambah data di firestore dan menyimpan gambar ke local

```
@Composable
fun TambahTempatWisataDialog(
    firestore: FirebaseFirestore,
    context: Context,
    onDismiss: () -> Unit,
    onTambah: (String, String, String?) -> Unit
) {
    var nama by remember { mutableStateOf("") }
    var deskripsi by remember { mutableStateOf("") }
    var gambarUri by remember { mutableStateOf<Uri?>(null) }
    var isUploading by remember { mutableStateOf(false) }

    val gambarLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.GetContent()
    ) { uri: Uri? ->
        gambarUri = uri
    }

    AlertDialog(
        onDismissRequest = onDismiss,
        title = { Text("Tambah Tempat Wisata Baru") },
        text = {
            Column {
                TextField(
                    value = nama,
                    onValueChange = { nama = it },
                    label = { Text("Nama Tempat") },
                    modifier = Modifier.fillMaxWidth(),
                    enabled = !isUploading
                )
                Spacer(modifier = Modifier.height(8.dp))
                TextField(
                    value = deskripsi,
                    onValueChange = { deskripsi = it },
                    label = { Text("Deskripsi") },
                    modifier = Modifier.fillMaxWidth(),
                    enabled = !isUploading
                )
                Spacer(modifier = Modifier.height(8.dp))
                gambarUri?.let { uri ->
                    Image(
```

```
                        painter = rememberAsyncImagePainter(model =
uri),
                        contentDescription = "Gambar yang dipilih",
                        modifier = Modifier
                            .fillMaxWidth()
                            .height(200.dp),
                        contentScale = ContentScale.Crop
                    )
                }
                Spacer(modifier = Modifier.height(8.dp))
                Button(
                    onClick = { gambarLauncher.launch("image/*") },
                    modifier = Modifier.fillMaxWidth(),
                    enabled = !isUploading
                ) {
                    Text("Pilih Gambar")
                }
            }
        },
        confirmButton = {
            Button(
                onClick = {
                    if (nama.isNotBlank() && deskripsi.isNotBlank()
&& gambarUri != null) {
                        isUploading = true
                        val tempatWisata = TempatWisata(nama,
deskripsi)

                        uploadImageToFirestore(
                            firestore,
                            context,
                            gambarUri!!,
                            tempatWisata,
                            onSuccess = { uploadedTempat ->
                                isUploading = false
                                onTambah(nama, deskripsi,
uploadedTempat.gambarUriString)
                                onDismiss()
                            },
                            onFailure = { e ->
                                isUploading = false
                            }
                        )
                    }
                },
                enabled = !isUploading
            ) {
                if (isUploading) {
                    CircularProgressIndicator(
                        modifier = Modifier.size(20.dp),
                        color = Color.White
                    )
```

```
                } else {
                    Text("Tambah")
                }
            }
        },
        dismissButton = {
            Button(
                onClick = onDismiss,
                colors =
ButtonDefaults.buttonColors(backgroundColor =
MaterialTheme.colors.surface),
                enabled = !isUploading
            ) {
                Text("Batal")
            }
        }
    )
}
```

15. Coba run, jika terjadi error silahkan lihat halaman improvisasi kode
16. Coba buat halaman registrasi yang membuat user dapat membuat akun baru dan tersimpan ke firebase

Bab 6 Kotlin Coroutines

Tujuan	Mahasiswa dapat memahami konsep dasar **Kotlin Coroutines** dan menerapkannya dalam aplikasi Android untuk menjalankan operasi asinkron, seperti mengambil data dari Firebase atau REST API, tanpa memblokir thread utama dan menjaga UI tetap responsif.

Pengenalan Firebase

Kotlin Coroutines adalah mekanisme untuk menangani operasi asinkron secara lebih sederhana dan elegan dibandingkan dengan penggunaan callback atau thread tradisional. Dengan menggunakan coroutines, kamu bisa menulis kode asinkron dalam cara yang lebih mirip dengan kode sinkron, menghindari penggunaan callback yang membingungkan.

Keuntungan Kotlin Coroutines:

- Membuat kode asinkron lebih bersih dan mudah dibaca.
- Menghindari masalah callback hell atau race conditions.
- Tidak memblokir thread utama (UI thread), sehingga aplikasi tetap responsif.

Kotlin coroutines sendiri sudah diterapkan pada bab sebelumnya yaitu pada bagian

```
coroutineScope.launch {
                    try {
                        // Firebase Authentication
                        val authResult = withContext(Dispatchers.IO) {

FirebaseAuth.getInstance().signInWithEmailAndPassword(email,
password).await()
                        }
                        isLoading = false
                        onLoginSuccess()
                    } catch (e: Exception) {
                        isLoading = false
                        errorMessage = "Login failed:
${e.localizedMessage}"
                    }
                }
```

Sehingga pada bab ini, kita akan membuat aplikasi menyimpan data pengguna saat ini yang mana ketika pengguna keluar dan menutup aplikasi maka pengguna tidak perlu melakukan login kembali

Langkah-langkah :

A. Menyiapkan Kotlin Coroutines ke dalam Proyek Android

1. Untuk menggunakan coroutines di Android, Anda perlu menambahkan dependensi pada file `build.gradle` (app level):

```
dependencies {
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-
android:1.6.4'
}
```

2. Sinkronisasi gradle

B. Melakukan Pengecekan Terhadap State Autentikasi Pengguna

3. Ubah MainActivity menjadi berikut

```
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.FirebaseUser

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        FirebaseApp.initializeApp(this)

        val currentUser: FirebaseUser? =
FirebaseAuth.getInstance().currentUser

        setContent {
            TravelupaTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = Color.White
                ) {
                    AppNavigation(currentUser)
                }
            }
        }
    }
}
```

C. Navigasi ke Halaman yang Sesuai

4. Ubah AppNavigation agar menyetujui pengguna saat ini

```
@Composable
fun AppNavigation(currentUser: FirebaseUser?) {
    val navController = rememberNavController()
```

```
    NavHost(
        navController = navController,
        startDestination = if (currentUser != null)
Screen.RekomendasiTempat.route else Screen.Login.route
    ) {
        composable(Screen.Login.route) {
            LoginScreen(
                onLoginSuccess = {

navController.navigate(Screen.RekomendasiTempat.route) {
                        popUpTo(Screen.Login.route) { inclusive =
true }
                    }
                }
            )
        }
        composable(Screen.RekomendasiTempat.route) {
            RekomendasiTempatScreen(
                onBackToLogin = {
                    FirebaseAuth.getInstance().signOut()
                    navController.navigate(Screen.Login.route) {
                        popUpTo(Screen.RekomendasiTempat.route) {
inclusive = true }
                    }
                }
            )
        }
    }
}
```

Bab 7 Jetpack Navigation

Tujuan	Mahasiswa dapat memahami dan mengimplementasikan **Jetpack Navigation** untuk menangani navigasi antar layar dalam aplikasi Android.

Apa itu Jetpack Compose
Jetpack Navigation adalah komponen Android yang membantu mempermudah dan memperjelas navigasi antar fragment dan aktivitas dalam aplikasi. Dengan menggunakan Navigation Component, aplikasi dapat mengelola fragment dan aktivitas secara deklaratif, tanpa harus menulis kode navigasi manual, seperti menggunakan `FragmentTransaction` atau `Intent`.

Saat ini, kita sudah memiliki 3 screen yaitu Greetings, Login, dan RekomendasiTempat. Tugas kita adalah menavigasikan ketiganya dari Greetings, lalu masuk ke halaman Login dan RekomendasiTempat.

Langkah-langkah :

A. Menyiapkan Jetpack Navigation ke dalam Proyek Android

1. Untuk menggunakan jetpack navigation di Android, Anda perlu menambahkan dependensi pada file `build.gradle` (app level):

```
dependencies {
    implementation 'androidx.navigation:navigation-fragment-
ktx:2.7.0'
    implementation 'androidx.navigation:navigation-ui-ktx:2.7.0'
}
```

2. Sinkronisasi gradle

B. Mengubah AppNavigation

3. Pada function AppNavigation, tambahkan Greetings

```
@Composable
fun AppNavigation(currentUser: FirebaseUser?) {
    val navController = rememberNavController()

    NavHost(
        navController = navController,
        startDestination = if (currentUser != null)
Screen.RekomendasiTempat.route else Screen.Greeting.route
    ) {
        composable(Screen.Greeting.route) {
```

```
            GreetingScreen(
                onStart = {
                    navController.navigate(Screen.Login.route) {
                        popUpTo(Screen.Greeting.route) { inclusive
= true }
                    }
                }
            )
        }
        composable(Screen.Login.route) {
            LoginScreen(
                onLoginSuccess = {

navController.navigate(Screen.RekomendasiTempat.route) {
                        popUpTo(Screen.Login.route) { inclusive =
true }
                    }
                }
            )
        }
        composable(Screen.RekomendasiTempat.route) {
            RekomendasiTempatScreen(
                onBackToLogin = {
                    FirebaseAuth.getInstance().signOut()
                    navController.navigate(Screen.Greeting.route) {
                        popUpTo(Screen.RekomendasiTempat.route) {
inclusive = true }
                    }
                }
            )
        }
    }
}
```

C. Mengubah Greeting Page

4. Update function Greeting agar bisa bernavigasi menuju Login

```
@Composable
fun GreetingScreen(
    onStart: () -> Unit
) {
    Box(
        modifier = Modifier.fillMaxSize(),
        contentAlignment = Alignment.Center,
    ) {
        Column(
            horizontalAlignment = Alignment.CenterHorizontally,
            modifier = Modifier
                .fillMaxWidth()
                .padding(16.dp)
```

```
        ) {
            Text(
                text = "Selamat Datang di Travelupa!",
                style = MaterialTheme.typography.h4,
                textAlign = TextAlign.Center
            )

            Spacer(modifier = Modifier.height(16.dp))

            Text(
                text = "Solusi buat kamu yang lupa kemana-mana",
                style = MaterialTheme.typography.h6
            )

        }

        Button(
            onClick = onStart,
            modifier = Modifier
                .width(360.dp)
                .align(Alignment.BottomCenter)
                .padding(bottom = 16.dp)
        ) {
            Text(text = "Mulai")
        }
    }
}
```

Bab 8 Room Database

Tujuan	Mahasiswa dapat memahami dan mengimplementasikan **Room Database** untuk menyimpan dan mengelola data secara lokal dalam aplikasi Android menggunakan **SQLite** dengan antarmuka yang lebih sederhana dan lebih kuat.

Apa itu Room Database

Room adalah sebuah Object Relational Mapping (ORM) library yang memungkinkan pengembang Android untuk mengakses SQLite database dengan cara yang lebih mudah dan lebih terstruktur. Room menyediakan API yang memungkinkan aplikasi menyimpan dan mengelola data secara lokal dalam database SQLite tanpa harus menulis query SQL secara manual.

Keuntungan menggunakan Room:

- Mudah Digunakan: Room menyediakan API berbasis objek untuk mengakses data.
- Mengurangi Boilerplate Code: Room mengurangi kebutuhan untuk menulis banyak kode boilerplate, seperti kode untuk membuka, menutup, dan mengelola koneksi database.
- Integrasi dengan LiveData: Room mendukung integrasi dengan LiveData, yang memungkinkan aplikasi merespons perubahan data secara otomatis.
- Validasi Query di Kompilasi: Room memeriksa query SQL pada waktu kompilasi untuk memastikan bahwa query tersebut valid.

Langkah-langkah :

A. Menyiapkan Room ke dalam Proyek Android

1. Untuk menggunakan room di Android, Anda perlu menambahkan dependensi pada file `build.gradle` (app level):

```
plugin{
      kotlin("kapt")
}

dependencies{
      implementation("androidx.room:room-runtime:2.5.0")
      kapt("androidx.room:room-compiler:2.5.0")
      implementation("androidx.room:room-ktx:2.5.0")
}
```

2. Jika terjadi error, coba sinkronisasi gradle

B. Membuat Entity di Room

3. Buat kelas data dengan anotasi `@Entity`.
4. Tentukan **primary key** menggunakan anotasi `@PrimaryKey`.
5. Tentukan kolom-kolom data yang ingin disimpan.

```
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "images")
data class ImageEntity(
    @PrimaryKey(autoGenerate = true)
    val id: Long = 0,
    val localPath: String,
    val tempatWisataId: String? = null
)
```

C. Membuat Data Access Object (DAO)

6. Buat file ImageDao.kt

```
import androidx.room.Dao
import androidx.room.Insert
import androidx.room.Query
import androidx.room.Delete
import kotlinx.coroutines.flow.Flow

@Dao
interface ImageDao {
    @Insert
    fun insert(image: ImageEntity): Long

    @Query("SELECT * FROM images WHERE id = :imageId")
    fun getImageById(imageId: Long): ImageEntity?

    @Query("SELECT * FROM images WHERE tempatWisataId =
:firestoreId")
    fun getImageByTempatWisataId(firestoreId: String): ImageEntity?

    @Query("SELECT * FROM images")
    fun getAllImages(): Flow<List<ImageEntity>>

    @Delete
    fun delete(image: ImageEntity)
}
```

D. Membuat Database dengan Room

7. Buat AppDatabase.kt

```
import androidx.room.Database
import androidx.room.RoomDatabase

@Database(entities = [ImageEntity::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun imageDao(): ImageDao
}
```

E. Mengakses Room Database dan Mengunggah ke Firestore

8. Simpan gambar ke local database lalu simpan direktori tersebut ke firestore

```
fun uploadImageToFirestore(
    firestore: FirebaseFirestore,
    context: Context,
    imageUri: Uri,
    tempatWisata: TempatWisata,
    onSuccess: (TempatWisata) -> Unit,
    onFailure: (Exception) -> Unit
) {
    val db = Room.databaseBuilder(
        context,
        AppDatabase::class.java, "travelupa-database"
    ).build()

    val imageDao = db.imageDao()

    val localPath = saveImageLocally(context, imageUri)

    CoroutineScope(Dispatchers.IO).launch {
        val imageId = imageDao.insert(ImageEntity(localPath =
localPath))

        val updatedTempatWisata = tempatWisata.copy(gambarUriString
= localPath)
        firestore.collection("tempat_wisata")
            .add(updatedTempatWisata)
            .addOnSuccessListener {
                onSuccess(updatedTempatWisata)
            }
            .addOnFailureListener { e ->
                onFailure(e)
            }
    }
}


fun saveImageLocally(context: Context, uri: Uri): String {
    try {
        val inputStream =
context.contentResolver.openInputStream(uri)
```

```
        val file = File(context.filesDir,
"image_${System.currentTimeMillis()}.jpg")

        inputStream?.use { input ->
            file.outputStream().use { output ->
                input.copyTo(output)
            }
        }

        Log.d("ImageSave", "Image saved successfully to:
${file.absolutePath}")
        return file.absolutePath
    } catch (e: Exception) {
        Log.e("ImageSave", "Error saving image", e)
        throw e
    }
}
```

9. Coba run, jika terdapat error silahkan cek halaman improvisasi kode
10. Coba buat tombol hapus dapat benar-benar menghapus data dari firebase

Bab 9 CameraX

Tujuan	Mahasiswa dapat memahami dan mengimplementasikan **CameraX** untuk menangani fungsionalitas kamera di aplikasi Android, dengan menyediakan antarmuka yang lebih sederhana dan lebih fleksibel dibandingkan API kamera lama.

Apa itu CameraX

CameraX adalah API kamera modern yang dikembangkan oleh Android untuk memberikan cara yang lebih sederhana, konsisten, dan fleksibel dalam mengakses kamera di perangkat Android. CameraX dirancang untuk bekerja dengan berbagai perangkat Android, menyederhanakan integrasi fungsionalitas kamera dalam aplikasi, dan mendukung berbagai fitur, seperti mengambil gambar, merekam video, dan menggunakan pengolahan gambar real-time.

Keuntungan menggunakan CameraX:

- Pengembangan yang lebih mudah: Menyederhanakan integrasi kamera dengan API yang lebih mudah digunakan.
- Dukungan untuk berbagai perangkat: CameraX mendukung berbagai macam perangkat, dari perangkat lama hingga perangkat terbaru.
- Kinerja yang lebih baik: Optimasi untuk menangani berbagai pengaturan kamera dan pengolahan gambar secara real-time.
- Kompatibilitas dengan Jetpack: Terintegrasi dengan komponen Jetpack lainnya, seperti LiveData dan ViewModel.

Langkah-langkah :

A. Menyiapkan CameraX ke dalam Proyek Android

1. Untuk menggunakan CameraX di Android, Anda perlu menambahkan dependensi pada file `build.gradle` (app level):

```
dependencies {
      implementation("androidx.camera:camera-core:1.1.0")
      implementation("androidx.camera:camera-camera2:1.1.0")
      implementation("androidx.camera:camera-lifecycle:1.1.0")
      implementation("androidx.camera:camera-view:1.0.0-alpha30")
      implementation("androidx.camera:camera-extensions:1.0.0-
      alpha30")
}
```

2. Sinkronisasi gradle

B. Konfigurasi CameraX

3. Buat halaman galeri dan tambahkan opsi untuk mengunggah file local atau menggunakan kamera

```
@OptIn(ExperimentalFoundationApi::class)
@Composable
fun GalleryScreen(
    imageDao: ImageDao,
    onImageSelected: (Uri) -> Unit,
    onBack: () -> Unit
) {
    val images by imageDao.getAllImages().collectAsState(initial =
emptyList())
    var showAddImageDialog by remember { mutableStateOf(false) }
    var selectedImageEntity by remember {
mutableStateOf<ImageEntity?>(null) }
    val context = LocalContext.current
    var showDeleteConfirmation by remember {
mutableStateOf<ImageEntity?>(null) }

    LaunchedEffect(images) {
        Log.d("GalleryScreen", "Total images: ${images.size}")
        images.forEachIndexed { index, image ->
            Log.d("GalleryScreen", "Image $index path:
${image.localPath}")
            val file = File(image.localPath)
            Log.d("GalleryScreen", "File exists: ${file.exists()},
is readable: ${file.canRead()}")
        }
    }

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text("Gallery") },
                navigationIcon = {
                    IconButton(onClick = onBack) {
                        Icon(Icons.Filled.ArrowBack,
contentDescription = "Back")
                    }
                }
            )
        },
        floatingActionButton = {
            FloatingActionButton(
                onClick = { showAddImageDialog = true },
                backgroundColor = MaterialTheme.colors.primary
            ) {
                Icon(Icons.Filled.Add, contentDescription = "Add
Image")
            }
        }
```

```
    ) { paddingValues ->
        LazyVerticalGrid(
            columns = GridCells.Fixed(3),
            modifier = Modifier.padding(paddingValues)
        ) {
            items(images) { image ->
                Image(
                    painter = rememberAsyncImagePainter(
                        model = image.localPath
                    ),
                    contentDescription = null,
                    modifier = Modifier
                        .size(100.dp)
                        .padding(4.dp)
                        .clickable {
                            selectedImageEntity = image

onImageSelected(Uri.parse(image.localPath))
                        },
                    contentScale = ContentScale.Crop
                )
            }
        }

        if (showAddImageDialog) {
            AddImageDialog(
                onDismiss = { showAddImageDialog = false },
                onImageAdded = { uri ->
                    try {
                        val localPath = saveImageLocally(context,
uri)
                        val newImage = ImageEntity(localPath =
localPath)
                        CoroutineScope(Dispatchers.IO).launch {
                            imageDao.insert(newImage)
                        }
                        showAddImageDialog = false
                    } catch (e: Exception) {
                        Log.e("ImageSave", "Failed to save image",
e)
                    }
                }
            )
        }

        selectedImageEntity?.let { imageEntity ->
            ImageDetailDialog(
                imageEntity = imageEntity,
                onDismiss = { selectedImageEntity = null },
                onDelete = { imageToDelete ->
                    showDeleteConfirmation = imageToDelete
                }
```

```
            )
        }

        showDeleteConfirmation?.let { imageToDelete ->
            AlertDialog(
                onDismissRequest = { showDeleteConfirmation = null
},
                title = { Text("Delete Image") },
                text = { Text("Are you sure you want to delete this
image?") },
                confirmButton = {
                    TextButton(
                        onClick = {
                            CoroutineScope(Dispatchers.IO).launch {
                                imageDao.delete(imageToDelete)
                                val file =
File(imageToDelete.localPath)
                                if (file.exists()) {
                                    file.delete()
                                }
                            }
                            showDeleteConfirmation = null
                        }
                    ) {
                        Text("Delete")
                    }
                },
                dismissButton = {
                    TextButton(
                        onClick = { showDeleteConfirmation = null }
                    ) {
                        Text("Cancel")
                    }
                }
            )
        }
    }
}
```

4. Dalam halaman galeri, buat floating button

```
@Composable
fun AddImageDialog(
    onDismiss: () -> Unit,
    onImageAdded: (Uri) -> Unit
) {
    var imageUri by remember { mutableStateOf<Uri?>(null) }
    val context = LocalContext.current

    val imagePickerLauncher = rememberLauncherForActivityResult(
```

```
        contract = ActivityResultContracts.GetContent()
    ) { uri: Uri? ->
        imageUri = uri
    }

    val cameraLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.TakePicturePreview()
    ) { bitmap ->
        bitmap?.let {
            val uri = saveBitmapToUri(context, it)
            imageUri = uri
        }
    }

    AlertDialog(
        onDismissRequest = onDismiss,
        title = { Text("Add New Image") },
        text = {
            Column {
                imageUri?.let { uri ->
                    Image(
                        painter = rememberAsyncImagePainter(model =
uri),
                        contentDescription = "Selected Image",
                        modifier = Modifier
                            .fillMaxWidth()
                            .height(200.dp),
                        contentScale = ContentScale.Crop
                    )
                }
                Spacer(modifier = Modifier.height(8.dp))
                Row {
                    Button(
                        onClick = {
imagePickerLauncher.launch("image/*") },
                        modifier = Modifier.weight(1f)
                    ) {
                        Text("Select from File")
                    }
                    Spacer(modifier = Modifier.width(8.dp))
                    Button(
                        onClick = { cameraLauncher.launch(null) },
                        modifier = Modifier.weight(1f)
                    ) {
                        Text("Take Photo")
                    }
                }
            }
        },
        confirmButton = {
            Button(
                onClick = {
```

```
                    imageUri?.let { uri ->
                        onImageAdded(uri)
                    }
                }
            ) {
                Text("Add")
            }
        },
        dismissButton = {
            Button(onClick = onDismiss) {
                Text("Cancel")
            }
        }
    )
}
```

5. Tambahkan function agar dapat melihat detail gambar

```
@Composable
fun ImageDetailDialog(
    imageEntity: ImageEntity,
    onDismiss: () -> Unit,
    onDelete: (ImageEntity) -> Unit
) {
    AlertDialog(
        onDismissRequest = onDismiss,
        text = {
            Image(
                painter = rememberAsyncImagePainter(model =
imageEntity.localPath),
                contentDescription = "Detailed Image",
                modifier = Modifier
                    .fillMaxWidth()
                    .aspectRatio(1f),
                contentScale = ContentScale.Crop
            )
        },
        confirmButton = {
            Row {
                Button(onClick = { onDelete(imageEntity) }) {
                    Text("Delete")
                }
                Spacer(modifier = Modifier.width(8.dp))
                Button(onClick = onDismiss) {
                    Text("Close")
                }
            }
        }
    )
}
```

```
fun saveBitmapToUri(context: Context, bitmap: Bitmap): Uri {
    val file = File(context.cacheDir, "${UUID.randomUUID()}.jpg")
    val outputStream = FileOutputStream(file)
    bitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream)
    outputStream.close()
    return Uri.fromFile(file)
}
```

6. Coba run, jika menemukan error silahkan lihat halaman improvisasi kode.

Improvisasi Kode

Tujuan	Halaman ini digunakan sebagai referensi utama untuk melihat kesalahan kode yang mungkin terjadi

1. Gradle.kts (app level)

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.jetbrains.kotlin.android)
    id("com.google.gms.google-services") version "4.4.2" apply
false
    kotlin("kapt")
}

android {
    namespace = "com.example.travelupa"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.example.travelupa"
        minSdk = 24
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner =
"androidx.test.runner.AndroidJUnitRunner"
        vectorDrawables {
            useSupportLibrary = true
        }
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-
optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = "1.8"
    }
```

```
    buildFeatures {
        compose = true
    }
    composeOptions {
        kotlinCompilerExtensionVersion = "1.5.1"
    }
    packaging {
        resources {
            excludes += "/META-INF/{AL2.0,LGPL2.1}"
        }
    }
}

dependencies {

    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.activity.compose)
    implementation(platform(libs.androidx.compose.bom))
    implementation(libs.androidx.ui)
    implementation(libs.androidx.ui.graphics)
    implementation(libs.androidx.ui.tooling.preview)
    implementation(libs.androidx.material3)
    implementation("androidx.activity:activity-compose:1.7.2")
    implementation("androidx.compose.ui:ui:1.6.0")
    implementation("androidx.compose.material:material:1.6.0")
    implementation("androidx.compose.material3:material3:1.0.0")
    implementation("androidx.compose.ui:ui-tooling-preview:1.6.0")
    implementation("androidx.compose.material:material-icons-
extended:1.5.4")
    implementation("androidx.core:core-ktx:1.10.1")
    implementation("androidx.compose.foundation:foundation:1.5.3")
    implementation("androidx.lifecycle:lifecycle-runtime-
ktx:2.6.1")
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-
core:1.7.3")
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-
android:1.7.3")
    implementation("io.coil-kt:coil-compose:2.4.0")
    implementation(platform("com.google.firebase:firebase-
bom:33.6.0"))
    implementation("com.google.firebase:firebase-analytics")
    implementation("com.google.firebase:firebase-auth")
    implementation("androidx.navigation:navigation-compose:2.5.3")
    implementation("com.google.firebase:firebase-firestore-ktx")
    implementation("com.google.firebase:firebase-storage-ktx")
    implementation("androidx.room:room-runtime:2.5.0")
    kapt("androidx.room:room-compiler:2.5.0")
    implementation("androidx.room:room-ktx:2.5.0")
    implementation("androidx.camera:camera-core:1.1.0")
    implementation("androidx.camera:camera-camera2:1.1.0")
    implementation("androidx.camera:camera-lifecycle:1.1.0")
```

```
    implementation("androidx.camera:camera-view:1.0.0-alpha30")
    implementation("androidx.camera:camera-extensions:1.0.0-
alpha30")
    implementation("com.google.accompanist:accompanist-
flowlayout:0.24.13-rc")
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    androidTestImplementation(platform(libs.androidx.compose.bom))
    androidTestImplementation(libs.androidx.ui.test.junit4)
    debugImplementation(libs.androidx.ui.tooling)
    debugImplementation(libs.androidx.ui.test.manifest)
    androidTestImplementation("androidx.test.espresso:espresso-
core:3.5.1")
    androidTestImplementation("androidx.test.espresso:espresso-
intents:3.5.1")
    androidTestImplementation("androidx.test.espresso:espresso-
contrib:3.5.1")
}

apply(plugin = "com.google.gms.google-services")

kapt {
    correctErrorTypes = true
}
```

2. ImageEntity.kt

```
package com.example.travelupa

import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "images")
data class ImageEntity(
    @PrimaryKey(autoGenerate = true)
    val id: Long = 0,
    val localPath: String,
    val tempatWisataId: String? = null
)
```

3. ImageDao.kt

```
package com.example.travelupa

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.Query
import androidx.room.Delete
```

```
import kotlinx.coroutines.flow.Flow

@Dao
interface ImageDao {
    @Insert
    fun insert(image: ImageEntity): Long

    @Query("SELECT * FROM images WHERE id = :imageId")
    fun getImageById(imageId: Long): ImageEntity?

    @Query("SELECT * FROM images WHERE tempatWisataId =
:firestoreId")
    fun getImageByTempatWisataId(firestoreId: String): ImageEntity?

    @Query("SELECT * FROM images")
    fun getAllImages(): Flow<List<ImageEntity>>

    @Delete
    fun delete(image: ImageEntity)
}
```

4. AppDatabase.kt

```
package com.example.travelupa

import androidx.room.Database
import androidx.room.RoomDatabase

@Database(entities = [ImageEntity::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun imageDao(): ImageDao
}
```

5. MainActivity.kt

```
package com.example.travelupa

import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.activity.ComponentActivity
import android.os.Bundle
import android.util.Log
import androidx.activity.compose.setContent
import androidx.compose.ui.text.style.TextAlign
```

```
import com.example.travelupa.ui.theme.TravelupaTheme
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.background
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.runtime.*
import androidx.compose.ui.res.painterResource
import androidx.compose.foundation.Image
import androidx.compose.ui.layout.ContentScale
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Add
import androidx.compose.material.icons.filled.MoreVert
import androidx.compose.material.icons.filled.ArrowBack
import androidx.compose.foundation.ExperimentalFoundationApi
import androidx.compose.foundation.layout.ExperimentalLayoutApi
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.grid.GridCells
import androidx.compose.foundation.lazy.grid.LazyVerticalGrid
import androidx.compose.foundation.lazy.grid.items
import androidx.compose.material3.LocalTextStyle
import android.net.Uri
import androidx.compose.ui.platform.LocalContext
import coil.compose.rememberAsyncImagePainter
import coil.request.ImageRequest
import androidx.compose.foundation.clickable
import androidx.activity.compose.rememberLauncherForActivityResult
import androidx.activity.result.contract.ActivityResultContracts
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.FirebaseUser
import com.google.firebase.storage.FirebaseStorage
import androidx.compose.runtime.getValue
import androidx.compose.runtime.setValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.material3.Text
import androidx.compose.material3.TextField
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material.icons.filled.Email
import androidx.compose.material.icons.outlined.Visibility
import androidx.compose.material.icons.outlined.VisibilityOff
import androidx.compose.material.icons.filled.Menu
import androidx.compose.material.icons.filled.Delete
import androidx.compose.material.icons.filled.Close
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.text.input.VisualTransformation
import androidx.compose.ui.layout.onGloballyPositioned
import androidx.compose.ui.layout.positionInWindow
```

```
import
com.google.firebase.auth.FirebaseAuthInvalidCredentialsException
import com.google.firebase.auth.FirebaseAuthInvalidUserException
import com.google.accompanist.flowlayout.FlowRow
import com.google.accompanist.flowlayout.SizeMode
import androidx.navigation.NavController
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import androidx.compose.ui.unit.DpOffset
import android.graphics.Bitmap
import androidx.compose.ui.geometry.Offset
import com.google.firebase.FirebaseApp
import com.google.firebase.firestore.FirebaseFirestore
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import kotlinx.coroutines.withContext
import kotlinx.coroutines.tasks.await
import java.util.UUID
import android.content.Context
import androidx.room.Room
import java.io.File
import java.io.FileOutputStream
import java.io.InputStream

sealed class Screen(val route: String) {
    object Greeting : Screen("greeting")
    object Login : Screen("login")
    object RekomendasiTempat : Screen("rekomendasi_tempat")
}

class MainActivity : ComponentActivity() {
    private lateinit var firestore: FirebaseFirestore
    private lateinit var storage: FirebaseStorage
    private lateinit var imageDao: ImageDao

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

        firestore = FirebaseFirestore.getInstance()
        storage = FirebaseStorage.getInstance()

        val db = Room.databaseBuilder(
            applicationContext,
            AppDatabase::class.java, "travelupa-database"
        ).build()
        imageDao = db.imageDao()
        val currentUser: FirebaseUser? =
FirebaseAuth.getInstance().currentUser
```

```
        setContent {
            TravelupaTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = Color.White
                ) {
                    AppNavigation(currentUser, firestore, storage,
imageDao)
                }
            }
        }
    }
}

@Composable
fun AppNavigation(
    currentUser: FirebaseUser?,
    firestore: FirebaseFirestore,
    storage: FirebaseStorage,
    imageDao: ImageDao
) {
    val navController = rememberNavController()

    NavHost(
        navController = navController,
        startDestination = if (currentUser != null)
Screen.RekomendasiTempat.route else Screen.Greeting.route
    ) {
        composable(Screen.Greeting.route) {
            GreetingScreen(
                onStart = {
                    navController.navigate(Screen.Login.route) {
                        popUpTo(Screen.Greeting.route) { inclusive
= true }
                    }
                }
            )
        }
        composable(Screen.Login.route) {
            LoginScreen(
                onLoginSuccess = {

navController.navigate(Screen.RekomendasiTempat.route) {
                        popUpTo(Screen.Login.route) { inclusive =
true }
                    }
                }
            )
        }
        composable(Screen.RekomendasiTempat.route) {
            RekomendasiTempatScreen(
                firestore = firestore,
```

```
                onBackToLogin = {
                    FirebaseAuth.getInstance().signOut()
                    navController.navigate(Screen.Login.route) {
                        popUpTo(Screen.RekomendasiTempat.route) {
inclusive = true }
                    }
                },
                onGallerySelected = {
                    navController.navigate("gallery")
                }
            )
        }
        composable("gallery") {
            GalleryScreen(
                imageDao = imageDao,
                onImageSelected = { uri ->
                    // Handle image selection
                },
                onBack = {
                    navController.popBackStack()
                }
            )
        }
    }
}

@Composable
fun LoginScreen(
    onLoginSuccess: () -> Unit
) {
    var email by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var isPasswordVisible by remember { mutableStateOf(false) }
    var errorMessage by remember { mutableStateOf<String?>(null) }
    var isLoading by remember { mutableStateOf(false) }

    val coroutineScope = rememberCoroutineScope()

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center
    ) {
        TextField(
            value = email,
            onValueChange = {
                email = it
                errorMessage = null
            },
            label = { Text("Email") },
            modifier = Modifier.fillMaxWidth(),
```

```
            keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Email)
        )

        Spacer(modifier = Modifier.height(8.dp))

        TextField(
            value = password,
            onValueChange = {
                password = it
                errorMessage = null
            },
            label = { Text("Password") },
            visualTransformation = if (isPasswordVisible)
VisualTransformation.None else PasswordVisualTransformation(),
            trailingIcon = {
                IconButton(onClick = { isPasswordVisible =
!isPasswordVisible }) {
                    Icon(
                        imageVector = if (isPasswordVisible)
Icons.Outlined.VisibilityOff else Icons.Outlined.Visibility,
                        contentDescription = if (isPasswordVisible)
"Hide password" else "Show password"
                    )
                }
            },
            modifier = Modifier.fillMaxWidth(),
            keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Password)
        )

        Spacer(modifier = Modifier.height(16.dp))

        Button(
            onClick = {
                if (email.isBlank() || password.isBlank()) {
                    errorMessage = "Please enter email and
password"
                    return@Button
                }

                isLoading = true
                errorMessage = null

                coroutineScope.launch {
                    try {
                        val authResult =
withContext(Dispatchers.IO) {

FirebaseAuth.getInstance().signInWithEmailAndPassword(email,
password).await()
                        }
```

```
                        isLoading = false
                        onLoginSuccess()
                    } catch (e: Exception) {
                        isLoading = false
                        errorMessage = "Login failed:
${e.localizedMessage}"
                    }
                }
            },
            modifier = Modifier.fillMaxWidth(),
            enabled = !isLoading
        ) {
            if (isLoading) {
                CircularProgressIndicator(
                    modifier = Modifier.size(20.dp),
                    color = Color.White
                )
            } else {
                Text("Login")
            }
        }
        errorMessage?.let {
            Text(
                text = it,
                color = Color.Red,
                modifier = Modifier.padding(top = 8.dp)
            )
        }
    }
}

@Composable
fun GreetingScreen(
    onStart: () -> Unit
) {
    Box(
        modifier = Modifier.fillMaxSize(),
        contentAlignment = Alignment.Center,
    ) {
        Column(
            horizontalAlignment = Alignment.CenterHorizontally,
            modifier = Modifier
                .fillMaxWidth()
                .padding(16.dp)
        ) {
            Text(
                text = "Selamat Datang di Travelupa!",
                style = MaterialTheme.typography.h4,
                textAlign = TextAlign.Center
            )

            Spacer(modifier = Modifier.height(16.dp))
```

```
            Text(
                text = "Solusi buat kamu yang lupa kemana-mana",
                style = MaterialTheme.typography.h6
            )

        }

        Button(
            onClick = onStart,
            modifier = Modifier
                .width(360.dp)
                .align(Alignment.BottomCenter)
                .padding(bottom = 16.dp)
        ) {
            Text(text = "Mulai")
        }
    }
}

data class TempatWisata(
    val nama: String = "",
    val deskripsi: String = "",
    val gambarUriString: String? = null,
    val gambarResId: Int? = null
)

@Composable
fun RekomendasiTempatScreen(
    firestore: FirebaseFirestore,
    onBackToLogin: (() -> Unit)? = null,
    onGallerySelected: () -> Unit
) {
    var daftarTempatWisata by remember {
mutableStateOf(listOf<TempatWisata>()) }
    var showTambahDialog by remember { mutableStateOf(false) }
    var drawerState = rememberDrawerState(DrawerValue.Closed)
    val coroutineScope = rememberCoroutineScope()

    LaunchedEffect(Unit) {
        val tempatWisataList = mutableListOf<TempatWisata>()
        firestore.collection("tempat_wisata")
            .get()
            .addOnSuccessListener { result ->
                for (document in result) {
                    val tempatWisata =
document.toObject(TempatWisata::class.java)
                    tempatWisataList.add(tempatWisata)
                }
                daftarTempatWisata = tempatWisataList
            }
            .addOnFailureListener { exception ->
```

```
                // Handle error
            }
    }

    ModalDrawer(
        drawerState = drawerState,
        drawerContent = {
            Column {
                Text(
                    text = "Menu",
                    style = MaterialTheme.typography.h6,
                    modifier = Modifier.padding(16.dp)
                )
                Divider()
                Text(
                    text = "Gallery",
                    modifier = Modifier
                        .fillMaxWidth()
                        .clickable { onGallerySelected() }
                        .padding(16.dp)
                )
                Text(
                    text = "Logout",
                    modifier = Modifier
                        .fillMaxWidth()
                        .clickable {
                            coroutineScope.launch {
                                drawerState.close()
                            }
                            onBackToLogin?.invoke()
                        }
                        .padding(16.dp)
                )
            }
        }
    ) {
        Scaffold(
            topBar = {
                TopAppBar(
                    title = {
                        Box(
                        ) {
                            Text(
                                text = "Rekomendasi Tempat Wisata",
                                style =
MaterialTheme.typography.h6,
                                color = Color.White
                            )
                        }
                    },
                    navigationIcon = {
                        IconButton(onClick = {
```

```
                            coroutineScope.launch {
                                drawerState.open()
                            }
                        }) {
                            Icon(Icons.Filled.Menu,
contentDescription = "Menu")
                        }
                    }
                )
            },
            floatingActionButton = {
                FloatingActionButton(
                    onClick = { showTambahDialog = true },
                    backgroundColor = MaterialTheme.colors.primary
                ) {
                    Icon(Icons.Filled.Add, contentDescription =
"Tambah Tempat Wisata")
                }
            }
        ) { paddingValues ->
            Column(
                modifier = Modifier
                    .padding(paddingValues)
                    .padding(16.dp)
            ) {
                LazyColumn {
                    items(daftarTempatWisata) { tempat ->
                        TempatItemEditable(
                            tempat = tempat,
                            onDelete = {
                                daftarTempatWisata =
daftarTempatWisata.filter { it != tempat }
                            }
                        )
                    }
                }
            }

            if (showTambahDialog) {
                TambahTempatWisataDialog(
                    firestore = firestore,
                    context = LocalContext.current,
                    onDismiss = { showTambahDialog = false },
                    onTambah = { nama, deskripsi, gambarUri ->
                        val uriString = gambarUri?.toString() ?: ""
                        val nuevoTempat = TempatWisata(nama,
deskripsi, uriString)
                        daftarTempatWisata = daftarTempatWisata +
nuevoTempat
                        showTambahDialog = false
                    }
                )
```

```
            }
        }
    }
}

@Composable
fun TempatItemEditable(
    tempat: TempatWisata,
    onDelete: () -> Unit
) {
    val firestore = FirebaseFirestore.getInstance()
    var expanded by remember { mutableStateOf(false) }

    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(vertical = 8.dp)
            .background(MaterialTheme.colors.surface),
        elevation = 4.dp
    ) {
        Column(modifier = Modifier.padding(16.dp)) {
            Image(
                painter = tempat.gambarUriString?.let { uriString -
>
                    rememberAsyncImagePainter(
                        ImageRequest.Builder(LocalContext.current)
                            .data(Uri.parse(uriString))
                            .build()
                    )
                } ?: tempat.gambarResId?.let {
                    painterResource(id = it)
                } ?: painterResource(id =
R.drawable.default_image),
                contentDescription = tempat.nama,
                modifier = Modifier
                    .fillMaxWidth()
                    .height(200.dp),
                contentScale = ContentScale.Crop
            )
            Box(modifier = Modifier.fillMaxWidth()) {
                Column(modifier = Modifier
                    .align(Alignment.CenterStart)
                ) {
                    Text(
                        text = tempat.nama,
                        style = MaterialTheme.typography.h6,
                        modifier = Modifier.padding(bottom = 8.dp,
top = 12.dp)
                    )
                    Text(
                        text = tempat.deskripsi,
                        style = MaterialTheme.typography.body2,
```

```
                        modifier = Modifier.padding(top = 4.dp)
                    )
                }
                IconButton(
                    onClick = { expanded = true },
                    modifier = Modifier.align(Alignment.TopEnd)
                ) {
                    Icon(Icons.Default.MoreVert, contentDescription
= "More options")
                }
                DropdownMenu(
                    expanded = expanded,
                    onDismissRequest = { expanded = false },
                    offset = DpOffset(250.dp, 0.dp),
                ) {
                    DropdownMenuItem(onClick = {
                        expanded = false

firestore.collection("tempat_wisata").document(tempat.nama)
                            .delete()
                            .addOnSuccessListener {
                                onDelete()
                            }
                            .addOnFailureListener { e ->
                                Log.w("TempatItemEditable", "Error
deleting document", e)
                            }
                    }) {
                        Text("Delete")
                    }
                }
            }
        }
    }
}

fun uploadImageToFirestore(
    firestore: FirebaseFirestore,
    context: Context,
    imageUri: Uri,
    tempatWisata: TempatWisata,
    onSuccess: (TempatWisata) -> Unit,
    onFailure: (Exception) -> Unit
) {
    val db = Room.databaseBuilder(
        context,
        AppDatabase::class.java, "travelupa-database"
    ).build()

    val imageDao = db.imageDao()

    val localPath = saveImageLocally(context, imageUri)
```

```
    CoroutineScope(Dispatchers.IO).launch {
        val imageId = imageDao.insert(ImageEntity(localPath =
localPath))

        val updatedTempatWisata = tempatWisata.copy(gambarUriString
= localPath)
        firestore.collection("tempat_wisata")
            .add(updatedTempatWisata)
            .addOnSuccessListener {
                onSuccess(updatedTempatWisata)
            }
            .addOnFailureListener { e ->
                onFailure(e)
            }
    }
}

fun saveImageLocally(context: Context, uri: Uri): String {
    try {
        val inputStream =
context.contentResolver.openInputStream(uri)
        val file = File(context.filesDir,
"image_${System.currentTimeMillis()}.jpg")

        inputStream?.use { input ->
            file.outputStream().use { output ->
                input.copyTo(output)
            }
        }

        Log.d("ImageSave", "Image saved successfully to:
${file.absolutePath}")
        return file.absolutePath
    } catch (e: Exception) {
        Log.e("ImageSave", "Error saving image", e)
        throw e
    }
}

@Composable
fun TambahTempatWisataDialog(
    firestore: FirebaseFirestore,
    context: Context,
    onDismiss: () -> Unit,
    onTambah: (String, String, String?) -> Unit
) {
    var nama by remember { mutableStateOf("") }
    var deskripsi by remember { mutableStateOf("") }
    var gambarUri by remember { mutableStateOf<Uri?>(null) }
    var isUploading by remember { mutableStateOf(false) }
```

```
    val gambarLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.GetContent()
    ) { uri: Uri? ->
        gambarUri = uri
    }

    AlertDialog(
        onDismissRequest = onDismiss,
        title = { Text("Tambah Tempat Wisata Baru") },
        text = {
            Column {
                TextField(
                    value = nama,
                    onValueChange = { nama = it },
                    label = { Text("Nama Tempat") },
                    modifier = Modifier.fillMaxWidth(),
                    enabled = !isUploading
                )
                Spacer(modifier = Modifier.height(8.dp))
                TextField(
                    value = deskripsi,
                    onValueChange = { deskripsi = it },
                    label = { Text("Deskripsi") },
                    modifier = Modifier.fillMaxWidth(),
                    enabled = !isUploading
                )
                Spacer(modifier = Modifier.height(8.dp))
                gambarUri?.let { uri ->
                    Image(
                        painter = rememberAsyncImagePainter(model =
uri),
                        contentDescription = "Gambar yang dipilih",
                        modifier = Modifier
                            .fillMaxWidth()
                            .height(200.dp),
                        contentScale = ContentScale.Crop
                    )
                }
                Spacer(modifier = Modifier.height(8.dp))
                Button(
                    onClick = { gambarLauncher.launch("image/*") },
                    modifier = Modifier.fillMaxWidth(),
                    enabled = !isUploading
                ) {
                    Text("Pilih Gambar")
                }
            }
        },
        confirmButton = {
            Button(
                onClick = {
                    if (nama.isNotBlank() && deskripsi.isNotBlank()
```

```
&& gambarUri != null) {
                        isUploading = true
                        val tempatWisata = TempatWisata(nama,
deskripsi)

                        uploadImageToFirestore(
                            firestore,
                            context,
                            gambarUri!!,
                            tempatWisata,
                            onSuccess = { uploadedTempat ->
                                isUploading = false
                                onTambah(nama, deskripsi,
uploadedTempat.gambarUriString)
                                onDismiss()
                            },
                            onFailure = { e ->
                                isUploading = false
                            }
                        )
                    }
                },
                enabled = !isUploading
            ) {
                if (isUploading) {
                    CircularProgressIndicator(
                        modifier = Modifier.size(20.dp),
                        color = Color.White
                    )
                } else {
                    Text("Tambah")
                }
            }
        },
        dismissButton = {
            Button(
                onClick = onDismiss,
                colors =
ButtonDefaults.buttonColors(backgroundColor =
MaterialTheme.colors.surface),
                enabled = !isUploading
            ) {
                Text("Batal")
            }
        }
    )
}

@OptIn(ExperimentalFoundationApi::class)
@Composable
fun GalleryScreen(
    imageDao: ImageDao,
```

```
    onImageSelected: (Uri) -> Unit,
    onBack: () -> Unit
) {
    val images by imageDao.getAllImages().collectAsState(initial =
emptyList())
    var showAddImageDialog by remember { mutableStateOf(false) }
    var selectedImageEntity by remember {
mutableStateOf<ImageEntity?>(null) }
    val context = LocalContext.current
    var showDeleteConfirmation by remember {
mutableStateOf<ImageEntity?>(null) }

    LaunchedEffect(images) {
        Log.d("GalleryScreen", "Total images: ${images.size}")
        images.forEachIndexed { index, image ->
            Log.d("GalleryScreen", "Image $index path:
${image.localPath}")
            val file = File(image.localPath)
            Log.d("GalleryScreen", "File exists: ${file.exists()},
is readable: ${file.canRead()}")
        }
    }

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text("Gallery") },
                navigationIcon = {
                    IconButton(onClick = onBack) {
                        Icon(Icons.Filled.ArrowBack,
contentDescription = "Back")
                    }
                }
            )
        },
        floatingActionButton = {
            FloatingActionButton(
                onClick = { showAddImageDialog = true },
                backgroundColor = MaterialTheme.colors.primary
            ) {
                Icon(Icons.Filled.Add, contentDescription = "Add
Image")
            }
        }
    ) { paddingValues ->
        LazyVerticalGrid(
            columns = GridCells.Fixed(3),
            modifier = Modifier.padding(paddingValues)
        ) {
            items(images) { image ->
                Image(
                    painter = rememberAsyncImagePainter(
```

```
                        model = image.localPath
                    ),
                    contentDescription = null,
                    modifier = Modifier
                        .size(100.dp)
                        .padding(4.dp)
                        .clickable {
                            selectedImageEntity = image

onImageSelected(Uri.parse(image.localPath))
                        },
                    contentScale = ContentScale.Crop
                )
            }
        }

        if (showAddImageDialog) {
            AddImageDialog(
                onDismiss = { showAddImageDialog = false },
                onImageAdded = { uri ->
                    try {
                        val localPath = saveImageLocally(context,
uri)
                        val newImage = ImageEntity(localPath =
localPath)
                        CoroutineScope(Dispatchers.IO).launch {
                            imageDao.insert(newImage)
                        }
                        showAddImageDialog = false
                    } catch (e: Exception) {
                        Log.e("ImageSave", "Failed to save image",
e)
                    }
                }
            )
        }

        selectedImageEntity?.let { imageEntity ->
            ImageDetailDialog(
                imageEntity = imageEntity,
                onDismiss = { selectedImageEntity = null },
                onDelete = { imageToDelete ->
                    showDeleteConfirmation = imageToDelete
                }
            )
        }

        showDeleteConfirmation?.let { imageToDelete ->
            AlertDialog(
                onDismissRequest = { showDeleteConfirmation = null
},
                title = { Text("Delete Image") },
```

```
                text = { Text("Are you sure you want to delete this
image?") },
                confirmButton = {
                    TextButton(
                        onClick = {
                            CoroutineScope(Dispatchers.IO).launch {
                                imageDao.delete(imageToDelete)
                                val file =
File(imageToDelete.localPath)
                                if (file.exists()) {
                                    file.delete()
                                }
                            }
                            showDeleteConfirmation = null
                        }
                    ) {
                        Text("Delete")
                    }
                },
                dismissButton = {
                    TextButton(
                        onClick = { showDeleteConfirmation = null }
                    ) {
                        Text("Cancel")
                    }
                }
            )
        }
    }
}

@Composable
fun AddImageDialog(
    onDismiss: () -> Unit,
    onImageAdded: (Uri) -> Unit
) {
    var imageUri by remember { mutableStateOf<Uri?>(null) }
    val context = LocalContext.current

    val imagePickerLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.GetContent()
    ) { uri: Uri? ->
        imageUri = uri
    }

    val cameraLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.TakePicturePreview()
    ) { bitmap ->
        bitmap?.let {
            val uri = saveBitmapToUri(context, it)
            imageUri = uri
        }
```

```
    }

    AlertDialog(
        onDismissRequest = onDismiss,
        title = { Text("Add New Image") },
        text = {
            Column {
                imageUri?.let { uri ->
                    Image(
                        painter = rememberAsyncImagePainter(model =
uri),
                        contentDescription = "Selected Image",
                        modifier = Modifier
                            .fillMaxWidth()
                            .height(200.dp),
                        contentScale = ContentScale.Crop
                    )
                }
                Spacer(modifier = Modifier.height(8.dp))
                Row {
                    Button(
                        onClick = {
imagePickerLauncher.launch("image/*") },
                        modifier = Modifier.weight(1f)
                    ) {
                        Text("Select from File")
                    }
                    Spacer(modifier = Modifier.width(8.dp))
                    Button(
                        onClick = { cameraLauncher.launch(null) },
                        modifier = Modifier.weight(1f)
                    ) {
                        Text("Take Photo")
                    }
                }
            }
        },
        confirmButton = {
            Button(
                onClick = {
                    imageUri?.let { uri ->
                        onImageAdded(uri)
                    }
                }
            ) {
                Text("Add")
            }
        },
        dismissButton = {
            Button(onClick = onDismiss) {
                Text("Cancel")
            }
```

```
        }
    )
}

@Composable
fun ImageDetailDialog(
    imageEntity: ImageEntity,
    onDismiss: () -> Unit,
    onDelete: (ImageEntity) -> Unit
) {
    AlertDialog(
        onDismissRequest = onDismiss,
        text = {
            Image(
                painter = rememberAsyncImagePainter(model =
imageEntity.localPath),
                contentDescription = "Detailed Image",
                modifier = Modifier
                    .fillMaxWidth()
                    .aspectRatio(1f),
                contentScale = ContentScale.Crop
            )
        },
        confirmButton = {
            Row {
                Button(onClick = { onDelete(imageEntity) }) {
                    Text("Delete")
                }
                Spacer(modifier = Modifier.width(8.dp))
                Button(onClick = onDismiss) {
                    Text("Close")
                }
            }
        }
    )
}

fun saveBitmapToUri(context: Context, bitmap: Bitmap): Uri {
    val file = File(context.cacheDir, "${UUID.randomUUID()}.jpg")
    val outputStream = FileOutputStream(file)
    bitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream)
    outputStream.close()
    return Uri.fromFile(file)
}
```

Bab 10 TestLab Firebase

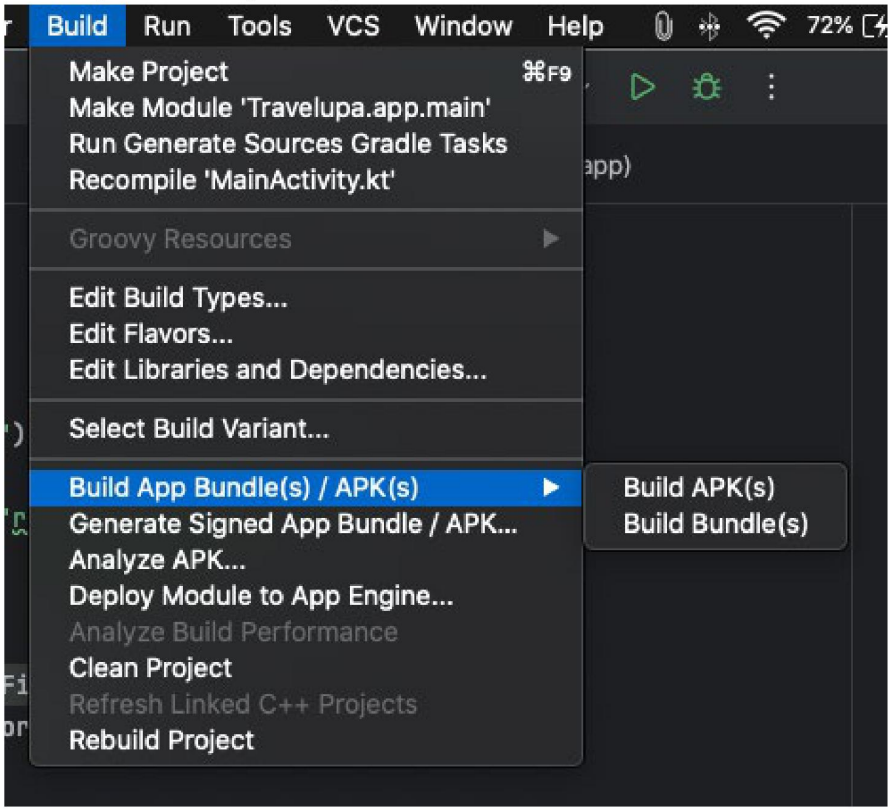
Tujuan	Mahasiswa dapat memahami dan mengimplementasikan **Firebase Test Lab** untuk melakukan pengujian aplikasi Android di berbagai perangkat dan konfigurasi secara otomatis, sehingga dapat memastikan aplikasi berjalan dengan baik di berbagai jenis perangkat.

Apa itu Firebase Test Lab

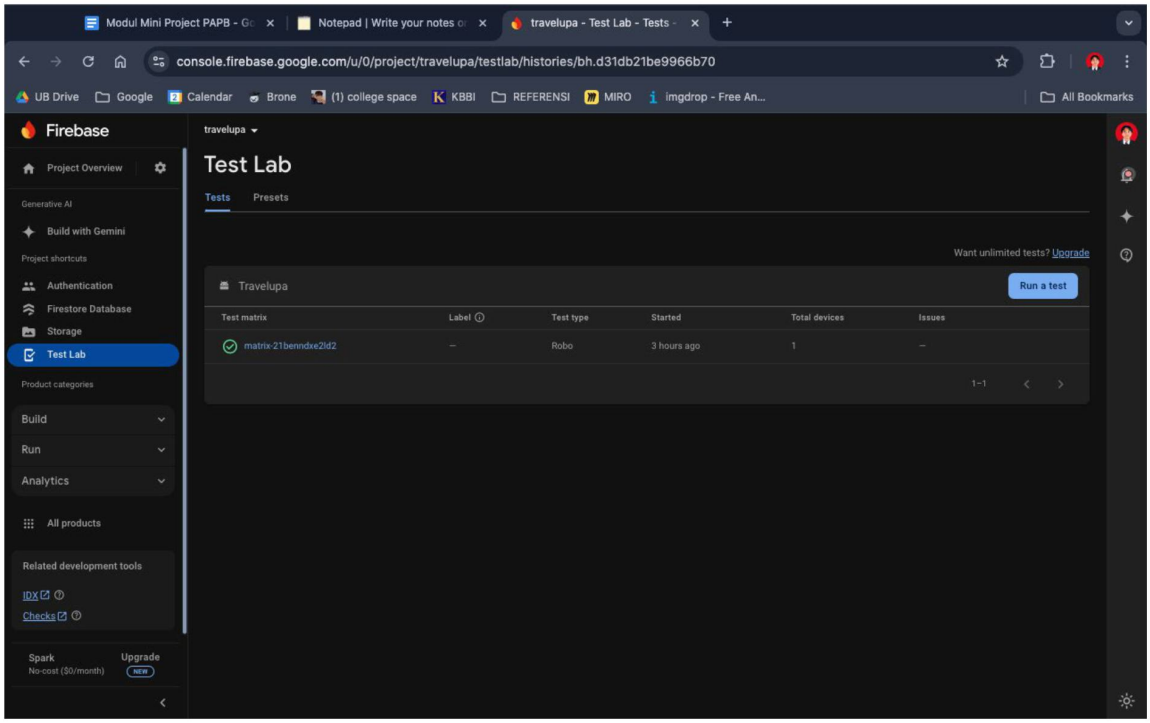
Firebase Test Lab adalah layanan yang disediakan oleh Google Firebase yang memungkinkan pengembang untuk melakukan pengujian aplikasi Android pada berbagai perangkat fisik dan virtual secara otomatis. Test Lab menyediakan akses ke perangkat nyata yang dapat digunakan untuk menjalankan pengujian aplikasi, memastikan aplikasi bekerja dengan baik di berbagai kondisi dan perangkat yang berbeda.

Langkah-langkah :

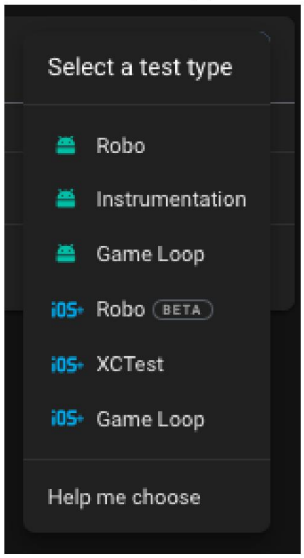
1. Build aplikasi menjadi .apk



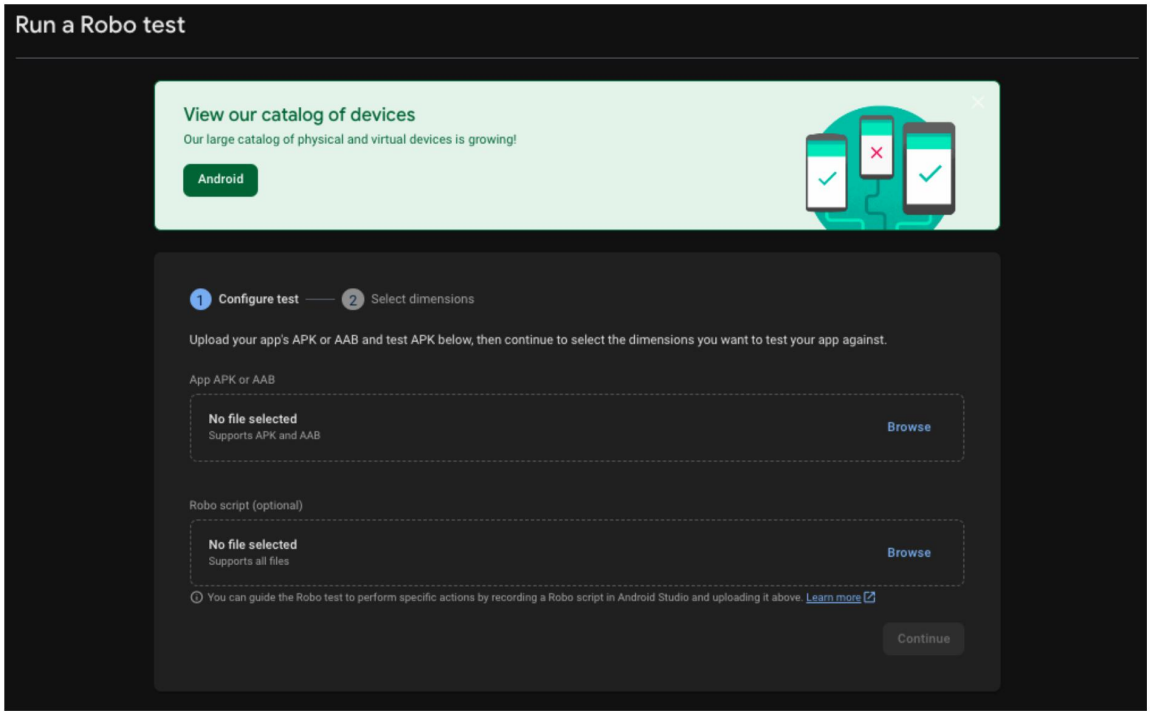
2. Setelah menjadi .apk, kunjungi firebase console dan buka testlab



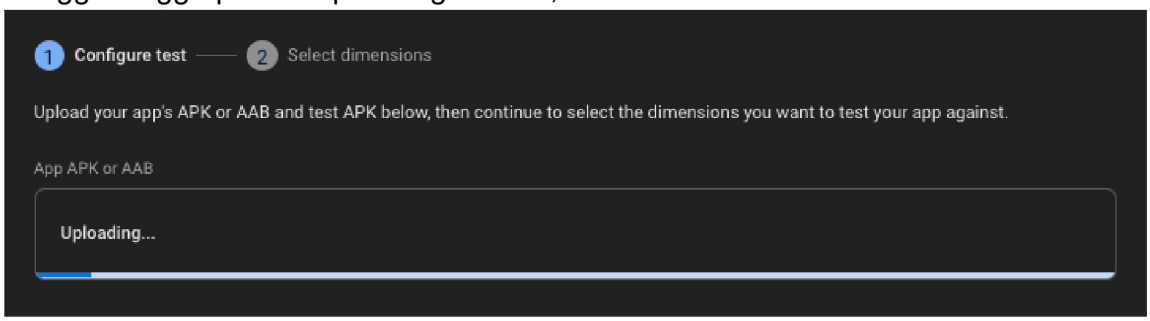
3. Klik run a test, pilih Robo



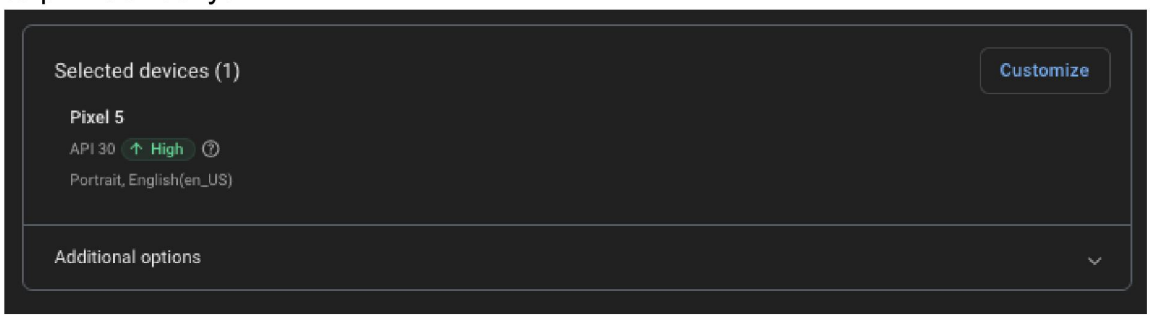
4. Pilih file .apk yang sudah di build sebelumnya. Tampilan ini berbeda dengan ketika pertama kali melakukan pengujian



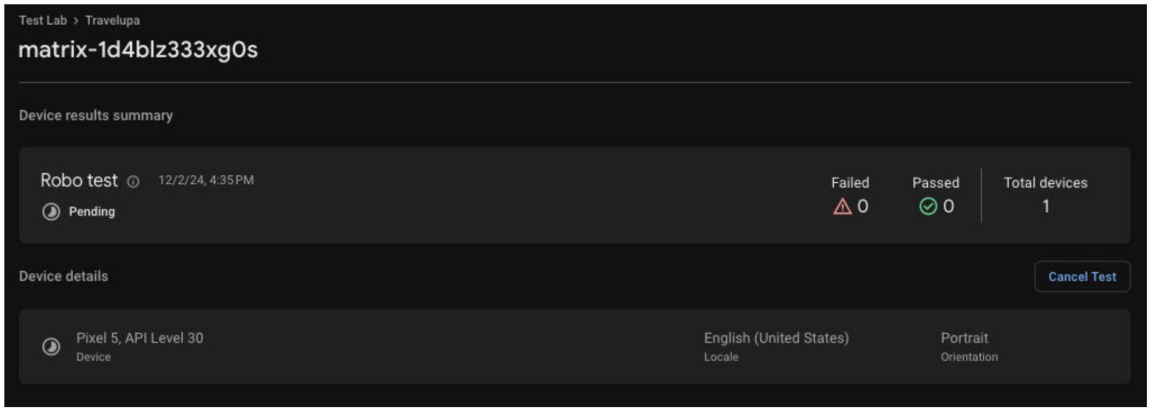
5. Tunggu hingga proses uploading selesai, lalu klik continue



6. Jika pernah melakukan proses testing, maka akan diarahkan untuk memilih device. Namun jika pertama kali melakukan testing maka akan otomatis terpilih devicenya.



7. Jika telah memilih device, maka klik start 1 test



8. Tunggu hingga proses selesai. Biasanya memakan waktu 5-7 menit.
9. Jika telah selesai maka akan muncul hasilnya sebagai berikut.