PENGEMBANGAN APLIKASI CALENDER EVENT BERBASIS FLUTTER DENGAN INTEGRASI FIREBASE

Disusun oleh :

Faisal Dahrul Ananto (17220728)

Muhammad Hanif Zidane (17220883)

Rizky David Saputra (17220907)

Izza Rahmat Sanjaya (17220922)

FAKULTAS TEKNOLOGI DAN INFORMASI UNIVERSITAS BINA SARANA

INFORMATIKA 2024

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa, yang telah melimpahkan rahmat dan karunia-Nya sehingga makalah yang berjudul **"Pengembangan Aplikasi Calendar Event Berbasis Flutter dengan Integrasi Firebase"** ini dapat diselesaikan dengan baik. Makalah ini disusun sebagai salah satu tugas akhir mata kuliah Mobile Programming dengan tujuan untuk memberikan pemahaman yang mendalam mengenai pengembangan aplikasi berbasis Flutter dan Firebase, serta sebagai referensi bagi pengembang aplikasi yang membutuhkan panduan dalam mengintegrasikan teknologi tersebut.

Makalah ini menjelaskan proses pengembangan aplikasi Calendar Event mulai dari latar belakang, tinjauan pustaka, metodologi pengembangan, implementasi teknis, hingga hasil pengujian. Aplikasi ini dirancang untuk membantu pengguna mengelola jadwal dan acara dengan fitur-fitur modern yang mudah digunakan. Dengan memanfaatkan kemampuan Flutter untuk membangun antarmuka multiplatform dan Firebase sebagai solusi backend real-time, diharapkan aplikasi ini dapat menjadi solusi praktis bagi kebutuhan manajemen waktu yang efisien.

Proses penyusunan makalah ini melibatkan berbagai literatur, referensi, serta hasil implementasi dan pengujian aplikasi secara langsung. Kami juga berusaha menyajikan isi makalah ini secara sistematis, lengkap, dan mudah dipahami agar dapat menjadi referensi yang bermanfaat bagi mahasiswa, dosen, maupun pengembang aplikasi.

Kami menyadari bahwa makalah ini masih jauh dari sempurna. Oleh karena itu, kami mengharapkan kritik dan saran yang membangun dari para pembaca demi perbaikan makalah ini di masa mendatang. Terima kasih kami sampaikan kepada dosen pengampu mata kuliah Mobile Programming yang telah memberikan arahan dan bimbingan selama penyusunan makalah ini. Ucapan terima kasih juga kami sampaikan kepada keluarga, teman, serta pihak-pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung.

Semoga makalah ini dapat memberikan wawasan baru dan bermanfaat bagi pembaca, khususnya dalam bidang pengembangan aplikasi mobile.

Hormat kami,

Kelompok 2

ABSTRAK

Aplikasi kalender digital merupakan salah satu kebutuhan utama dalam pengelolaan waktu di era modern. Banyak individu dan organisasi menggunakan kalender digital untuk mencatat, mengelola, dan mengingat jadwal harian maupun acara penting. Namun, sebagian besar aplikasi kalender yang tersedia memiliki keterbatasan dalam fitur personalisasi dan integrasi data real-time. Oleh karena itu, penelitian ini dilakukan untuk mengembangkan aplikasi Calendar Event berbasis Flutter dengan integrasi Firebase sebagai solusi yang fungsional, modern, dan efisien.

Flutter dipilih sebagai framework pengembangan karena kemampuannya dalam menghasilkan aplikasi multiplatform dengan antarmuka pengguna (UI) yang responsif dan performa tinggi. Firebase digunakan sebagai platform backend karena keunggulannya dalam menyediakan layanan seperti Firestore untuk penyimpanan data real-time, Authentication untuk autentikasi pengguna, dan integrasi yang mudah dengan Flutter.

Metodologi pengembangan aplikasi ini menggunakan model Agile, yang memungkinkan iterasi cepat dan fleksibilitas dalam menambahkan atau menyempurnakan fitur. Proses pengembangan mencakup desain antarmuka yang intuitif, implementasi fitur CRUD (Create, Read, Update, Delete), serta pengujian fungsionalitas dan responsivitas aplikasi pada berbagai perangkat. Aplikasi ini memungkinkan pengguna untuk menambahkan, membaca, mengedit, dan menghapus acara, serta menyinkronkan data secara real-time melalui Firebase Firestore

Hasil pengujian menunjukkan bahwa aplikasi ini dapat berjalan dengan baik pada berbagai platform (Android dan iOS) dengan tampilan yang konsisten dan responsif. Fungsi CRUD yang diimplementasikan juga terbukti berfungsi tanpa kendala. Dari segi performa, aplikasi ini menunjukkan waktu respons yang cepat dalam mengelola data.

Kesimpulannya, aplikasi Calendar Event berbasis Flutter dengan integrasi Firebase berhasil dikembangkan dan dapat memenuhi kebutuhan pengelolaan waktu dan jadwal secara modern. Penelitian ini memberikan kontribusi terhadap pengembangan aplikasi mobile, khususnya yang memanfaatkan teknologi Flutter dan Firebase. Pengembangan lebih lanjut dapat dilakukan dengan menambahkan fitur seperti pengingat berbasis notifikasi dan integrasi dengan kalender eksternal untuk meningkatkan fungsionalitas aplikasi.

Kata Kunci: **Flutter, Firebase, Calendar Event, CRUD, Agile, Aplikasi Mobile**

DAFTAR ISI

Pengembangan Aplikasi Calender Event Berbasis Flutter dengan Integrasi Firebase i
Kata Pengantar ii
Abstrak.....iii
Daftar Isi iv
BAB I.....5
Pendahuluan.....5
1.1 Latar Belakang.....5
1.2 Rumusan Masalah.....3
1.3 Tujuan Penelitian.....3
1.4 Manfaat Penelitian.....3
BAB II.....4
Tinjauan Pustaka.....4
2.1 Flutter.....4
2.2 Firebase.....4
2.3 Konsep CRUD.....5
BAB III.....7
Metodologi Pengembangan.....7
3.1 Model Pengembangan Sistem.....7
3.2 Arsitektur Sistem.....7
3.3 Teknologi yang Digunakan.....8
3.4 Fitur Utama Aplikasi.....8
BAB IV.....9
Implementasi dan Pengujian.....9
4.1 Implementasi Antarmuka Pengguna (UI).....9
4.1.1 Desain UI Aplikasi.....9
4.1.2 Struktur Widget Flutter yang Digunakan.....9
4.1.3 Kode Halaman Utama (List Task).....9
4.2 Integrasi Firebase.....13
4.2.1 Setup Firebase dengan Flutter.....13
4.2.2 Menambahkan Data ke Firebase.....14
4.2.3Membaca Data dari Firebase.....14
4.2.4 Menghapus Data dari Firebase.....14

4.2.5 Mengupdate Data dari Firebase 15
..... 15
4.3 Pengujian Aplikasi 15
4.3.1 Pengujian Fungsionalitas 15
4.3.2 Pengujian Responsivitas 15
BAB V 16
Penutup 16
5.1 Kesimpulan 16
5.2 Saran 16
Daftar Pustaka 17

BAB I

Pendahuluan

1.1 Latar Belakang

Manajemen waktu merupakan elemen krusial dalam kehidupan sehari-hari, baik bagi individu maupun organisasi. Dengan banyaknya aktivitas yang harus dilakukan, penggunaan alat bantu seperti kalender menjadi solusi praktis untuk mengelola jadwal dan acara. Dalam era digital saat ini, aplikasi kalender tidak hanya digunakan untuk mencatat jadwal, tetapi juga berfungsi sebagai pengingat otomatis dan alat kolaborasi dalam tim.

Namun, banyak aplikasi kalender digital yang tersedia di pasaran sering kali memiliki keterbatasan, seperti kurangnya kemampuan personalisasi, tidak mendukung sinkronisasi data secara real-time, atau sulit digunakan pada berbagai platform. Hal ini mendorong kebutuhan akan aplikasi kalender yang tidak hanya menawarkan fungsionalitas dasar, tetapi juga memberikan kemudahan akses, antarmuka yang responsif, dan kemampuan integrasi data secara real-time.

Flutter, sebagai framework pengembangan aplikasi multiplatform, menawarkan solusi untuk menciptakan aplikasi dengan antarmuka yang menarik, responsif, dan kompatibel di berbagai perangkat. Firebase, sebagai platform backend, menyediakan fitur seperti penyimpanan data real-time, autentikasi pengguna, dan analitik, yang dapat mendukung pengelolaan data secara efisien. Kombinasi Flutter dan Firebase menghadirkan potensi besar dalam membangun aplikasi modern yang mampu memenuhi kebutuhan pengguna dengan lebih baik.

Oleh karena itu, penelitian ini dilakukan untuk mengembangkan aplikasi Calendar Event berbasis Flutter dengan integrasi Firebase, yang dirancang untuk membantu pengguna dalam mengelola jadwal dan acara secara lebih efektif dan efisien. Aplikasi ini diharapkan mampu menyediakan fitur utama seperti pencatatan acara, pengelolaan data dengan operasi CRUD (Create, Read, Update, Delete), serta sinkronisasi data secara real-time dengan backend Firebase.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, beberapa permasalahan yang dapat dirumuskan adalah:

1. Bagaimana merancang antarmuka aplikasi kalender berbasis Flutter yang responsif, intuitif, dan user-friendly?
2. Bagaimana mengimplementasikan fitur CRUD untuk pengelolaan acara pada aplikasi kalender berbasis Flutter?
3. Bagaimana mengintegrasikan Firebase sebagai backend untuk mendukung sinkronisasi data secara real-time?
4. Bagaimana memastikan aplikasi dapat berjalan dengan optimal di berbagai perangkat (multiplatform)?

1.3 Tujuan Penelitian

Penelitian ini bertujuan untuk:

1. Mengembangkan aplikasi Calendar Event berbasis Flutter yang mendukung operasi CRUD untuk pengelolaan jadwal dan acara.
2. Mengintegrasikan Firebase sebagai backend untuk penyimpanan dan sinkronisasi data secara real-time.
3. Merancang antarmuka pengguna (UI) yang responsif, modern, dan mudah digunakan.
4. Menguji fungsionalitas dan performa aplikasi untuk memastikan kompatibilitasnya pada berbagai perangkat.

1.4 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah:

1. Bagi Pengguna:
    - Menyediakan solusi digital yang efisien untuk mengelola jadwal dan acara.
    - Membantu pengguna dalam meningkatkan produktivitas dengan pengelolaan waktu yang lebih baik.
2. Bagi Pengembang:
    - Memberikan referensi teknis dalam pengembangan aplikasi berbasis Flutter dengan integrasi Firebase.
    - Menunjukkan penerapan konsep CRUD dan sinkronisasi data real-time dalam aplikasi mobile.
3. Bagi Akademisi:
    - Memberikan kontribusi dalam penelitian dan pengembangan teknologi berbasis Flutter dan Firebase.
    - Menambah wawasan dalam bidang pengembangan aplikasi mobile yang responsif dan fungsional.
4. Bagi Industri:

- Menawarkan potensi implementasi aplikasi kalender sebagai alat bantu produktivitas di sektor bisnis atau organisasi.

Dengan manfaat-manfaat tersebut, penelitian ini diharapkan dapat memberikan dampak positif, baik secara praktis maupun teoritis, dalam pengembangan aplikasi mobile modern.

BAB II

Tinjauan Pustaka

2.1 Flutter

Flutter adalah sebuah framework open-source yang dikembangkan oleh Google untuk membangun antarmuka pengguna (UI) aplikasi multiplatform menggunakan satu basis kode. Dengan Flutter, pengembang dapat membuat aplikasi untuk platform Android, iOS, web, dan desktop secara bersamaan tanpa harus membuat kode terpisah untuk masing-masing platform.

Keunggulan Flutter:

1. Hot Reload: Fitur ini memungkinkan pengembang melihat perubahan kode secara instan tanpa harus menjalankan ulang aplikasi.
2. Kompatibilitas Multiplatform: Dengan satu kode dasar, aplikasi dapat dijalankan di berbagai platform seperti Android, iOS, Windows, macOS, dan Linux.
3. Widget yang Kaya: Flutter memiliki pustaka widget yang luas, memungkinkan pengembangan antarmuka pengguna yang menarik dan konsisten.
4. Performansi Tinggi: Flutter menggunakan mesin rendering sendiri (Skia), sehingga UI aplikasi terlihat sama di semua perangkat dan platform.
5. Dukungan Komunitas: Flutter memiliki komunitas pengembang yang besar dan aktif, menyediakan berbagai paket dan pustaka yang mempermudah pengembangan aplikasi.

Arsitektur Flutter:

Flutter menggunakan arsitektur berbasis widget, di mana setiap elemen dalam aplikasi, baik teks, tombol, maupun layout, merupakan widget. Ada dua jenis utama widget:

1. Stateless Widget: Widget yang tidak berubah selama siklus hidupnya.
2. Stateful Widget: Widget yang dapat berubah seiring waktu, tergantung pada interaksi pengguna atau data aplikasi.

Dengan pendekatan berbasis widget ini, Flutter memungkinkan pengembangan antarmuka pengguna yang fleksibel, modular, dan mudah dikelola.

2.2 Firebase

Firebase adalah platform pengembangan aplikasi yang awalnya dikembangkan oleh Firebase, Inc. pada tahun 2011 dan diakuisisi oleh Google pada tahun 2014. Firebase menyediakan berbagai layanan backend yang memungkinkan pengembang untuk fokus pada pengembangan aplikasi tanpa perlu mengelola infrastruktur server.

Fitur Utama Firebase yang Digunakan dalam Penelitian Ini:

1. Cloud Firestore: Layanan basis data NoSQL yang mendukung penyimpanan data secara real-time dan sinkronisasi di berbagai perangkat. Firestore memungkinkan pengelolaan data terstruktur dan tidak terstruktur dengan skema yang fleksibel.
2. Firebase Authentication: Fitur autentikasi yang mendukung login menggunakan email, Google, Facebook, dan metode lainnya.
3. Cloud Storage: Penyimpanan untuk file seperti gambar, video, atau dokumen.
4. Firebase Hosting: Layanan hosting untuk aplikasi web atau API backend.
5. Firebase Analytics: Alat analitik untuk melacak perilaku pengguna dalam aplikasi.

Keunggulan Firebase:

1. Integrasi Real-Time: Firebase memungkinkan sinkronisasi data antar perangkat secara real-time.
2. Kemudahan Penggunaan: Firebase menawarkan SDK yang mudah digunakan untuk berbagai bahasa dan framework, termasuk Flutter.
3. Dukungan untuk Aplikasi Skala Kecil hingga Besar: Firebase dirancang untuk mendukung aplikasi mulai dari proyek kecil hingga aplikasi dengan skala besar.

Firebase menjadi pilihan ideal untuk penelitian ini karena kemampuannya dalam mendukung pengelolaan data real-time, skalabilitas, dan integrasi yang mudah dengan Flutter.

2.3 Konsep CRUD

CRUD (Create, Read, Update, Delete) adalah konsep dasar dalam pengelolaan data yang digunakan dalam berbagai sistem informasi, termasuk aplikasi berbasis database. Operasi CRUD memungkinkan pengguna untuk mengelola data dengan cara:

- Create (C): Membuat atau menambahkan data baru ke dalam sistem atau basis data.
- Read (R): Membaca atau menampilkan data yang tersimpan di dalam basis data.
- Update (U): Memperbarui atau mengubah data yang sudah ada di dalam sistem.
- Delete (D): Menghapus data dari sistem atau basis data.

Implementasi CRUD dalam Penelitian Ini:

Dalam pengembangan aplikasi Calendar Event berbasis Flutter dengan integrasi Firebase, CRUD diterapkan pada pengelolaan data acara.

- Create: Pengguna dapat menambahkan acara 5ar uke kalender, yang disimpan ke dalam Firestore.
- Read: Data acara ditampilkan dalam bentuk daftar pada halaman utama aplikasi.
- Update: Pengguna dapat memperbarui detail acara yang telah disimpan, seperti tanggal, waktu, atau deskripsi.
- Delete: Pengguna dapat menghapus acara tertentu dari basis data Firestore.

Keuntungan Menggunakan Konsep CRUD:

1. Mempermudah pengelolaan data secara terstruktur.
2. Mendukung integrasi dengan berbagai platform database, termasuk Firebase Firestore.
3. Menyediakan alur kerja yang jelas untuk pengembangan aplikasi berbasis data.

Dengan penerapan konsep CRUD, aplikasi Calendar Event dapat memberikan pengalaman pengguna yang fungsional, intuitif, dan efisien dalam mengelola jadwal dan acara.

BAB III

Metodologi Pengembangan

3.1 Model Pengembangan Sistem

Model pengembangan sistem yang digunakan dalam penelitian ini adalah Agile Development, yang menawarkan fleksibilitas dalam menyesuaikan perubahan kebutuhan selama proses pengembangan. Agile memiliki pendekatan iteratif dan inkremental, yang memungkinkan evaluasi berkelanjutan berdasarkan umpan balik dari pengguna.

Tahapan dalam Agile Development:

1. Planning (Perencanaan):
   - Mengidentifikasi kebutuhan aplikasi.
   - Merancang arsitektur sistem dan menentukan teknologi yang akan digunakan.
2. Design (Desain):
   - Membuat wireframe dan mockup untuk antarmuka pengguna.
   - Menyusun struktur database Firebase Firestore.
3. Development (Pengembangan):
   - Mengimplementasikan fitur utama, termasuk CRUD dan autentikasi.
   - Mengintegrasikan Flutter dengan Firebase.
4. Testing (Pengujian):
   - Menguji fungsionalitas aplikasi (CRUD, autentikasi, sinkronisasi data).
   - Melakukan pengujian tampilan (responsivitas) pada berbagai perangkat.
5. Release (Rilis):
   - Menyelesaikan versi final aplikasi untuk didistribusikan.
6. Feedback (Umpan Balik):
   - Mengumpulkan masukan pengguna untuk pengembangan lebih lanjut.

Pendekatan Agile ini memungkinkan pengembangan aplikasi dilakukan secara fleksibel dan iteratif, menghasilkan aplikasi yang lebih sesuai dengan kebutuhan pengguna.

3.2 Arsitektur Sistem

Aplikasi Calendar Event menggunakan arsitektur berbasis client-server dengan Flutter sebagai client-side dan Firebase sebagai server-side.

Komponen Utama Arsitektur:

1. Client-Side (Frontend):
   - Dikembangkan menggunakan Flutter.
   - Bertanggung jawab untuk mengelola antarmuka pengguna dan interaksi pengguna.
2. Server-Side (Backend):
   - Menggunakan Firebase Firestore untuk penyimpanan data acara.
   - Firebase Authentication untuk autentikasi pengguna.
   - Firebase Cloud Messaging untuk pengiriman notifikasi (opsional).

3.3 Teknologi yang Digunakan

Pengembangan aplikasi ini memanfaatkan teknologi berikut:

1. Flutter:
   - Framework pengembangan aplikasi multiplatform.
   - Menawarkan fitur seperti hot reload dan pustaka widget lengkap.
2. Firebase:
   - Firestore: Basis data NoSQL untuk penyimpanan data real-time.
   - Authentication: Mendukung login dan registrasi pengguna.
   - Cloud Messaging: Mendukung notifikasi acara (opsional).
3. Dart:
   - Bahasa pemrograman yang digunakan untuk membangun aplikasi Flutter.
4. Figma (Opsional):
   - Alat desain prototipe antarmuka pengguna.
5. Visual Studio Code:
   - Editor kode utama untuk pengembangan aplikasi.

Teknologi ini dipilih karena kemudahan penggunaan, fleksibilitas, dan integrasinya dalam mendukung aplikasi berbasis data real-time.

3.4 Fitur Utama Aplikasi

Fitur utama aplikasi Calendar Event dirancang untuk memudahkan pengguna dalam mengelola jadwal dan acara mereka, meliputi:

1. Pencatatan Acara:
   - Menambahkan acara baru dengan informasi seperti nama, deskripsi, tanggal, dan waktu.
2. Operasi CRUD:
   - Create: Menambahkan acara baru.
   - Read: Menampilkan daftar acara dalam bentuk list atau kalender.
   - Update: Memperbarui informasi acara.
   - Delete: Menghapus acara yang tidak diperlukan.
3. Sinkronisasi Data Real-Time:
   - Data disimpan di Firebase Firestore dan dapat diakses secara real-time di berbagai perangkat.
4. Autentikasi Pengguna:
   - Mendukung login dan registrasi menggunakan email dan password.
5. Antarmuka Pengguna yang Responsif:
   - Desain modern yang mendukung berbagai ukuran layar dan perangkat.
6. Pencarian dan Filter Acara:
   - Fitur untuk mencari acara berdasarkan kata kunci atau filter berdasarkan tanggal tertentu.
7. Pengingat Acara (Opsional):
   - Notifikasi otomatis untuk mengingatkan pengguna tentang acara yang akan datang.

Fitur-fitur ini dirancang untuk memberikan pengalaman pengguna yang efisien, intuitif, dan sesuai dengan kebutuhan pengelolaan jadwal.

BAB IV

Implementasi dan Pengujian

4.1 Implementasi Antarmuka Pengguna (UI)

Antarmuka pengguna (UI) aplikasi Calendar Event dikembangkan menggunakan Flutter dengan pendekatan berbasis widget. Desain UI dirancang agar sederhana, intuitif, dan mudah digunakan oleh pengguna dari berbagai kalangan.

4.1.1 Desain UI Aplikasi

Desain antarmuka pengguna (UI) aplikasi dirancang dengan prinsip sederhana dan intuitif agar mudah digunakan. Tampilan utama mencakup:

1. Halaman Utama (List Task): Menampilkan daftar acara dalam format list atau kalender.
2. Halaman Tambah Tugas (Add Task): Formulir untuk menambahkan acara baru dengan input seperti nama, deskripsi, tanggal, dan waktu.

Proses perancangan dilakukan menggunakan Figma untuk menghasilkan prototipe antarmuka yang mendukung berbagai ukuran layar perangkat. Desain memperhatikan:

- Konsistensi warna dan font.
- Tata letak responsif (responsiveness).
- Navigasi sederhana antar halaman.

4.1.2 Struktur Widget Flutter yang Digunakan

Struktur widget Flutter didesain berbasis Material Design, dengan hierarki berikut:

1. Halaman Utama:

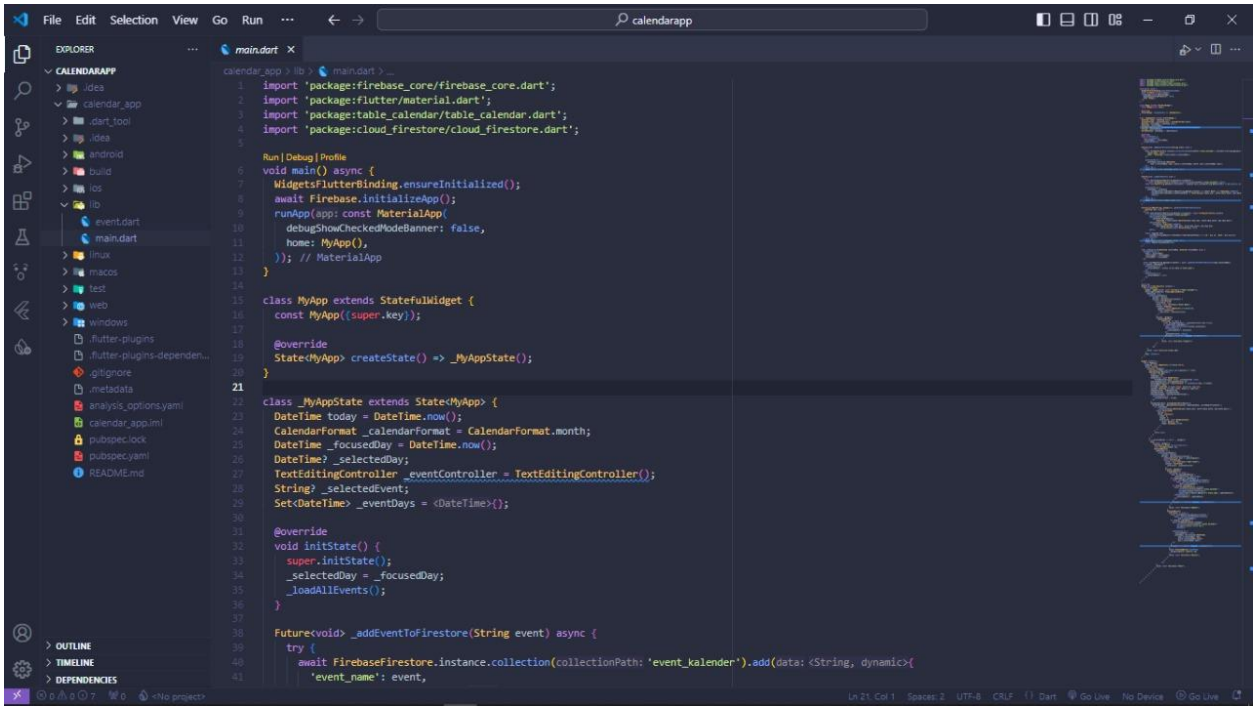
- Scaffold: Kerangka utama untuk menampung elemen UI.
- AppBar: Menampilkan judul halaman dan menu tambahan.
- Body: Berisi widget seperti ListView untuk daftar acara atau kalender menggunakan plugin TableCalendar.

2. Halaman Tambah Tugas:

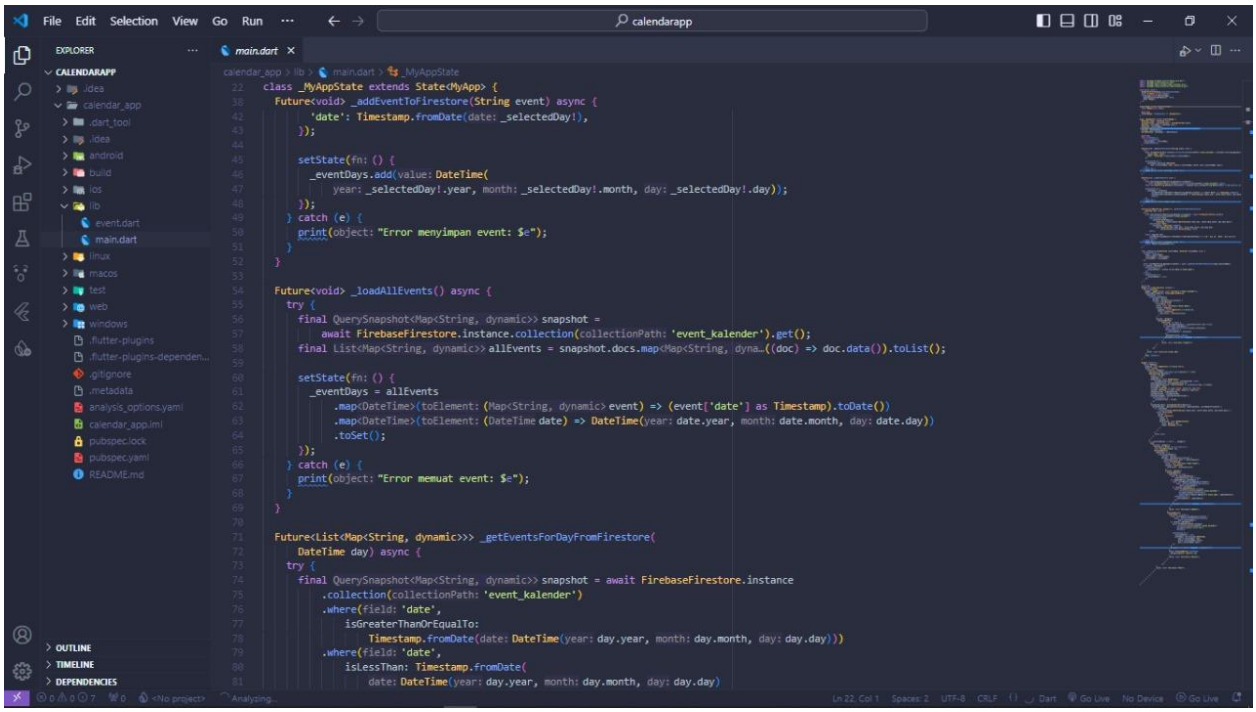
- Form: Menggunakan widget TextFormField untuk input data.
- DatePicker: Untuk memilih tanggal acara.
- TimePicker: Untuk memilih waktu acara.
- Button: Tombol untuk menyimpan data ke Firebase.

4.1.3 Kode Halaman Utama (List Task)

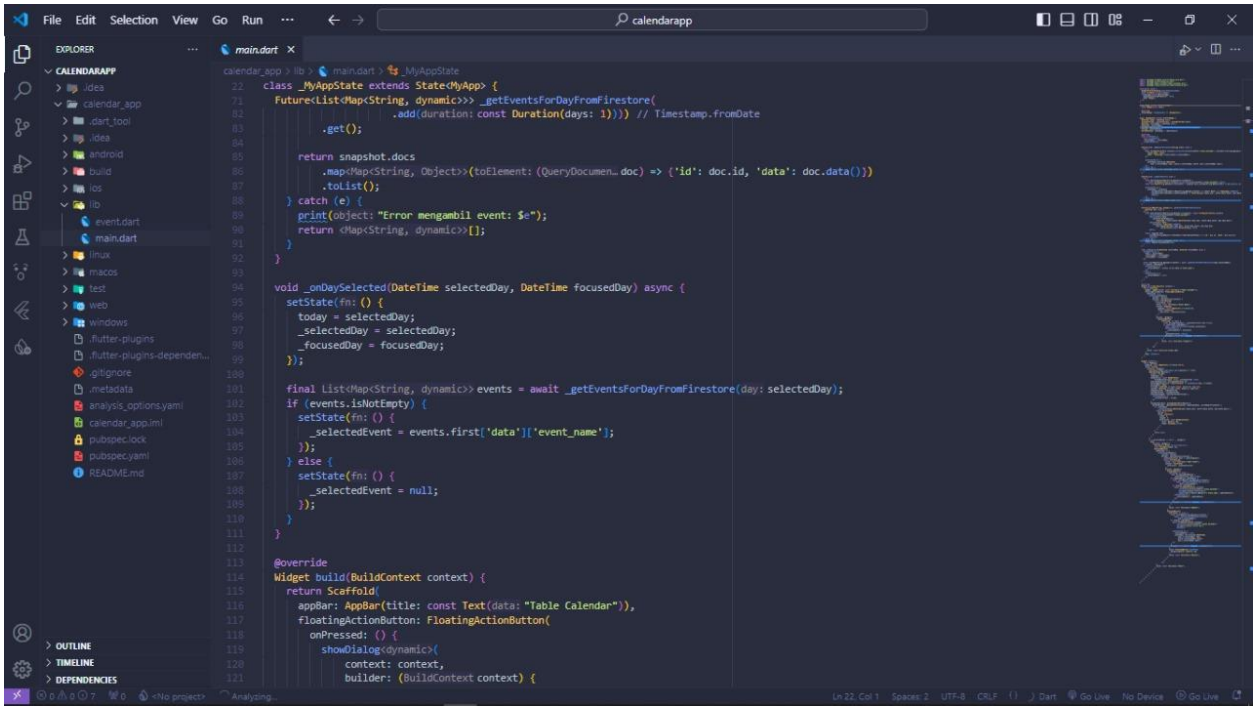
Berikut adalah implementasi halaman untuk menambahkan acara baru:



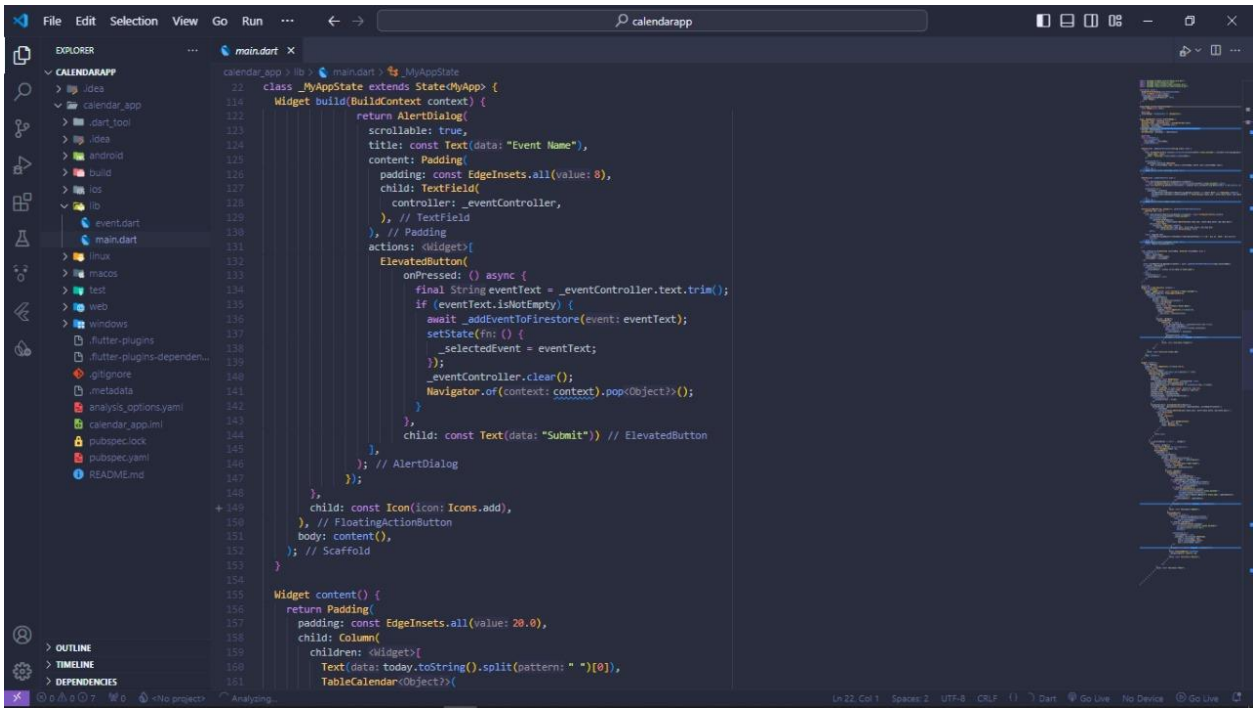
```dart
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:table_calendar/table_calendar.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

Run | Debug | Profile
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(app: const MaterialApp(
    debugShowCheckedModeBanner: false,
    home: MyApp(),
  )); // MaterialApp
}

class MyApp extends StatefulWidget {
  const MyApp({super.key});

  @override
  State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  DateTime today = DateTime.now();
  CalendarFormat _calendarFormat = CalendarFormat.month;
  DateTime _focusedDay = DateTime.now();
  DateTime? _selectedDay;
  TextEditingController _eventController = TextEditingController();
  String? _selectedEvent;
  Set<DateTime> _eventDays = <DateTime>{};

  @override
  void initState() {
    super.initState();
    _selectedDay = _focusedDay;
    _loadAllEvents();
  }

  Future<void> _addEventToFirestore(String event) async {
    try {
      await FirebaseFirestore.instance.collection(collectionPath: 'event_kalender').add(data: <String, dynamic>{
        'event_name': event,
```



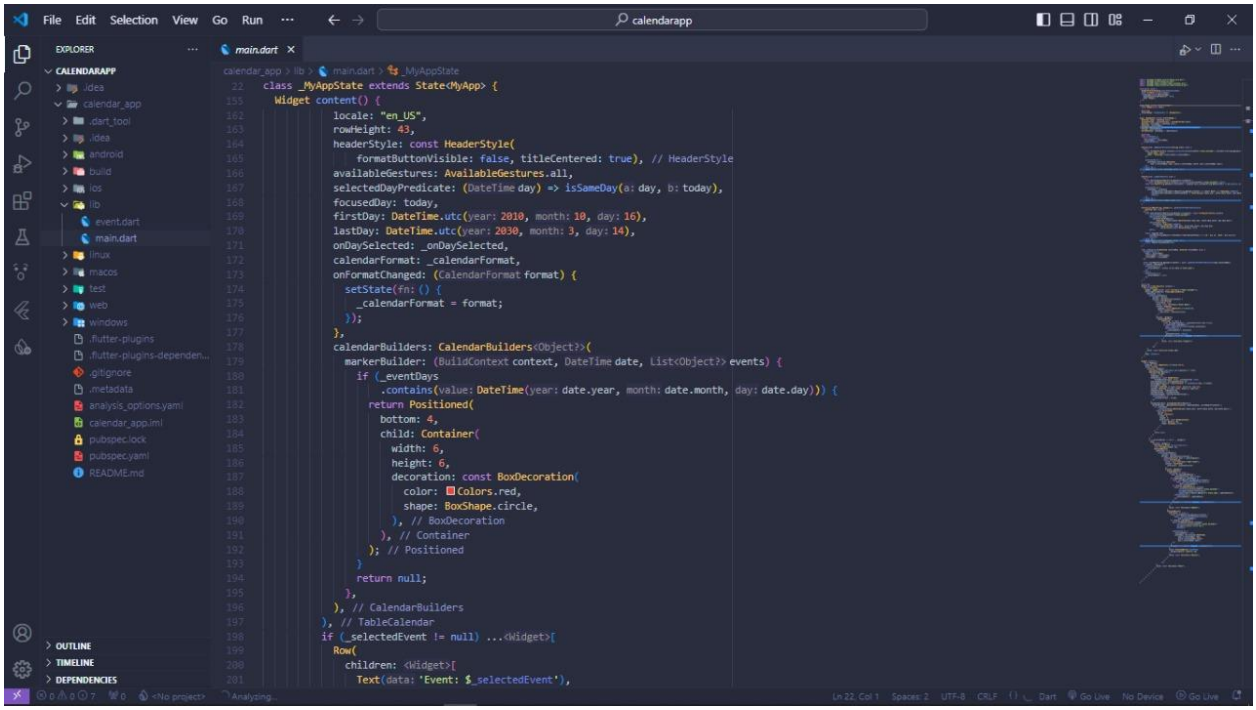
```dart
class _MyAppState extends State<MyApp> {
  Future<void> _addEventToFirestore(String event) async {
        'date': Timestamp.fromDate(date: _selectedDay!),
      });

      setState(fn: () {
        _eventDays.add(value: DateTime(
            year: _selectedDay!.year, month: _selectedDay!.month, day: _selectedDay!.day));
      });
    } catch (e) {
      print(object: "Error menyimpan event: $e");
    }
  }

  Future<void> _loadAllEvents() async {
    try {
      final QuerySnapshot<Map<String, dynamic>> snapshot =
          await FirebaseFirestore.instance.collection(collectionPath: 'event_kalender').get();
      final List<Map<String, dynamic>> allEvents = snapshot.docs.map<Map<String, dyna…((doc) => doc.data()).toList();

      setState(fn: () {
        _eventDays = allEvents
            .map<DateTime>(toElement: (Map<String, dynamic> event) => (event['date'] as Timestamp).toDate())
            .map<DateTime>(toElement: (DateTime date) => DateTime(year: date.year, month: date.month, day: date.day))
            .toSet();
      });
    } catch (e) {
      print(object: "Error memuat event: $e");
    }
  }

  Future<List<Map<String, dynamic>>> _getEventsForDayFromFirestore(
      DateTime day) async {
    try {
      final QuerySnapshot<Map<String, dynamic>> snapshot = await FirebaseFirestore.instance
          .collection(collectionPath: 'event_kalender')
          .where(field: 'date',
              isGreaterThanOrEqualTo:
                  Timestamp.fromDate(date: DateTime(year: day.year, month: day.month, day: day.day)))
          .where(field: 'date',
              isLessThan: Timestamp.fromDate(
                  date: DateTime(year: day.year, month: day.month, day: day.day)
```



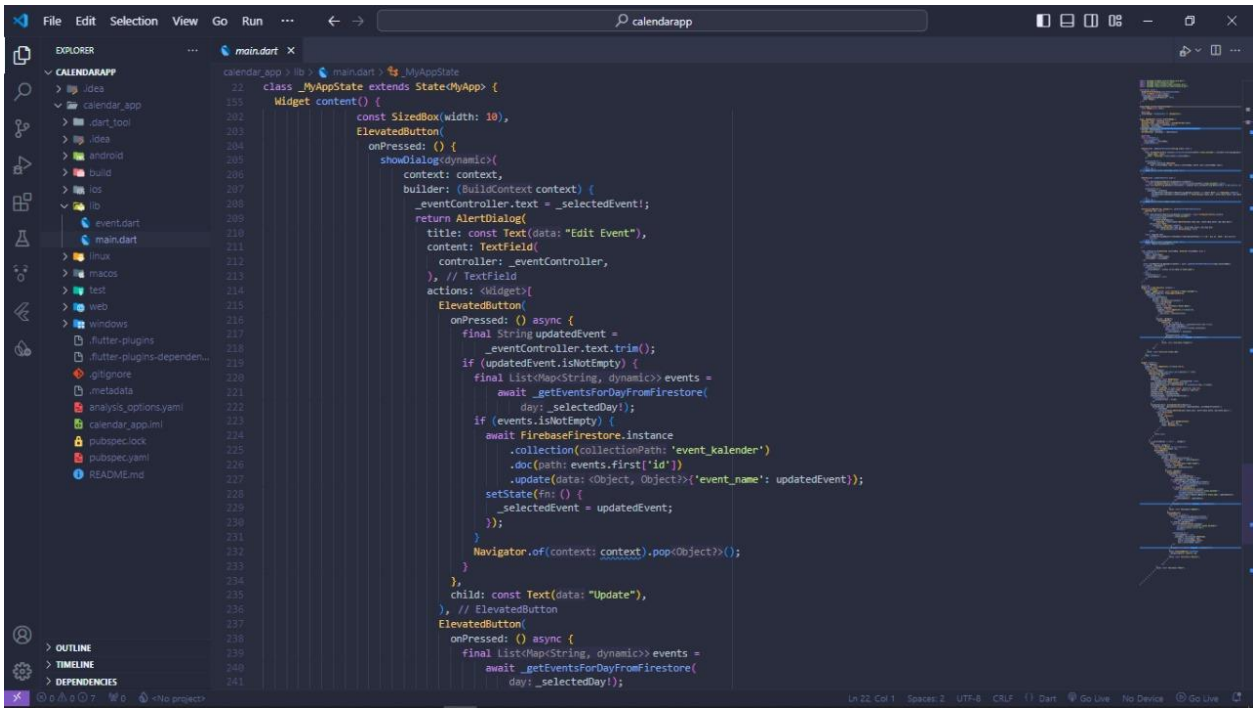
```
class _MyAppState extends State<MyApp> {
  Future<List<Map<String, dynamic>>> _getEventsForDayFromFirestore(
                    .add(duration: const Duration(days: 1)))) // Timestamp.fromDate
            .get();

        return snapshot.docs
            .map<Map<String, Object>>(toElement: (QueryDocumen… doc) => {'id': doc.id, 'data': doc.data()})
            .toList();
      } catch (e) {
        print(object: "Error mengambil event: $e");
        return <Map<String, dynamic>>[];
      }
    }

    void _onDaySelected(DateTime selectedDay, DateTime focusedDay) async {
      setState(fn: () {
        today = selectedDay;
        _selectedDay = selectedDay;
        _focusedDay = focusedDay;
      });

      final List<Map<String, dynamic>> events = await _getEventsForDayFromFirestore(day: selectedDay);
      if (events.isNotEmpty) {
        setState(fn: () {
          _selectedEvent = events.first['data']['event_name'];
        });
      } else {
        setState(fn: () {
          _selectedEvent = null;
        });
      }
    }

    @override
    Widget build(BuildContext context) {
      return Scaffold(
        appBar: AppBar(title: const Text(data: "Table Calendar")),
        floatingActionButton: FloatingActionButton(
          onPressed: () {
            showDialog<dynamic>(
                context: context,
                builder: (BuildContext context) {
```



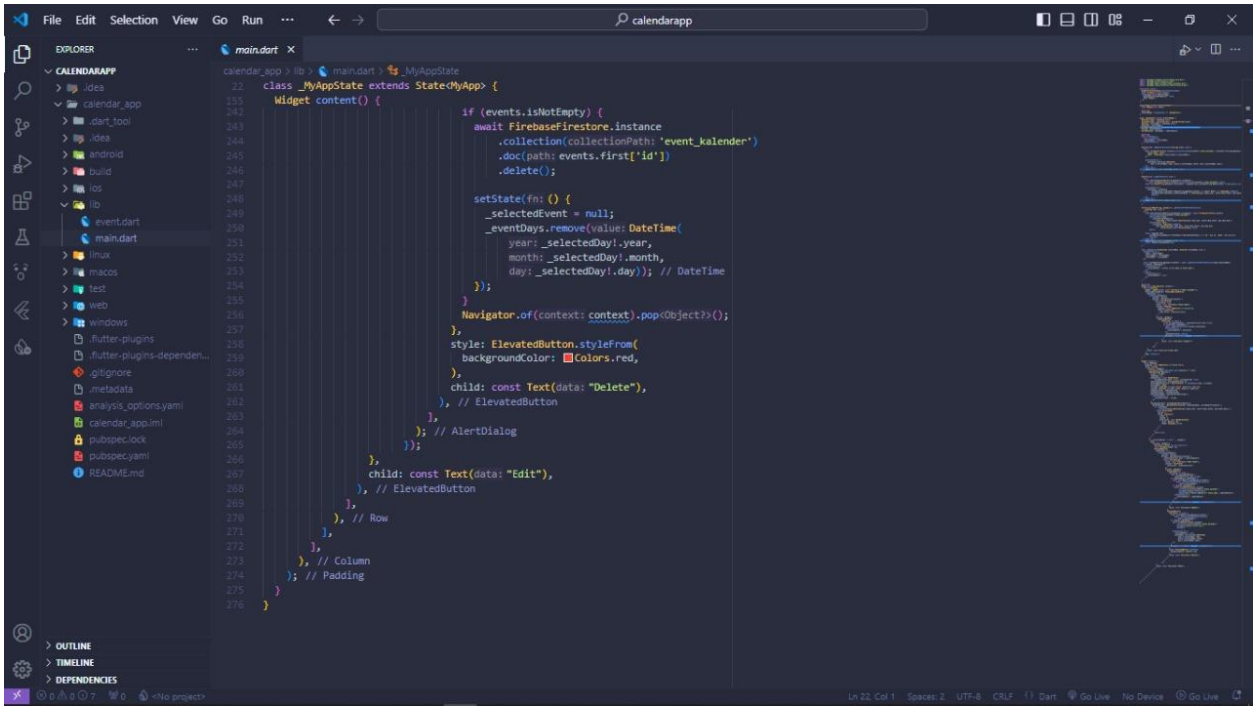
```
class _MyAppState extends State<MyApp> {
  Widget build(BuildContext context) {
              return AlertDialog(
                scrollable: true,
                title: const Text(data: "Event Name"),
                content: Padding(
                  padding: const EdgeInsets.all(value: 8),
                  child: TextField(
                    controller: _eventController,
                  ), // TextField
                ), // Padding
                actions: <Widget>[
                  ElevatedButton(
                      onPressed: () async {
                        final String eventText = _eventController.text.trim();
                        if (eventText.isNotEmpty) {
                          await _addEventToFirestore(event: eventText);
                          setState(fn: () {
                            _selectedEvent = eventText;
                          });
                          _eventController.clear();
                          Navigator.of(context: context).pop<Object?>();
                        }
                      },
                      child: const Text(data: "Submit")) // ElevatedButton
                ],
              ); // AlertDialog
            });
        },
        child: const Icon(icon: Icons.add),
      ), // FloatingActionButton
      body: content(),
    ); // Scaffold
  }

  Widget content() {
    return Padding(
      padding: const EdgeInsets.all(value: 20.0),
      child: Column(
        children: <Widget>[
          Text(data: today.toString().split(pattern: " ")[0]),
          TableCalendar<Object?>(
```



```
class _MyAppState extends State<MyApp> {
  Widget content() {
            locale: "en_US",
            rowHeight: 43,
            headerStyle: const HeaderStyle(
                formatButtonVisible: false, titleCentered: true), // HeaderStyle
            availableGestures: AvailableGestures.all,
            selectedDayPredicate: (DateTime day) => isSameDay(a: day, b: today),
            focusedDay: today,
            firstDay: DateTime.utc(year: 2010, month: 10, day: 16),
            lastDay: DateTime.utc(year: 2030, month: 3, day: 14),
            onDaySelected: _onDaySelected,
            calendarFormat: _calendarFormat,
            onFormatChanged: (CalendarFormat format) {
              setState(fn: () {
                _calendarFormat = format;
              });
            },
            calendarBuilders: CalendarBuilders<Object?>(
              markerBuilder: (BuildContext context, DateTime date, List<Object?> events) {
                if (_eventDays
                    .contains(value: DateTime(year: date.year, month: date.month, day: date.day))) {
                  return Positioned(
                    bottom: 4,
                    child: Container(
                      width: 6,
                      height: 6,
                      decoration: const BoxDecoration(
                        color: Colors.red,
                        shape: BoxShape.circle,
                      ), // BoxDecoration
                    ), // Container
                  ); // Positioned
                }
                return null;
              },
            ), // CalendarBuilders
          ), // TableCalendar
          if (_selectedEvent != null) ...<Widget>[
            Row(
              children: <Widget>[
                Text(data: 'Event: $_selectedEvent'),
```



```
class _MyAppState extends State<MyApp> {
  Widget content() {
            const SizedBox(width: 10),
            ElevatedButton(
              onPressed: () {
                showDialog<dynamic>(
                    context: context,
                    builder: (BuildContext context) {
                      _eventController.text = _selectedEvent!;
                      return AlertDialog(
                        title: const Text(data: "Edit Event"),
                        content: TextField(
                          controller: _eventController,
                        ), // TextField
                        actions: <Widget>[
                          ElevatedButton(
                            onPressed: () async {
                              final String updatedEvent =
                                  _eventController.text.trim();
                              if (updatedEvent.isNotEmpty) {
                                final List<Map<String, dynamic>> events =
                                    await _getEventsForDayFromFirestore(
                                        day: _selectedDay!);
                                if (events.isNotEmpty) {
                                  await FirebaseFirestore.instance
                                      .collection(collectionPath: 'event_kalender')
                                      .doc(path: events.first['id'])
                                      .update(data: <Object, Object?>{'event_name': updatedEvent});
                                  setState(fn: () {
                                    _selectedEvent = updatedEvent;
                                  });
                                }
                                Navigator.of(context: context).pop<Object?>();
                              }
                            },
                            child: const Text(data: "Update"),
                          ), // ElevatedButton
                          ElevatedButton(
                            onPressed: () async {
                              final List<Map<String, dynamic>> events =
                                  await _getEventsForDayFromFirestore(
                                      day: _selectedDay!);
```



4.2 Integrasi Firebase

4.2.1 Setup Firebase dengan Flutter

Langkah-langkah integrasi Firebase:

1. Tambahkan Firebase ke Proyek Flutter:

   - Buat proyek Firebase di Firebase Console.
   - Unduh file google-services.json dan tambahkan ke direktori proyek android/app/

2. Tambahkan Plugin Firebase:

   - Tambahkan dependensi berikut ke *pubspec.yaml:*

3. Inisialisasi Firebase:

   - Inisialisasi Firebase di file utama aplikasi (*main.dart*):

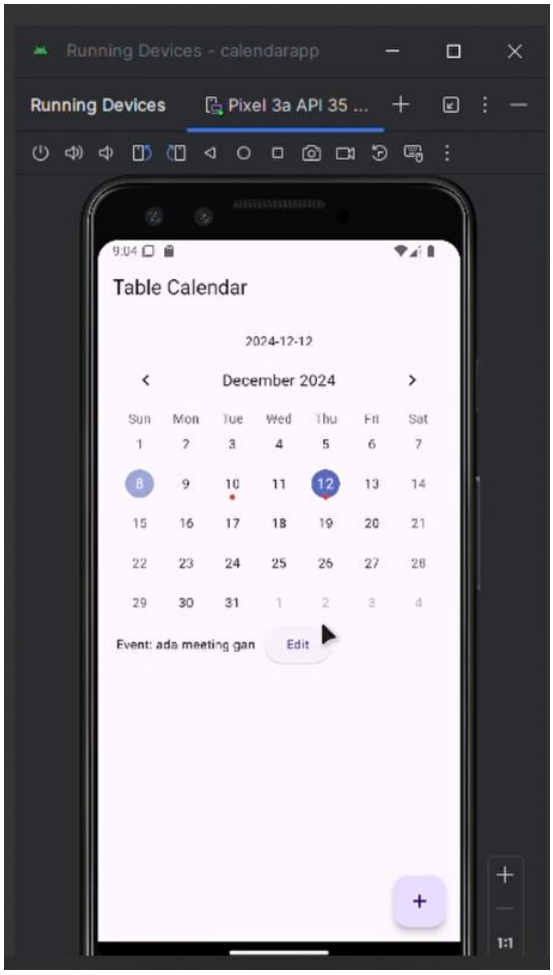
4.2.2 Menambahkan Data ke Firebase

Data ditambahkan menggunakan metode add() pada koleksi:

```dart
Future<void> _addEventToFirestore(String event) async {
  try {
    await FirebaseFirestore.instance.collection('event_kalender').add({
      'event_name': event,
      'date': Timestamp.fromDate(_selectedDay!),
    });

    setState(() {
      _eventDays.add(DateTime(
          _selectedDay!.year, _selectedDay!.month, _selectedDay!.day));
    });
  } catch (e) {
    print("Error menyimpan event: $e");
  }
}
```

4.2.3 Membaca Data dari Firebase

Data diambil dengan StreamBuilder untuk sinkronisasi real-time:



4.2.4 Menghapus Data dari Firebase

Data dihapus dengan metode delete():

```
if (events.isNotEmpty) {
  await FirebaseFirestore.instance
      .collection(collectionPath: 'event_kalender')
      .doc(path: events.first['id'])
      .delete();
```

4.2.5 Mengupdate Data dari Firebase

```
        day: _selectedDay!);
if (events.isNotEmpty) {
  await FirebaseFirestore.instance
      .collection(collectionPath: 'event_kalender')
      .doc(path: events.first['id'])
      .update(data: <Object, Object?>{'event_name': updatedEvent});
  setState(fn: () {
    _selectedEvent = updatedEvent;
  });
}
```

4.3 Pengujian Aplikasi

4.3.1 Pengujian Fungsionalitas

Pengujian dilakukan untuk memastikan operasi CRUD bekerja dengan baik:

- Create: Menambah acara baru berhasil menyimpan data ke Firestore.
- Read: Menampilkan data acara secara real-time di aplikasi.
- Update: Memperbarui informasi acara berhasil tersimpan di Firestore.
- Delete: Menghapus acara berhasil menghapus data di Firestore.

4.3.2 Pengujian Responsivitas

Aplikasi diuji pada berbagai ukuran layar, seperti:

- Smartphone dengan resolusi kecil, sedang, dan besar.
- Tablet.

Hasil pengujian menunjukkan aplikasi mampu menyesuaikan tampilan secara optimal pada berbagai perangkat.

BAB V

Penutup

5.1 Kesimpulan

Berdasarkan hasil pengembangan dan pengujian aplikasi Calendar Event berbasis Flutter dengan integrasi Firebase, dapat disimpulkan bahwa aplikasi ini berhasil memenuhi tujuan yang ditetapkan. Aplikasi ini memungkinkan pengguna untuk mengelola acara secara efektif dan efisien dengan fitur utama seperti menambah, melihat, mengedit, dan menghapus acara. Integrasi Firebase memastikan data acara disimpan secara real-time, memungkinkan pengguna untuk mengakses data dari berbagai perangkat.

Pengujian fungsionalitas dan responsivitas juga menunjukkan bahwa aplikasi bekerja dengan baik pada berbagai perangkat dengan ukuran layar yang berbeda, memastikan pengalaman pengguna yang optimal. Dengan menggunakan framework Flutter dan Firebase, aplikasi ini menawarkan kemudahan pengembangan dan pemeliharaan yang efisien.

Secara keseluruhan, aplikasi ini dapat digunakan sebagai alat bantu untuk pengelolaan acara secara pribadi atau dalam skala kecil dan menengah. Teknologi yang digunakan, yaitu Flutter dan Firebase, mendukung pengembangan aplikasi multiplatform yang responsif dan dapat diskalakan.

5.2 Saran

Meskipun aplikasi telah berfungsi dengan baik, masih terdapat beberapa area yang dapat ditingkatkan untuk meningkatkan pengalaman pengguna dan fungsionalitas aplikasi:

1. Peningkatan Fitur Notifikasi:
- Aplikasi dapat dikembangkan lebih lanjut dengan menambahkan pengingat otomatis atau notifikasi push untuk acara yang akan datang, agar pengguna lebih terorganisir dalam mengelola jadwal mereka.
2. Fitur Kolaborasi Acara:
- Menambahkan kemampuan bagi pengguna untuk mengundang orang lain ke acara mereka dan berbagi informasi acara dengan lebih mudah dapat meningkatkan kegunaan aplikasi, terutama untuk acara yang melibatkan banyak orang.
3. Penggunaan Firebase Cloud Functions:
- Penggunaan Firebase Cloud Functions untuk menangani logika backend secara otomatis dapat mempercepat pengolahan data dan meningkatkan performa aplikasi, seperti pengingat otomatis atau pengelolaan sinkronisasi data.
4. Peningkatan Keamanan Data:
- Menambahkan fitur keamanan tambahan, seperti autentikasi dua faktor, akan memberikan tingkat keamanan lebih tinggi untuk melindungi data pengguna dan acara mereka.
5. Pengujian Pengguna Lebih Lanjut:
- Melakukan uji coba dengan pengguna dari berbagai latar belakang untuk mendapatkan umpan balik lebih mendalam mengenai pengalaman penggunaan aplikasi dapat membantu memperbaiki antarmuka pengguna dan menambah fitur yang lebih dibutuhkan.

Aplikasi ini merupakan langkah awal yang baik, namun dengan pengembangan lebih lanjut, dapat menjadi alat yang sangat berguna untuk manajemen acara dan tugas sehari-hari.

Daftar Pustaka

1. **Cahyani, R. (2021).** Pengembangan Aplikasi Kalender Berbasis Android Menggunakan Firebase. *Jurnal Teknologi dan Sistem Informasi*, 8(2), 134-145.
2. **Google. (2024).** Flutter Documentation. Diakses dari: https://flutter.dev/docs
3. **Google. (2024).** Firebase Documentation. Diakses dari: https://firebase.google.com/docs
4. **Moulay, R., & Rizki, I. (2022).** Implementasi CRUD pada Aplikasi Pengelolaan Data dengan *Firebase dan Flutter. Jurnal Informatika dan Teknologi*, 6(1), 98-104.
5. **Rahman, F. (2023).** Flutter untuk Pengembangan Aplikasi Mobile: Panduan Lengkap. Yogyakarta: *Penerbit Andalan.*
6. **Sari, D. (2022)**. Pemrograman Mobile dengan Flutter dan Firebase. *Bandung*: Penerbit Informatika.
7. **Yunita, M. (2023).** Penerapan Firebase untuk Penyimpanan Data Real-Time pada Aplikasi Mobile. *Jurnal Sistem Informasi dan Teknologi*, 9(3), 220-227.