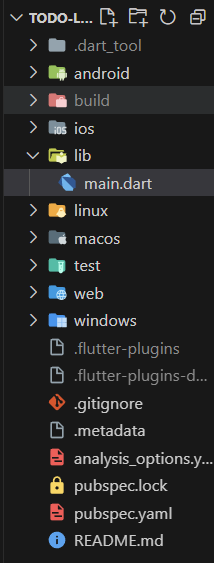
**Dokumentasi Aplikasi To Do List**

**Gambaran Umum**

Aplikasi Flutter ini adalah daftar tugas sederhana yang memungkinkan pengguna menambahkan, mengedit, menghapus, dan melihat detail dari setiap item tugas. Setiap item tugas memiliki tugas, status penyelesaian, dan tanggal. Pengguna juga dapat berbagi item tugas ke media sosial.

**Berkas Utama (main.dart)**



`main.dart` merupakan berkas utama aplikasi yang berisi semua kode program. Dibagi menjadi beberapa bagian utama, berikut penjelasan fungsionalitas masing-masing bagian.

**Import Package**

import 'package:flutter/material.dart';

import 'package:share/share.dart';

Bagian ini mengimpor paket Flutter yang diperlukan, yaitu `flutter/material.dart` untuk komponen UI dan `share/share.dart` untuk berbagi konten.

**Main Function**

void main() {

  runApp(MyApp());

}

Fungsi `main` menjalankan aplikasi dengan memanggil fungsi `runApp` dan memberikan instance dari widget `MyApp` sebagai root widget.

**Kelas `MyApp`**

class MyApp extends StatelessWidget {

  const MyApp({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      debugShowCheckedModeBanner: false,

      home: TodoListScreen(),

    );

  }

}

Kelas ini mewakili aplikasi utama. Berfungsi sebagai root widget yang menjalankan widget `TodoListScreen` sebagai layar utama.

**Kelas `TodoListScreen`**

class TodoListScreen extends StatefulWidget {

  @override

  \_TodoListScreenState createState() => \_TodoListScreenState();

}

Kelas ini merupakan widget dengan keadaan yang mewakili layar utama aplikasi. Menampilkan daftar item tugas dan menyediakan fungsionalitas untuk menambah, mengedit, menghapus, dan melihat detail item tugas.

**Kelas `DetailPage`**

class DetailPage extends StatelessWidget {

  final TodoItem todoItem;

  final Function(int) onToggle;

  DetailPage({required this.todoItem, required this.onToggle});

Kelas ini merupakan widget tanpa keadaan yang menampilkan informasi detail tentang suatu item tugas. Pengguna dapat beralih status penyelesaian dan berbagi item tugas.

**Kelas `TodoItem`**

class TodoItem {

  String task;

  bool isCompleted;

  String date;

  int index;

  TodoItem(

      {required this.task,

      required this.isCompleted,

      required this.date,

      required this.index});

}

Kelas ini mewakili item tugas dengan tugas, status penyelesaian, tanggal, dan indeks.

**Metode Utama (`main()`)**

void main() {

  runApp(MyApp());

}

Metode utama menjalankan aplikasi dengan membuat instance dari widget `MyApp` dan menjalankannya menggunakan `runApp`.

**Struktur UI (`build` Method)**

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      debugShowCheckedModeBanner: false,

      home: TodoListScreen(),

    );

  }

Metode `build` pada kelas `MyApp` menetapkan struktur UI utama dengan menggunakan widget `MaterialApp`. Ini menonaktifkan banner debug dan menetapkan `TodoListScreen` sebagai layar utama.

**Kelas `TodoListScreen` (`build` Method)**

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Todo List'),

      ),

      body: ListView.builder(

        itemCount: todoItems.length,

        itemBuilder: (context, index) {

          return ListTile(

            title: Text(

              todoItems[index].task,

              style: TextStyle(

                decoration: todoItems[index].isCompleted

                    ? TextDecoration.lineThrough

                    : TextDecoration.none,

              ),

            ),

            trailing: Icon(Icons.arrow\_forward\_ios),

            onTap: () {

              \_showTodoDetail(todoItems[index]);

            },

            onLongPress: () {

              \_showOptionsDialog(index);

            },

          );

        },

      ),

      floatingActionButton: FloatingActionButton(

        onPressed: () {

          \_addTodoItem();

        },

        child: Icon(Icons.add),

      ),

    );

  }

Metode `build` pada kelas `TodoListScreen` menetapkan struktur UI untuk layar utama. Ini mencakup AppBar, daftar item tugas dalam ListView.builder, dan FloatingActionButton untuk menambahkan item tugas baru.

**Fungsi Menambahkan Tugas (`\_addTodoItem` Method)**

  void \_addTodoItem() {

    showDialog(

      context: context,

      builder: (BuildContext context) {

        TextEditingController controller = TextEditingController();

        return AlertDialog(

          title: Text('Add Todo Item'),

          content: TextField(

            controller: controller,

            onChanged: (text) {

              // Handle the input text

            },

          ),

          actions: [

            TextButton(

              onPressed: () {

                Navigator.of(context).pop();

              },

              child: Text('Cancel'),

            ),

            TextButton(

              onPressed: () {

                String task = controller.text;

                DateTime now = DateTime.now();

                String date = "${now.day}/${now.month}/${now.year}";

                int index = todoItems.length;

                setState(() {

                  todoItems.add(TodoItem(

                      task: task,

                      isCompleted: false,

                      date: date,

                      index: index));

                });

                Navigator.of(context).pop();

              },

              child: Text('Add'),

            ),

          ],

        );

      },

    );

  }

Metode ini menampilkan dialog untuk menambahkan tugas baru ketika pengguna menekan tombol tambah.

**Fungsi Menampilkan Pilihan (`\_showOptionsDialog` Method)**

  void \_showOptionsDialog(int index) {

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: Text('Options'),

          actions: [

            TextButton(

              onPressed: () {

                \_editTodoItem(index);

                Navigator.of(context).pop();

              },

              child: Text('Edit'),

            ),

            TextButton(

              onPressed: () {

                \_deleteTodoItem(index);

                Navigator.of(context).pop();

              },

              child: Text('Delete'),

            ),

          ],

        );

      },

    );

  }

Metode ini menampilkan dialog dengan pilihan edit atau hapus ketika pengguna melakukan tap dan tahan pada item tugas.

**Kelas `DetailPage` (`build`)**

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Todo Detail'),

        actions: [

          IconButton(

            icon: Icon(Icons.share),

            onPressed: () {

              \_shareTask(context);

            },

          ),

        ],

      ),

      body: Padding(

        padding: const EdgeInsets.all(16.0),

        child: Column(

          crossAxisAlignment: CrossAxisAlignment.start,

          children: [

            Row(

              mainAxisAlignment: MainAxisAlignment.spaceBetween,

              children: [

                Text(

                  'Task: ${todoItem.task}',

                  style: TextStyle(

                    decoration: todoItem.isCompleted

                        ? TextDecoration.lineThrough

                        : TextDecoration.none,

                  ),

                ),

                Checkbox(

                  value: todoItem.isCompleted,

                  onChanged: (value) {

                    onToggle(todoItem.index);

                    Navigator.pop(context, true);

                  },

                ),

              ],

            ),

            SizedBox(height: 8),

            Text(

              'Status: ${todoItem.isCompleted ? 'Completed' : 'Incomplete'}',

            ),

            SizedBox(height: 8),

            Text(

              'Date: ${todoItem.date}',

            ),

          ],

        ),

      ),

    );

  }

Metode `build` pada kelas `DetailPage` menetapkan struktur UI untuk halaman detail item tugas. Ini mencakup informasi seperti tugas, status penyelesaian, dan tanggal. Juga menambahkan ikon berbagi untuk berbagi item tugas.

**Fungsi Berbagi Tugas (`\_shareTask` )**

  void \_shareTask(BuildContext context) {

    // Implementasi berbagi task ke media sosial

    String taskText =

        'Task: ${todoItem.task}\nStatus: ${todoItem.isCompleted ? 'Completed' : 'Incomplete'}\nDate: ${todoItem.date}';

    Share.share(taskText);

  }

Metode ini mengimplementasikan berbagi item tugas ke media sosial menggunakan paket `share`.

**Fungsi merubah To Do Item ( `\_editTodoItem`)**

  void \_editTodoItem(int index) {

    showDialog(

      context: context,

      builder: (BuildContext context) {

        TextEditingController controller =

            TextEditingController(text: todoItems[index].task);

        return AlertDialog(

          title: Text('Edit Todo Item'),

          content: TextField(

            controller: controller,

            onChanged: (text) {

              // Handle the input text

            },

          ),

          actions: [

            TextButton(

              onPressed: () {

                Navigator.of(context).pop();

              },

              child: Text('Cancel'),

            ),

            TextButton(

              onPressed: () {

                setState(() {

                  todoItems[index].task = controller.text;

                });

                Navigator.of(context).pop();

              },

              child: Text('Save'),

            ),

          ],

        );

      },

    );

  }

Metode ini digunakan untuk menampilkan dialog pengeditan item tugas ketika pengguna memilih untuk mengedit suatu item pada daftar tugas.

**Fungsi menghapus To Do Item( `\_deleteTodoItem`)**

  void \_deleteTodoItem(int index) {

    setState(() {

      todoItems.removeAt(index);

    });

  }

Metode ini digunakan untuk menghapus item tugas dari daftar tugas.

**Metode \_toggleTodoItem(int index)**

  void \_toggleTodoItem(int index) {

    setState(() {

      todoItems[index].isCompleted = !todoItems[index].isCompleted;

    });

  }

Metode ini digunakan untuk mengalihkan status penyelesaian (completed) dari suatu item tugas. Ini akan mengubah status penyelesaian item tugas yang dipilih. Jika status semula true, maka akan diubah menjadi false, dan sebaliknya.

**Penggunaan**

1. Jalankan aplikasi dengan menjalankan `main.dart`.

2. Lihat dan kelola item tugas pada layar utama.

3. Ketuk pada item tugas untuk melihat informasi detail.

4. Edit, hapus, atau alihkan status penyelesaian dari halaman detail.

5. Bagikan item tugas menggunakan ikon berbagi pada halaman detail.