IT 14 FINAL PROJECT DOCUMENTATION

PROJECT TITLE: TASK MANAGEMENT APPLICATION

GROUP MEMBERS:

Defelipe, Denard R.

Jumilla, Jeb C.

Magnase, Christian Joy Z.

I. The Project

The system is all about task management, offering users the ability to list and organize their daily responsibilities. Additionally, the benefits of the system for users are that it simplifies their daily task management, especially for those with busy schedules. With that being said, this makes their lives easier and more organized and helps them stay away from having disorganized tasks that can lead to distractions.

II. The Team

- Defelipe, Denard R.
  - UI Designer
  - Designs the application, simple yet aesthetically.
  - Programmer
- Jumilla, Jeb C.
  - Leader
  - Handles documentation.
  - Main system analyst of the project.
  - Handles Transactional functionality.
  - Handles Firebase.
  - Programmer
- Magnase, Christan Joy Z.
  - Project Manager
  - Handles documentation.
  - Handles Firebase.
  - Assistant system analyst.
  - Handles CRUD.
  - Programmer

III. The Functionalities.

Here's an overview of how these CRUD operations that are utilized in the project:

1. **Create (C):**

-The AddData widget is used to create new tasks. When the user taps the "+" button (FloatingActionButton), they are navigated to the AddData screen where they can input task details and create a new task.

2. **Read (R):**

-The readTasks function retrieves tasks from the Firebase Cloud Firestore using a stream. It listens to changes in the Firestore collection "Task" and maps them to a list of Task objects.

-The buildList function creates a ListView of tasks based on the retrieved task data. This allows the user to view existing tasks.

3. **Update (U):**

-The UpdateData widget allows users to update task details. When the user taps the "Edit" button (pencil icon), they are navigated to the UpdateData screen where they can modify the task details and update them.

-The updateTaskStatus function updates the status of a task (complete or incomplete) based on the user's interaction with the checkbox associated with each task.

4. **Delete (D):**

-The _showdeleteConfirmationDialog function shows a confirmation dialog for deleting a specific task. If the user confirms deletion, the deleteTask function is called to remove the task from the Firestore collection "Task".

-The _showDeleteAllConfirmationDialog function shows a confirmation dialog for deleting all tasks. If the user confirms deletion, the deleteAllTasks function is called to remove all tasks from the Firestore collection "Task".

5. **Transactional Functionality:**

-The app uses Firebase Cloud Firestore to store task data, providing a persistent and real-time database for CRUD operations.

-Transactions are used implicitly when performing CRUD operations on the Firestore collection "Task". For example, when deleting a task or updating its status, the app ensures that these operations are performed atomically and consistently together with the use of pie chart for better understanding of the user.

Overall, the app provides a user interface for managing tasks, allowing users to create, view, update, and delete tasks using Firebase Cloud Firestore for data storage. Transactions are handled by Firebase to maintain data integrity and consistency during CRUD operations.

IV. The User Interface

In the login and registration form, we used a simple design and functions that are still appealing to the users, and in the main form, especially on the home page, we used a pie chart as an overview so that a user can feel the impact of the weight and balance of the tasks, not just by numbers.

Here's a breakdown of the UI components and their functionality:

1. **App Bar:**

-The app bar at the top provides essential navigation and actions.

-It displays the app title "All Tasks" at the center and an icon for opening the drawer (navigation menu) on the left side.

2. **Drawer:**

-The drawer is a side navigation menu that opens from the left, providing options for navigation and actions.

3. **Task Statistics Display:**

-Below the app bar, the UI displays statistics related to tasks, such as the number of completed tasks, incomplete tasks, and total tasks.

4. **Task List Display:**

-The main portion of the UI is dedicated to displaying a list of tasks in a scrollable view.

-Tasks are conditionally displayed based on their completion status, allowing for hiding completed tasks if needed.

5. **Action Buttons:**

-The floating action button (FAB) is used for adding new tasks. When pressed, it navigates the user to the "Add Task" screen.

6. **Dialogs for Confirmation and Task Details:**

-AlertDialogs are used for confirming actions like task deletion and displaying task details.

-The "Delete Confirmation" dialog prompts the user to confirm the deletion of a task.

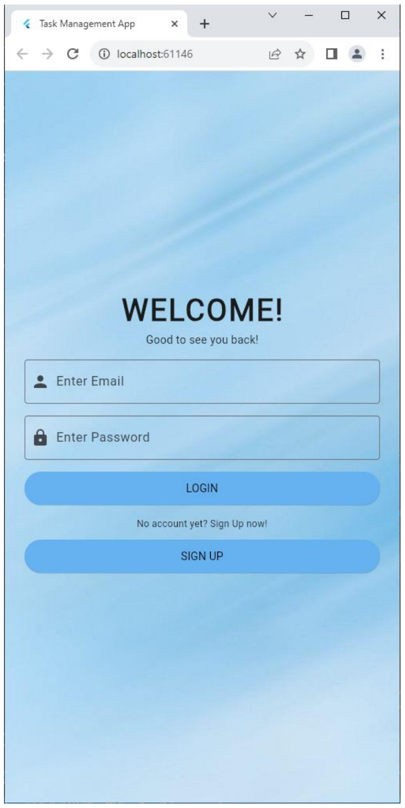
-The "Delete All Tasks Confirmation" dialog confirms the deletion of all tasks.

-The "Task Details" dialog provides additional information about a specific task.

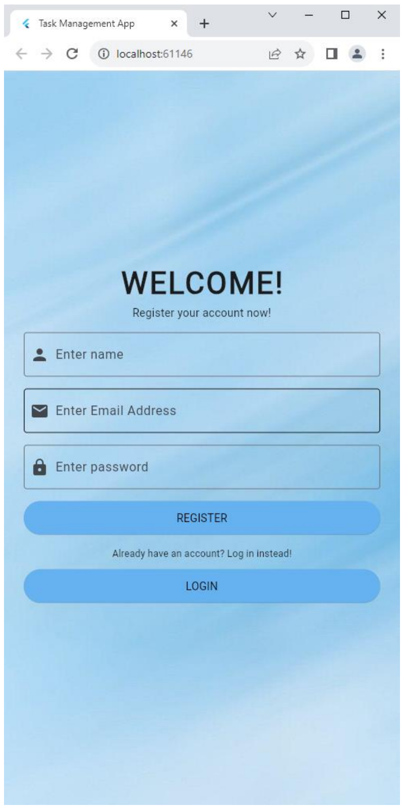
7. **Background Image & Color:**

-The image used for the background adds visual appeal and sets the overall theme for the application together with the chosen color theme which is blue.

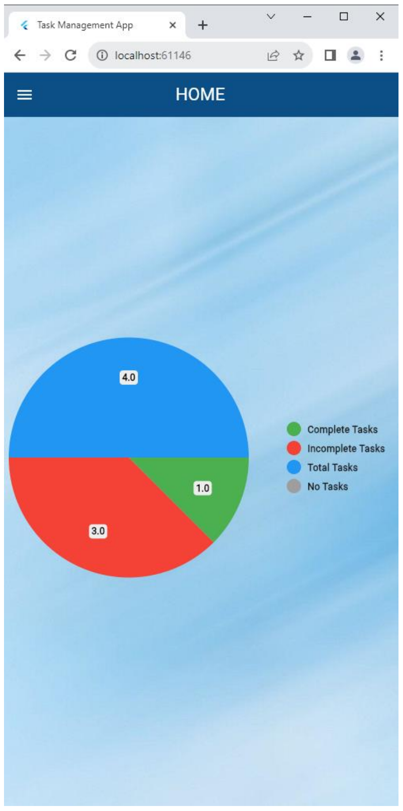
The UI components are designed to provide a clear and intuitive user interface for managing tasks, allowing users to view task statistics, perform CRUD operations on individual tasks, and navigate through the application seamlessly. The use of icons, styled text, and dialogs enhances the user experience and makes the application more interactive and user-friendly.



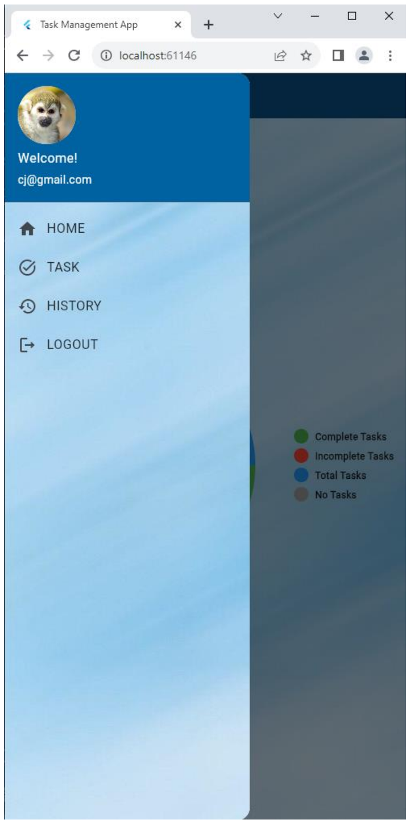
The main purpose of this display is to provide a login interface where users can input their credentials to log in. Additionally, it allows users to navigate to the registration page to create a new account.



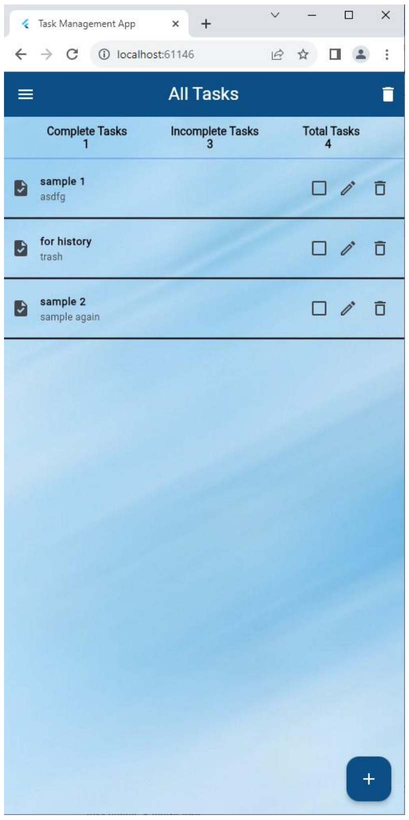
The main purpose of this display is to provide a register interface where users can input their credentials to register and log in directly. Additionally, it allows users to navigate also back to the login page to login an existing account.



The main purpose of this display is to present the task statistics using a pie chart, allowing users to quickly grasp the distribution of complete, incomplete, and total tasks. The chart is dynamic and will update based on the real-time data from the Firebase Cloud Firestore.



The main purpose of this display is to provides essential navigation options and a way for the user to access the homepage, task page, history page and logout the application.



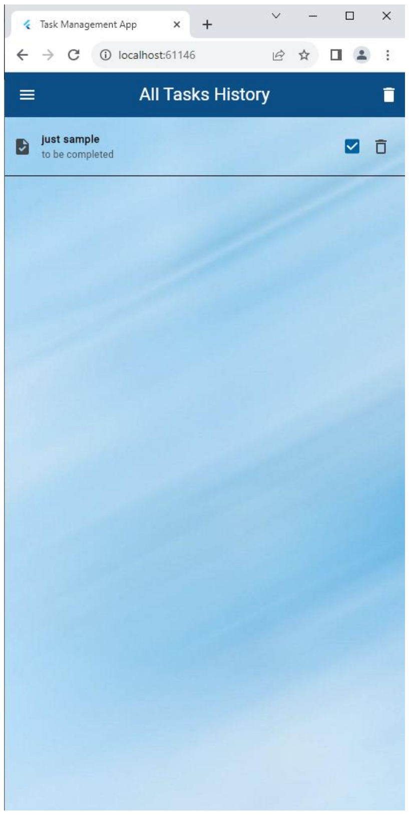
Complete Tasks	Incomplete Tasks	Total Tasks
1	3	4

sample 1
asdfg

for history
trash

sample 2
sample again

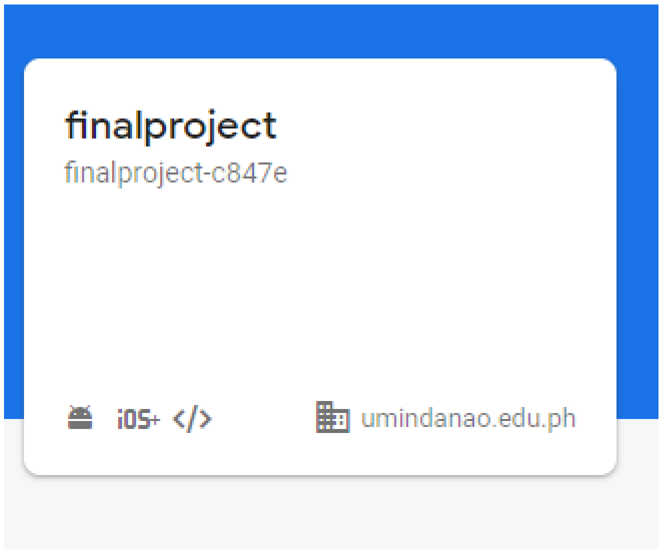
The main purpose of this display is to provide easy access to task management features and important information about task completion status. Users can interact with tasks, view details, and perform actions like editing and deleting tasks.

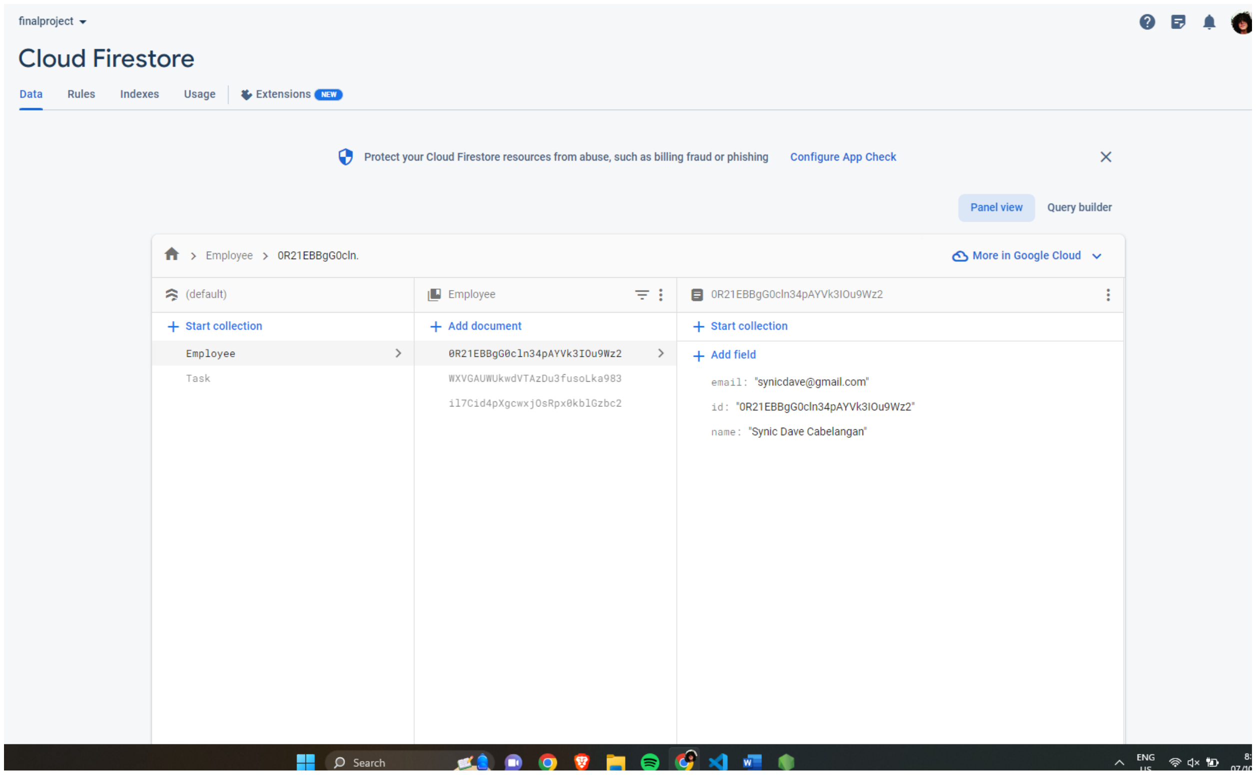


The main purpose of this display is to provide the task history where it will display completed tasks. The user can also update their status, view task details, and delete task.

V. Source Code

Firebase:





finalproject

Cloud Firestore

Data Rules Indexes Usage Extensions NEW

Protect your Cloud Firestore resources from abuse, such as billing fraud or phishing Configure App Check

Panel view Query builder

Employee > 0R21EBBgG0cln.

More in Google Cloud

(default) | Employee | 0R21EBBgG0cln34pAYVk3IOu9Wz2

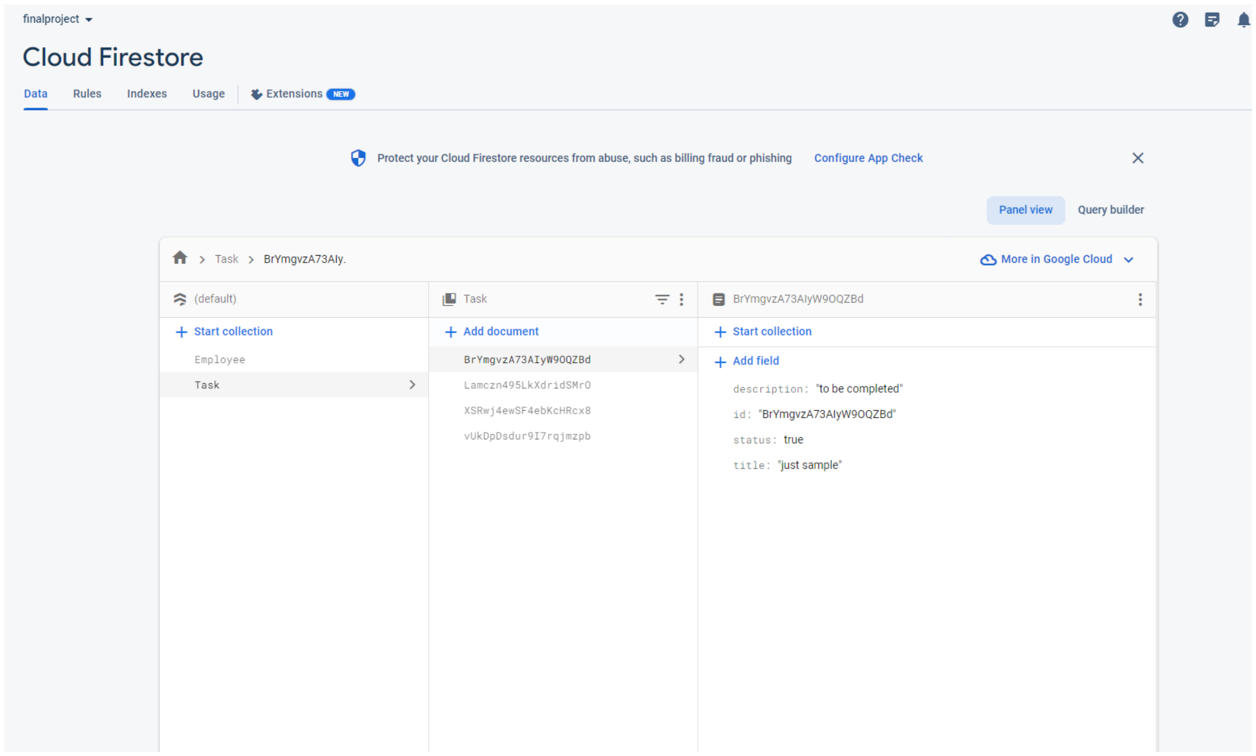
Start collection | Add document | Start collection

Employee | 0R21EBBgG0cln34pAYVk3IOu9Wz2 | Add field

Task | WXVGAUWUkwdVTAzDu3fusoLka983 | email: "synicdave@gmail.com"

i17Cid4pXgcwxjOsRpx0kblGzbc2 | id: "0R21EBBgG0cln34pAYVk3IOu9Wz2"

name: "Synic Dave Cabelangan"



finalproject

Cloud Firestore

Data Rules Indexes Usage Extensions NEW

Protect your Cloud Firestore resources from abuse, such as billing fraud or phishing Configure App Check

Panel view Query builder

Task > BrYmgvzA73Aly.

More in Google Cloud

(default) | Task | BrYmgvzA73AlyW90QZBd

Start collection | Add document | Start collection

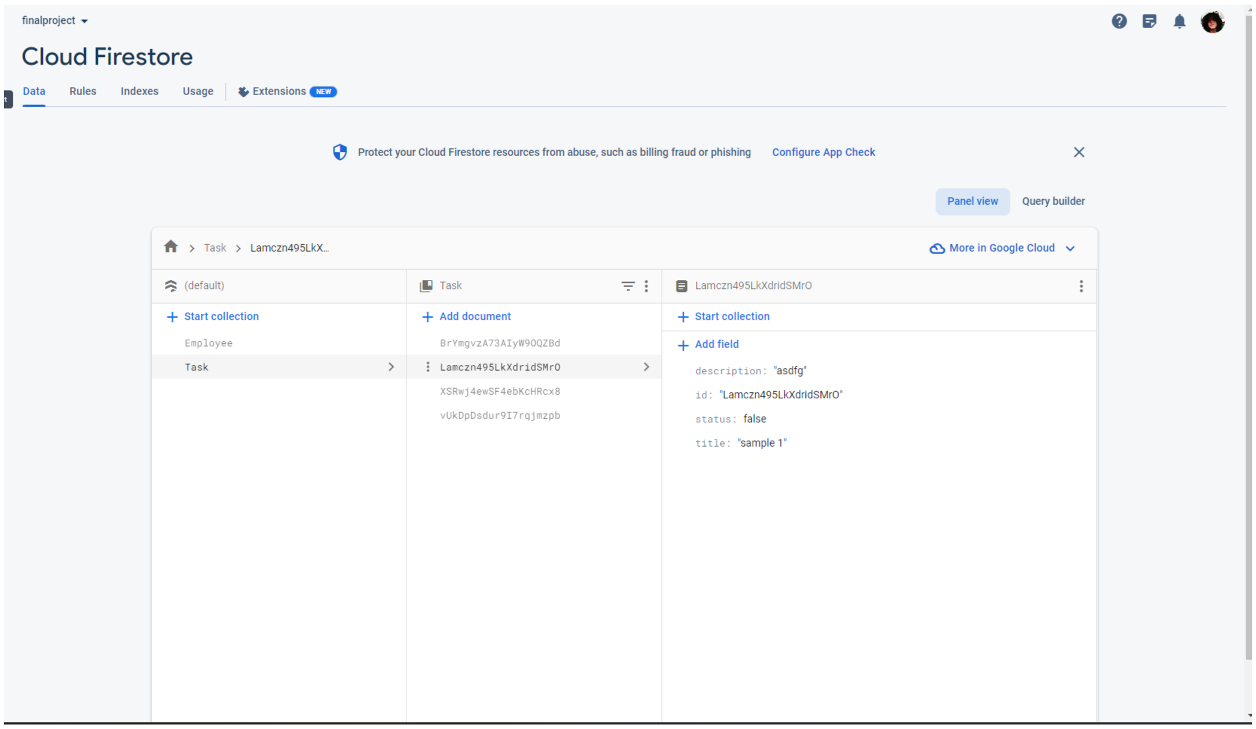
Employee | BrYmgvzA73AIyW90QZBd | Add field

Task | Lamczn495LkXdridSMrO | description: "to be completed"

XSRwj4ewSF4ebKcHRcx8 | id: "BrYmgvzA73AlyW90QZBd"

vUkDpDsdur9I7rqjmzpb | status: true

title: "just sample"


finalproject

Cloud Firestore

Data | Rules | Indexes | Usage | Extensions NEW

Protect your Cloud Firestore resources from abuse, such as billing fraud or phishing Configure App Check

Panel view | Query builder

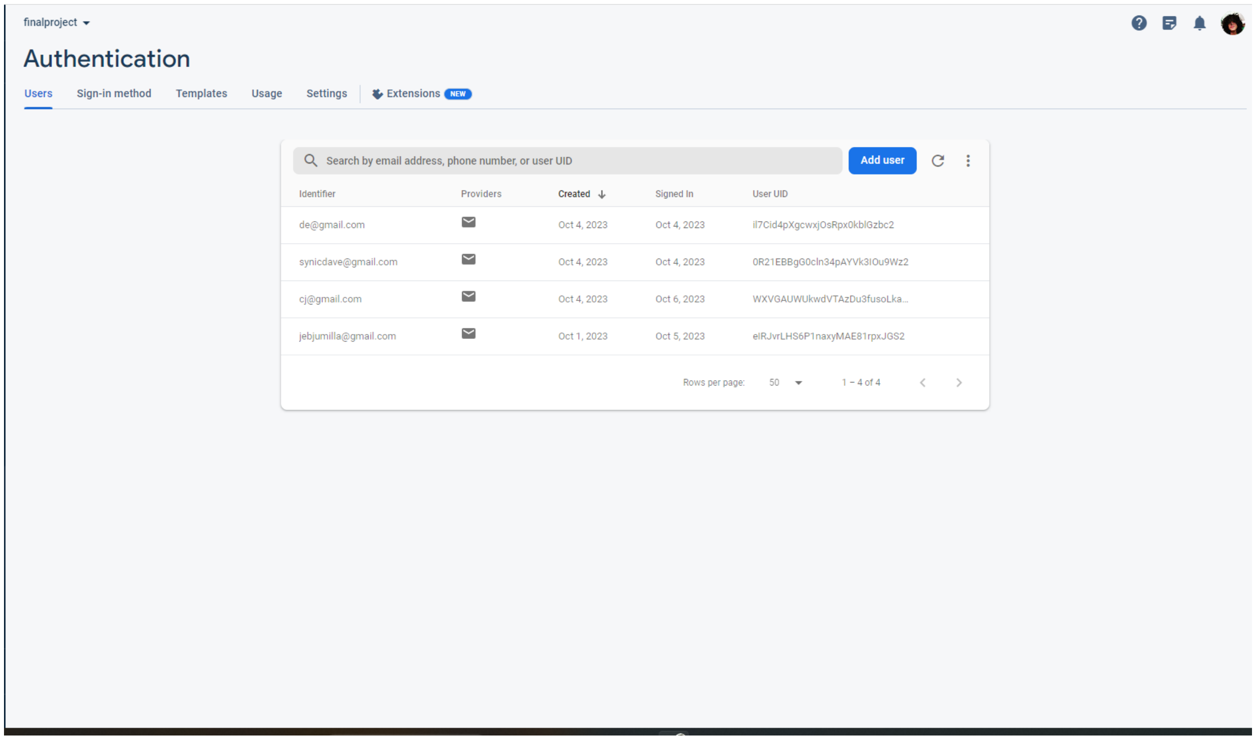
Task > Lamczn495LkX... More in Google Cloud

(default)
- Start collection
- Employee
- Task

Task
- Add document
- BrYmgvzA73AIyW9OQZBd
- Lamczn495LkXdridSMrO
- XSRwj4ewSF4ebKcHRcx8
- vUkDpDsdur9I7rqjmzpb

Lamczn495LkXdridSMrO
- Start collection
- Add field

```
description: "asdfg"
id: "Lamczn495LkXdridSMrO"
status: false
title: "sample 1"
```


finalproject

Authentication

Users | Sign-in method | Templates | Usage | Settings | Extensions NEW

Search by email address, phone number, or user UID Add user

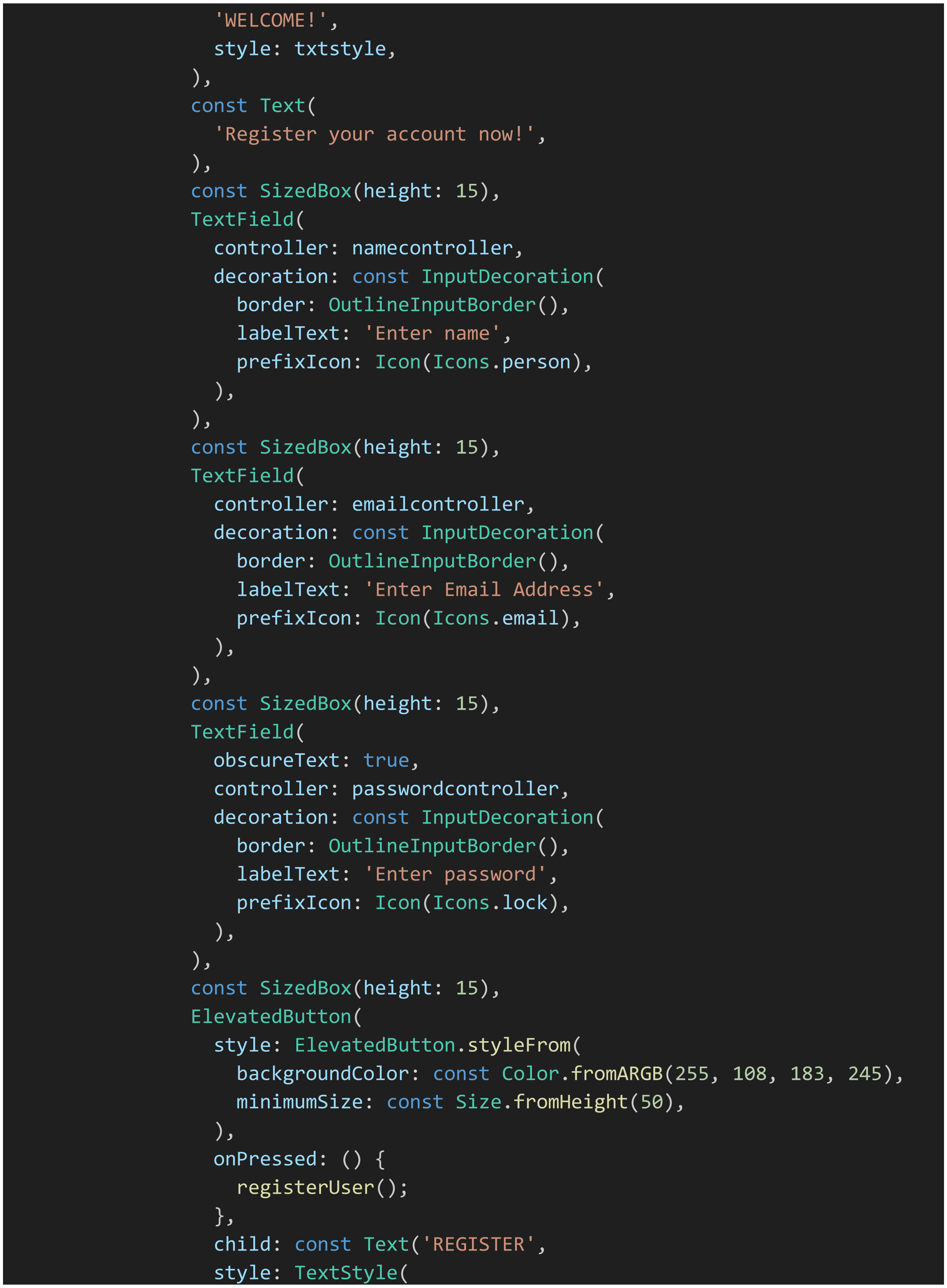
Identifier	Providers	Created	Signed In	User UID
de@gmail.com	✉	Oct 4, 2023	Oct 4, 2023	il7Cid4pXgcwxjOsRpx0kblGzbc2
synicdave@gmail.com	✉	Oct 4, 2023	Oct 4, 2023	0R21EBBgG0cln34pAYVk3lOu9Wz2
cj@gmail.com	✉	Oct 4, 2023	Oct 6, 2023	WXVGAUWUkwdVTAzDu3fusoLka...
jebjumilla@gmail.com	✉	Oct 1, 2023	Oct 5, 2023	elRJvrLHS6P1naxyMAE81rpxJGS2

Rows per page: 50 1 – 4 of 4

Source Code:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:task_manager_app/authenticator.dart';
```

```dart
import 'package:task_manager_app/employee.dart';
import 'package:task_manager_app/login.dart';

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

  @override
  State<Register> createState() => _RegisterState();
}

class _RegisterState extends State<Register> {
  TextEditingController namecontroller = TextEditingController();
  TextEditingController emailcontroller = TextEditingController();
  TextEditingController passwordcontroller = TextEditingController();
  late String errormessage;
  late bool isError;

  @override
  void initState() {
    errormessage = "This is an error";
    isError = false;
    super.initState();
  }

  @override
  void dispose() {
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: const Color.fromARGB(255, 182, 241, 252),
      body: Container(
        decoration: const BoxDecoration(
        image: DecorationImage(
          image: AssetImage('assets/images/finalbg.jpg'),
          fit: BoxFit.cover),),
        child: Center(
          child: Container(
            margin: const EdgeInsets.symmetric(horizontal: 25),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(
```



```dart
                  'WELCOME!',
                  style: txtstyle,
                ),
                const Text(
                  'Register your account now!',
                ),
                const SizedBox(height: 15),
                TextField(
                  controller: namecontroller,
                  decoration: const InputDecoration(
                    border: OutlineInputBorder(),
                    labelText: 'Enter name',
                    prefixIcon: Icon(Icons.person),
                  ),
                ),
                const SizedBox(height: 15),
                TextField(
                  controller: emailcontroller,
                  decoration: const InputDecoration(
                    border: OutlineInputBorder(),
                    labelText: 'Enter Email Address',
                    prefixIcon: Icon(Icons.email),
                  ),
                ),
                const SizedBox(height: 15),
                TextField(
                  obscureText: true,
                  controller: passwordcontroller,
                  decoration: const InputDecoration(
                    border: OutlineInputBorder(),
                    labelText: 'Enter password',
                    prefixIcon: Icon(Icons.lock),
                  ),
                ),
                const SizedBox(height: 15),
                ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    backgroundColor: const Color.fromARGB(255, 108, 183, 245),
                    minimumSize: const Size.fromHeight(50),
                  ),
                  onPressed: () {
                    registerUser();
                  },
                  child: const Text('REGISTER',
                  style: TextStyle(
                        color: Colors.black,
                      ),),
                    ),
                    (isError)
                        ? Text(
                            errormessage,
                            style: errortxtstyle,
                          )
                        : Container(),
                      const SizedBox(height: 15),
                  const Text(
                    'Already have an account? Log in instead!',
                    style: TextStyle(fontSize: 12),
                  ),
                  Container(
                    height: 12,
                  ),
                  ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: const Color.fromARGB(255, 108, 183, 245),
                      minimumSize: const Size.fromHeight(50),
                    ),
                    onPressed: () {
                      Navigator.of(context).push(
                        MaterialPageRoute(
                          builder: (context) => const Login(),
                        ),
                      );
                    },
                    child: const Text(
                      'LOGIN',
                      style: TextStyle(color: Colors.black),
                    ),
                  ),
                  const SizedBox(height: 15),
                  ],
                ),
              ),
            ),
          ),
        );
  }

  var errortxtstyle = const TextStyle(
    fontWeight: FontWeight.bold,
    color: Colors.red,
    letterSpacing: 1,
    fontSize: 18,
  );
  var txtstyle = const TextStyle(
    fontWeight: FontWeight.bold,
    letterSpacing: 2,
    fontSize: 38,
  );

  Future createUser() async {
    final user = FirebaseAuth.instance.currentUser!;
    final userid = user.uid;
    final docUser =
FirebaseFirestore.instance.collection('Employee').doc(userid);

    final employee = Employee(
      id: userid,
      name: namecontroller.text,
      email: emailcontroller.text,
    );

    final json = employee.toJson();
    await docUser.set(json);

    goToAuthenticator();

  }

  Future registerUser() async {
    try {
  await FirebaseAuth.instance.createUserWithEmailAndPassword(
    email: emailcontroller.text.trim(),
    password: passwordcontroller.text.trim(),
    );
    createUser();
      setState((){
        errormessage = "";
      });
} on FirebaseAuthException catch (e) {
  print(e);
  setState(() {
    errormessage = e.message.toString();
  });
}
```

```dart
  }

  goToAuthenticator() {
    Navigator.of(context).push(
      MaterialPageRoute(
        builder: (context) => const Authenticator(),
      ),
    );
  }
}
```

```dart
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:task_manager_app/register.dart';

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

  @override
  State<Login> createState() => _LoginState();
}

class _LoginState extends State<Login> {
  TextEditingController usernamecontroller = TextEditingController();
  TextEditingController passwordcontroller = TextEditingController();
  late String errormessage;
  late bool isError;

  @override
  void initState() {
    errormessage = "This is an error";
    isError = false;
    super.initState();
  }

  @override
  void dispose() {
    super.dispose();
  }

  @override
  void _showDialog(String title, String message) {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: Text(title),
      content: Text(message),
      actions: <Widget>[
        TextButton(
          onPressed: () {
            Navigator.of(context).pop(); // Close the dialog
          },
          child: Text('OK'),
        ),
      ],
    ),
  );
}

Widget build(BuildContext context) {
  return Scaffold(
    //backgroundColor: const Color.fromARGB(255, 182, 241, 252),
    body: Container(
      decoration: const BoxDecoration(
        image: DecorationImage(
          image: AssetImage('assets/images/finalbg.jpg'),
          fit: BoxFit.cover),),
      child: Center(
        child: Container(
          margin: const EdgeInsets.symmetric(horizontal: 25),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                'WELCOME!',
                style: txtstyle,
              ),
              const Text(
                'Good to see you back!',
              ),
              const SizedBox(height: 15),
              TextField(
                controller: usernamecontroller,
                decoration: const InputDecoration(
                  border: OutlineInputBorder(),
                  labelText: 'Enter Email',
                  prefixIcon: Icon(Icons.person),
                ),
              ),
              const SizedBox(height: 15),
              TextField(
                obscureText: true,
                controller: passwordcontroller,
                decoration: const InputDecoration(
                  border: OutlineInputBorder(),
                  labelText: 'Enter Password',
                  prefixIcon: Icon(Icons.lock),
                ),
              ),
              const SizedBox(height: 15),
              ElevatedButton(
                style: ElevatedButton.styleFrom(
                  backgroundColor: const Color.fromARGB(255, 108, 183, 245),
                  minimumSize: const Size.fromHeight(50),
                ),
                onPressed: () async {
                  try {
                    await FirebaseAuth.instance.signInWithEmailAndPassword(
                      email: usernamecontroller.text,
                      password: passwordcontroller.text,
                    );
                    setState(() {
                      errormessage = "";
                    });
                    _showDialog('Login Successful', 'You have successfully logged
in.');

                  } on FirebaseAuthException catch (e) {
                    print(e);
                    setState(() {
                      errormessage = e.message.toString();
                    });
                    _showDialog('Invalid Email or Password!', errormessage);
                  }
                },
                child: const Text(
                  'LOGIN',
                  style: TextStyle(color: Colors.black),
                ),
              ),
              const SizedBox(height: 15),
              const Text(
                'No account yet? Sign Up now!',
                style: TextStyle(fontSize: 12),
```

```dart
              ),
              Container(
                height: 12,
              ),
              ElevatedButton(
                style: ElevatedButton.styleFrom(
                  backgroundColor: const Color.fromARGB(255, 108, 183, 245),
                  minimumSize: const Size.fromHeight(50),
                ),
                onPressed: () {
                  Navigator.of(context).push(
                    MaterialPageRoute(
                      builder: (context) => const Register(),
                    ),
                  );
                },
                child: const Text(
                  'SIGN UP',
                  style: TextStyle(color: Colors.black),
                ),
              ),
              const SizedBox(height: 15),
            ],
          ),
        ),
      ),
    ),
  );
}


  var errortxtstyle = const TextStyle(
    fontWeight: FontWeight.bold,
    color: Colors.red,
    letterSpacing: 1,
    fontSize: 18,
  );
  var txtstyle = const TextStyle(
    fontWeight: FontWeight.bold,
    letterSpacing: 2,
    fontSize: 38,
  );

 Future checkLogin(username, password) async {
    showDialog(
      context: context,
      useRootNavigator: false,
      barrierDismissible: false,
      builder: (context) => const Center(
        child: CircularProgressIndicator(),
        ),);

    try {
      await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: username,
        password: password,
        );
        setState(() {
          errormessage="";
        });
  } on FirebaseAuthException catch (e) {
    print(e);
    setState(() {
      errormessage = e.message.toString();
    });
}
  Navigator.pop(context);
  }

}
```

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:task_manager_app/home.dart';
import 'login.dart';

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

  @override
  State<Authenticator> createState() => _AuthenticatorState();
}

class _AuthenticatorState extends State<Authenticator> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: StreamBuilder<User?>(
      stream: FirebaseAuth.instance.authStateChanges(),
      builder: (context, snapshot){
        if (snapshot.connectionState == ConnectionState.waiting){
          return const Center(
            child: CircularProgressIndicator(),
            );
        }
        else if (snapshot.hasError){
          return const Center(
            child: Text('Sonething went wrong!'),
            );
        }
        else if (snapshot.hasData){

          return const Home();
        }
        else {
          return const Login();
        }
      },
      ),
    );
  }
}
```

```
class Task {
  final id;
  final String title;
   bool status;
  final String description;

  Task({
    required this.id,
    required this.title,
    required this.status,
    required this.description,

  });

  Map<String, dynamic> toJson() => {
  'id': id,
  'title': title,
  'description': description,
  'status': status,
};


  static Task fromJson(Map<String, dynamic> json) => Task(
    id: json['id'],
    title: json['title'],
  status: json['status'] ?? false,
    description: json['description'],

    );

}



import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:task_manager_app/authenticator.dart';
import 'firebase_options.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
);
  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Task Management App',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: const Color.fromARGB(255,
11, 149, 241)),
        useMaterial3: true,
      ),
      home: const Authenticator(),
      //home: const AllData(),
      //home: const AddData(),
```

```dart
      //home: const Register(),
    );

  }
}





// File generated by FlutterFire CLI.
// ignore_for_file: lines_longer_than_80_chars,
avoid_classes_with_only_static_members
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      return web;
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        return ios;
      case TargetPlatform.macOS:
        return macos;
      case TargetPlatform.windows:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for windows - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
```

```
    case TargetPlatform.linux:
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for linux - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    default:
      throw UnsupportedError(
        'DefaultFirebaseOptions are not supported for this platform.',
      );
  }
}

static const FirebaseOptions web = FirebaseOptions(
  apiKey: 'AIzaSyDeps5p1DWEsgQKCu2auLfoqg16weHgNoo',
  appId: '1:185023998163:web:6b40f387c6f1a54eeb4c26',
  messagingSenderId: '185023998163',
  projectId: 'finalproject-c847e',
  authDomain: 'finalproject-c847e.firebaseapp.com',
  storageBucket: 'finalproject-c847e.appspot.com',
  measurementId: 'G-ZGG627TTGB',
);

static const FirebaseOptions android = FirebaseOptions(
  apiKey: 'AIzaSyB8DGoXE9S-qVNHRj8AzVqCoChSPkCwRZo',
  appId: '1:185023998163:android:9ae1791a0cfe7128eb4c26',
  messagingSenderId: '185023998163',
  projectId: 'finalproject-c847e',
  storageBucket: 'finalproject-c847e.appspot.com',
);

static const FirebaseOptions ios = FirebaseOptions(
  apiKey: 'AIzaSyBSCG3M7fLTMyxdxejDrNo_iNXjY5sZM78',
  appId: '1:185023998163:ios:cb2abd87719d0cedeb4c26',
  messagingSenderId: '185023998163',
  projectId: 'finalproject-c847e',
  storageBucket: 'finalproject-c847e.appspot.com',
  iosBundleId: 'com.example.flutterFinalProject',
);

static const FirebaseOptions macos = FirebaseOptions(
  apiKey: 'AIzaSyBSCG3M7fLTMyxdxejDrNo_iNXjY5sZM78',
  appId: '1:185023998163:ios:1ea8855b19afcf4eeb4c26',
  messagingSenderId: '185023998163',
  projectId: 'finalproject-c847e',
  storageBucket: 'finalproject-c847e.appspot.com',
    iosBundleId: 'com.example.flutterFinalProject.RunnerTests',
  );
}
```

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:task_manager_app/sidebar.dart';
import 'package:task_manager_app/task.dart';
import 'package:pie_chart/pie_chart.dart';
class Home extends StatefulWidget {
  const Home({super.key});

  @override
  State<Home> createState() => _AllDataState();
}

Stream<List<Task>> readTasks(){
  return FirebaseFirestore.instance
  .collection('Task')
  .snapshots()
  .map((snapshot) => snapshot.docs.map((doc) => Task.fromJson(doc.data(),
  ),
  )
  .toList(),
  );
}

class _AllDataState extends State<Home> {
 int completedTasksCount = 0;
  int incompleteTasksCount = 0;
  int totalTasksCount = 0;

  Map<String, double> dataMap = {
    'Complete Tasks': 0,
    'Incomplete Tasks': 0,
    'Total Tasks': 0,
    'No Tasks': 0,
  };

  @override
  void initState() {
    super.initState();
    // Fetch tasks and calculate counts
    readTasks().listen((tasks) {
      setState(() {
        completedTasksCount = tasks.where((task) => task.status).length;
        incompleteTasksCount = tasks.where((task) => !task.status).length;
        totalTasksCount = tasks.length;

        // Update dataMap after counts have been updated
        dataMap = {
          'Complete Tasks': completedTasksCount.toDouble(),
          'Incomplete Tasks': incompleteTasksCount.toDouble(),
          'Total Tasks': totalTasksCount.toDouble(),
          'No Tasks' : 0,
        };
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: const Color.fromARGB(255, 182, 241, 252),
      drawer: sidebar(),
      appBar: AppBar(
        iconTheme: const IconThemeData(color: Colors.white),
        centerTitle: true,
        title: const Text(
          'HOME',
          style: TextStyle(color: Colors.white),
        ),
        backgroundColor: const Color.fromARGB(255, 11, 79, 134),

      ),
      body: Container(
        decoration: const BoxDecoration(
          image: DecorationImage(
            image: AssetImage('assets/images/finalbg.jpg'),
            fit: BoxFit.cover,
          ),
        ),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: [
            const SizedBox(height: 10,),
            Container(
```

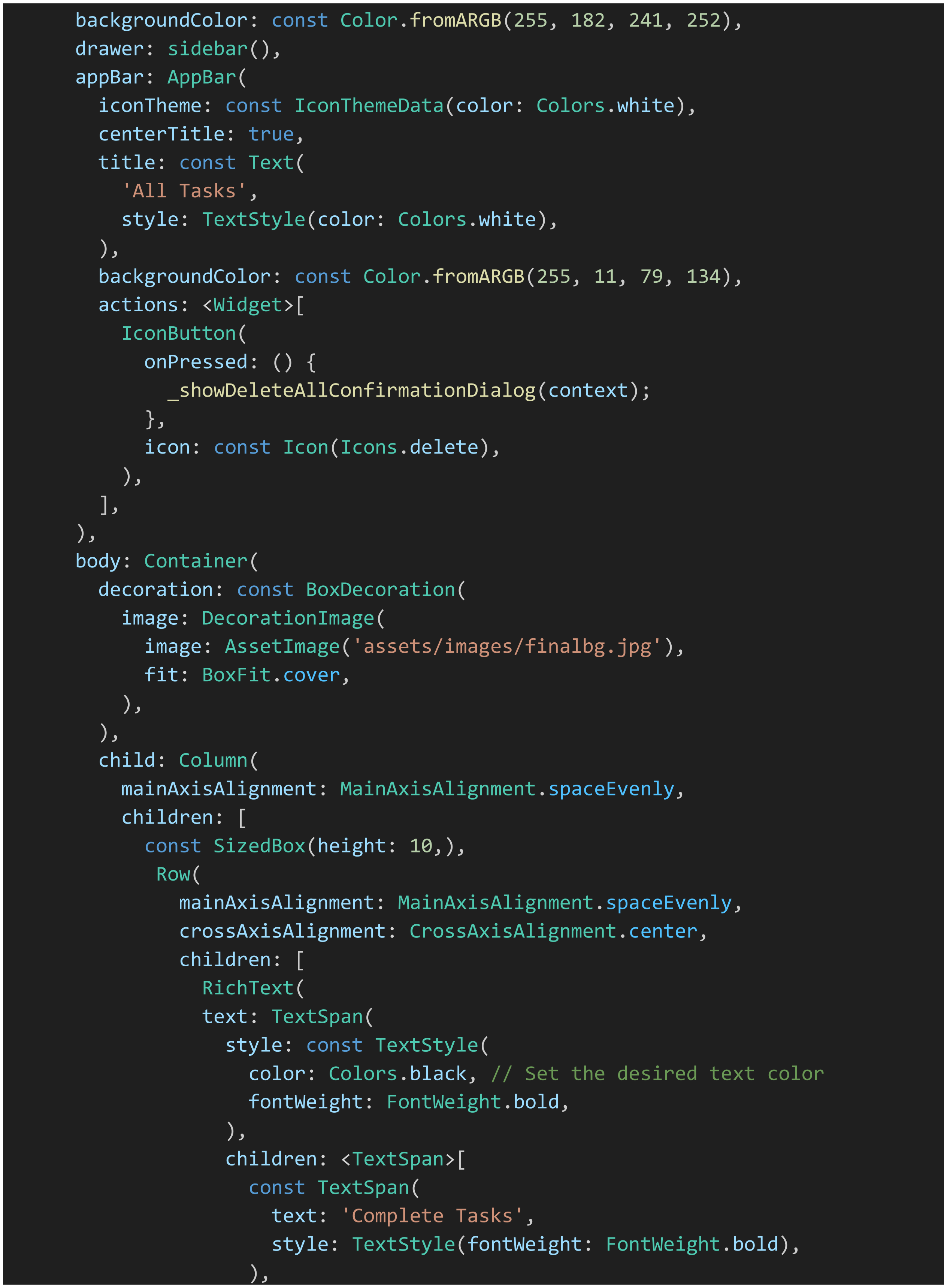
```dart
              width: 1000, // Set the width as per your requirements
              height: 600,
              child: PieChart(
                dataMap: dataMap,
              colorList: const [Color.fromRGBO(76, 175, 80, 1), Colors.red,
Colors.blue,Colors.grey],
              chartRadius: MediaQuery.of(context).size.width / 1.0,
              ),
              ) ,

            const SizedBox(height: 20,),

          ],
        ),
      ),
    );
  }


}
```

```dart
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:task_manager_app/add_data.dart';
import 'package:task_manager_app/login.dart';
import 'package:task_manager_app/sidebar.dart';
import 'package:task_manager_app/task.dart';
import 'package:task_manager_app/update_data.dart';
class AllData extends StatefulWidget {
  const AllData({super.key});

  @override
  State<AllData> createState() => _AllDataState();
}

Stream<List<Task>> readTasks(){
  return FirebaseFirestore.instance
  .collection('Task')
  .snapshots()
  .map((snapshot) => snapshot.docs.map((doc) => Task.fromJson(doc.data(),
  ),
  )
  .toList(),
  );
}

Future deleteTask(String id) async {
  final docTask = FirebaseFirestore.instance.collection('Task').doc(id);
  docTask.delete();
}

Future<void> deleteAllTasks() async {
  try {
    // Fetch all tasks from the collection
    QuerySnapshot querySnapshot = await
FirebaseFirestore.instance.collection('Task').get();

    // Iterate through each task and delete it
    for (QueryDocumentSnapshot documentSnapshot in querySnapshot.docs) {
      await
FirebaseFirestore.instance.collection('Task').doc(documentSnapshot.id).delete();
    }

    print('All tasks deleted successfully.');
  } catch (e) {
    print('Error deleting tasks: $e');
  }
}

Future updateTaskStatus(String id, bool newStatus) async {
  final docTask = FirebaseFirestore.instance.collection('Task').doc(id);
  await docTask.update({'status': newStatus}); // Update the "status" field
}
Future<void> _showdeleteConfirmationDialog(BuildContext context, String id) async
{
  showDialog<void>(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return AlertDialog(
        title: const Text('Delete Confirmation'),
        content: const Text('Delete file?'),
        actions: <Widget>[
          TextButton(
            child: const Text('No'),
            onPressed: () {
              Navigator.of(context).pop(); // Close the dialog
            },
          ),
          TextButton(
            child: const Text('Yes'),
            onPressed: () async {
              Navigator.of(context).pop();
              await deleteTask(id);  // Call deleteTask to delete the task
            },
          ),
        ],
      );
    },
  );
}

Future<void> _showDeleteAllConfirmationDialog(BuildContext context) async {
  showDialog<void>(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return AlertDialog(
        title: const Text('Delete All Tasks Confirmation'),
        content: const Text('Are you sure you want to delete all tasks?'),
        actions: <Widget>[
          TextButton(
            child: const Text('Cancel'),
            onPressed: () {
              Navigator.of(context).pop(); // Close the dialog
            },
          ),
          TextButton(
            child: const Text('Delete All'),
            onPressed: () async {
              await deleteAllTasks();  // Call deleteAllTasks to delete all tasks
              Navigator.of(context).pop(); // Close the dialog
            },
          ),
        ],
      );
    },
  );
}
```

```dart
Future<void> _showTaskDetailsDialog(BuildContext context, Task task) async {
  return showDialog<void>(
    context: context,
    barrierDismissible: false, // User must tap a button to close the dialog
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text(task.title),
        content: SingleChildScrollView(
          child: ListBody(
            children: <Widget>[
              Text('Description:\n${task.description}'),
              // Add more details here as needed
            ],
          ),
        ),
        actions: <Widget>[
          TextButton(
            child: Text('Close'),
            onPressed: () {
              Navigator.of(context).pop(); // Close the dialog
            },
          ),
        ],
      );
    },
  );
}
Future<void> handleSignOut(BuildContext context) async {
  try {
    await FirebaseAuth.instance.signOut(); // Sign the user out
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => const Login()), // Navigate back to
the Login screen
    );
  } catch (e) {
    print('Error signing out: $e');
  }
}


class _AllDataState extends State<AllData> {
 int completedTasksCount = 0;
  int incompleteTasksCount = 0;
  int totalTasksCount = 0;
```

```
Map<String, double> dataMap = {
  'Complete Tasks': 0,
  'Incomplete Tasks': 0,
  'Total Tasks': 0,
};

@override
void initState() {
  super.initState();
  // Fetch tasks and calculate counts
  readTasks().listen((tasks) {
    setState(() {
      completedTasksCount = tasks.where((task) => task.status).length;
      incompleteTasksCount = tasks.where((task) => !task.status).length;
      totalTasksCount = tasks.length;

      // Update dataMap after counts have been updated
      dataMap = {
        'Complete Tasks': completedTasksCount.toDouble(),
        'Incomplete Tasks': incompleteTasksCount.toDouble(),
        'Total Tasks': totalTasksCount.toDouble(),
      };
    });
  });
}
Widget buildList(Task task) => Visibility(
visible: !task.status, // Hide the ListTile when task.status is true
child: Container(
  decoration: const BoxDecoration(border: Border(
  bottom: BorderSide(width: 2.0, color: Colors.black))),
  child: ListTile(
    contentPadding: const EdgeInsets.fromLTRB(10, 10, 10, 10),
    leading: const Icon(Icons.task),
    title: Text(
      task.title,
      maxLines: 1,
      overflow: TextOverflow.ellipsis,
      style: const TextStyle(color: Colors.black, fontWeight: FontWeight.bold),
    ),
    subtitle: Text(
      task.description,
      maxLines: 1,
      overflow: TextOverflow.ellipsis,
    ),
```

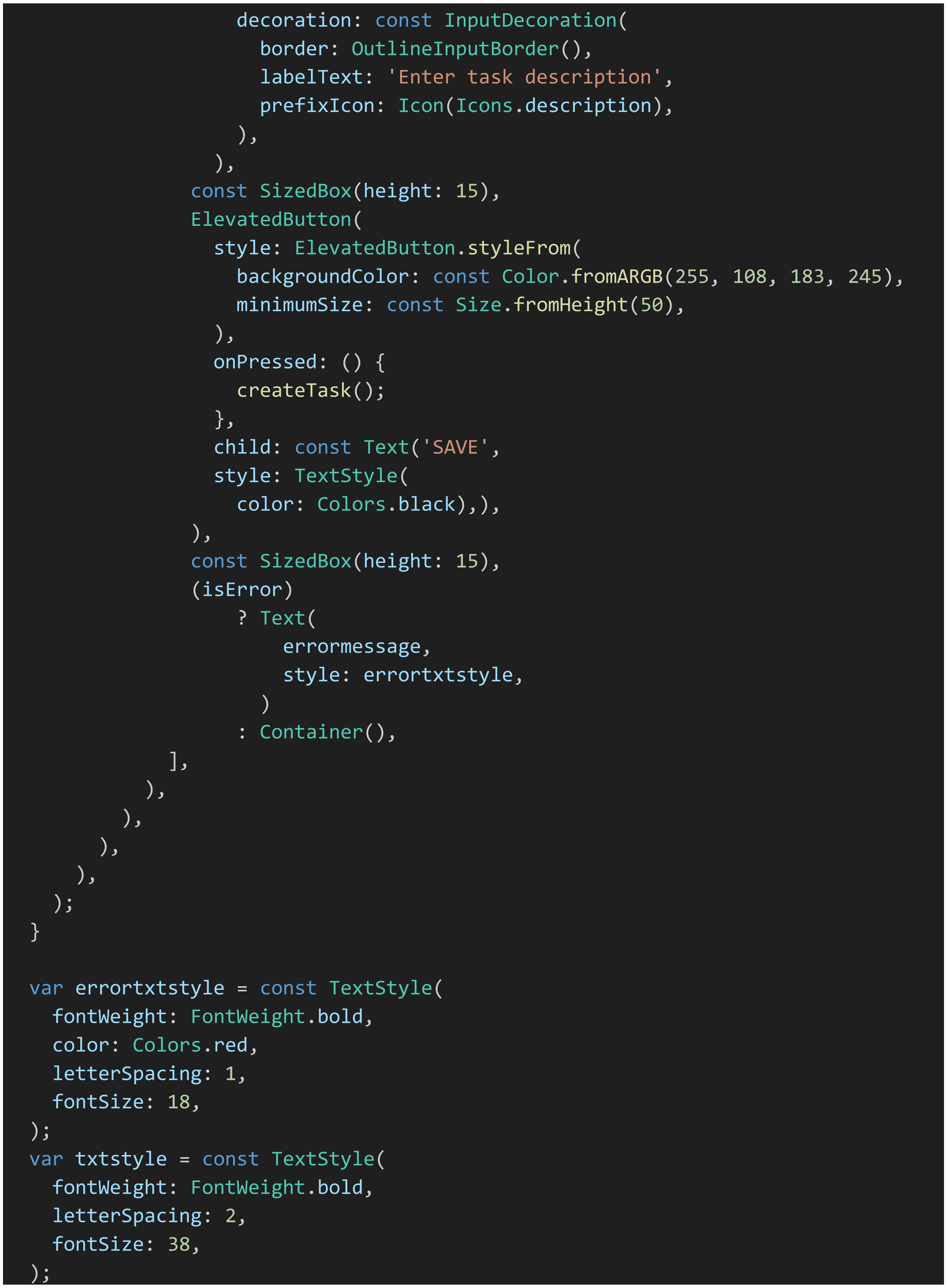
```dart
      dense: true,
      onTap: () {
        _showTaskDetailsDialog(context, task);
      },
      trailing: Row(
        mainAxisSize: MainAxisSize.min,
        children: [
          Checkbox(
            value: task.status,
            onChanged: (newValue) {
              if (newValue != null) {
                updateTaskStatus(task.id, newValue);
                setState(() {
                  task.status = newValue;
                });
              }
            },
          ),
          IconButton(
            onPressed: () {
              Navigator.of(context).push(
                MaterialPageRoute(
                  builder: (context) => UpdateData(task: task),
                ),
              );
            },
            icon: const Icon(Icons.edit_outlined),
          ),
          IconButton(
            onPressed: () {
              _showdeleteConfirmationDialog(context, task.id);
            },
            icon: const Icon(Icons.delete_outlined),
          ),
        ],
      ),
    ),
  ),
);



  @override
  Widget build(BuildContext context) {
    return Scaffold(
```


```
      backgroundColor: const Color.fromARGB(255, 182, 241, 252),
      drawer: sidebar(),
      appBar: AppBar(
        iconTheme: const IconThemeData(color: Colors.white),
        centerTitle: true,
        title: const Text(
          'All Tasks',
          style: TextStyle(color: Colors.white),
        ),
        backgroundColor: const Color.fromARGB(255, 11, 79, 134),
        actions: <Widget>[
          IconButton(
            onPressed: () {
              _showDeleteAllConfirmationDialog(context);
            },
            icon: const Icon(Icons.delete),
          ),
        ],
      ),
      body: Container(
        decoration: const BoxDecoration(
          image: DecorationImage(
            image: AssetImage('assets/images/finalbg.jpg'),
            fit: BoxFit.cover,
          ),
        ),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: [
            const SizedBox(height: 10,),
             Row(
               mainAxisAlignment: MainAxisAlignment.spaceEvenly,
               crossAxisAlignment: CrossAxisAlignment.center,
               children: [
                 RichText(
                 text: TextSpan(
                   style: const TextStyle(
                     color: Colors.black, // Set the desired text color
                     fontWeight: FontWeight.bold,
                   ),
                   children: <TextSpan>[
                     const TextSpan(
                       text: 'Complete Tasks',
                       style: TextStyle(fontWeight: FontWeight.bold),
                     ),
                     TextSpan(
                       text: '\n             $completedTasksCount',
                       style: const TextStyle(fontWeight: FontWeight.bold,),
                     ),
                   ],
                 ),
               ),

                 RichText(
                 text: TextSpan(
                   style: const TextStyle(
                     color: Colors.black, // Set the desired text color
                     fontWeight: FontWeight.bold,
                   ),
                   children: <TextSpan>[
                     const TextSpan(
                       text: 'Incomplete Tasks',
                       style: TextStyle(fontWeight: FontWeight.bold),
                     ),
                     TextSpan(
                       text: '\n             $incompleteTasksCount',
                       style: const TextStyle(fontWeight: FontWeight.bold),
                     ),
                   ],
                 ),
               ),
                 RichText(
                 text: TextSpan(
                   style: const TextStyle(
                     color: Colors.black, // Set the desired text color
                     fontWeight: FontWeight.bold,
                   ),
                   children: <TextSpan>[
                     const TextSpan(
                       text: 'Total Tasks',
                       style: TextStyle(fontWeight: FontWeight.bold),
                     ),
                     TextSpan(
                       text: '\n        $totalTasksCount',
                       style: const TextStyle(fontWeight: FontWeight.bold),
                     ),
                   ],
                 ),
               ),
               ],
```

```dart
              ),
              const SizedBox(height: 10,),
              const Divider(height: .5, color: Color.fromARGB(255, 94, 140, 238)),
             Expanded(
  child: StreamBuilder<List<Task>>(
  stream: readTasks(),
  builder: (context, snapshot) {
    if (snapshot.hasError) {
      return Text('Something went wrong! ${snapshot.error}');
    } else if (snapshot.hasData) {
      final tasks = snapshot.data!;

      return ListView(
        children: tasks.map(buildList).toList(),
      );
    } else {
      return const Center(
        child: CircularProgressIndicator(),
      );
    }
  },
)
),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          Navigator.of(context).push(
            MaterialPageRoute(
              builder: (context) => const AddData(),
            ),
          );
        },
        backgroundColor: const Color.fromARGB(255, 11, 79, 134),
        foregroundColor: Colors.white,
        child: const Icon(Icons.add),
      ),
    );
  }


}
```

```dart
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:task_manager_app/task.dart';

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

  @override
  State<AddData> createState() => _AddDataState();
}


class _AddDataState extends State<AddData> {
  TextEditingController titlecontroller = TextEditingController();
  TextEditingController descriptioncontroller = TextEditingController();
  late String errormessage;
  late bool isError;

  @override
  void initState() {
    errormessage = "This is an error";
    isError = false;
    super.initState();
  }

  @override
  void dispose() {
    super.dispose();
  }

Future createTask() async{
  final docTask = FirebaseFirestore.instance.collection('Task').doc();
  final newTask = Task(
    id: docTask.id,
    title: titlecontroller.text,
    description: descriptioncontroller.text,
    status: false
  );
  final json = newTask.toJson();
  await docTask.set(json);

  setState((){
    titlecontroller.text="";
    descriptioncontroller.text="";
    Navigator.pop(context);
  });}
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: const Color.fromARGB(255, 182, 241, 252),
    appBar: AppBar(
     backgroundColor: const Color.fromARGB(255, 11, 79, 134),
      title: const Text('',
      style: TextStyle(color: Colors.white),),
      leading: IconButton(
        color: Colors.white,
        onPressed: (){
          Navigator.pop(context);
        },
        icon: const Icon(Icons.arrow_back),
        ),
    ),
    body: Container(
      decoration: const BoxDecoration(
      image: DecorationImage(
        image: AssetImage('assets/images/finalbg.jpg'),
        fit: BoxFit.cover),),
      child: Center(
        child: Container(
          margin: const EdgeInsets.symmetric(horizontal: 25),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                'ADD TASK',
                style: txtstyle,
              ),
              const SizedBox(height: 15),
              TextField(
                controller: titlecontroller,
                decoration: const InputDecoration(
                  border: OutlineInputBorder(),
                  labelText: 'Enter task title',
                  prefixIcon: Icon(Icons.add_task_sharp),
                ),
              ),
              const SizedBox(height: 15),
                              TextField(
                  controller: descriptioncontroller,
                  maxLines: null,
```

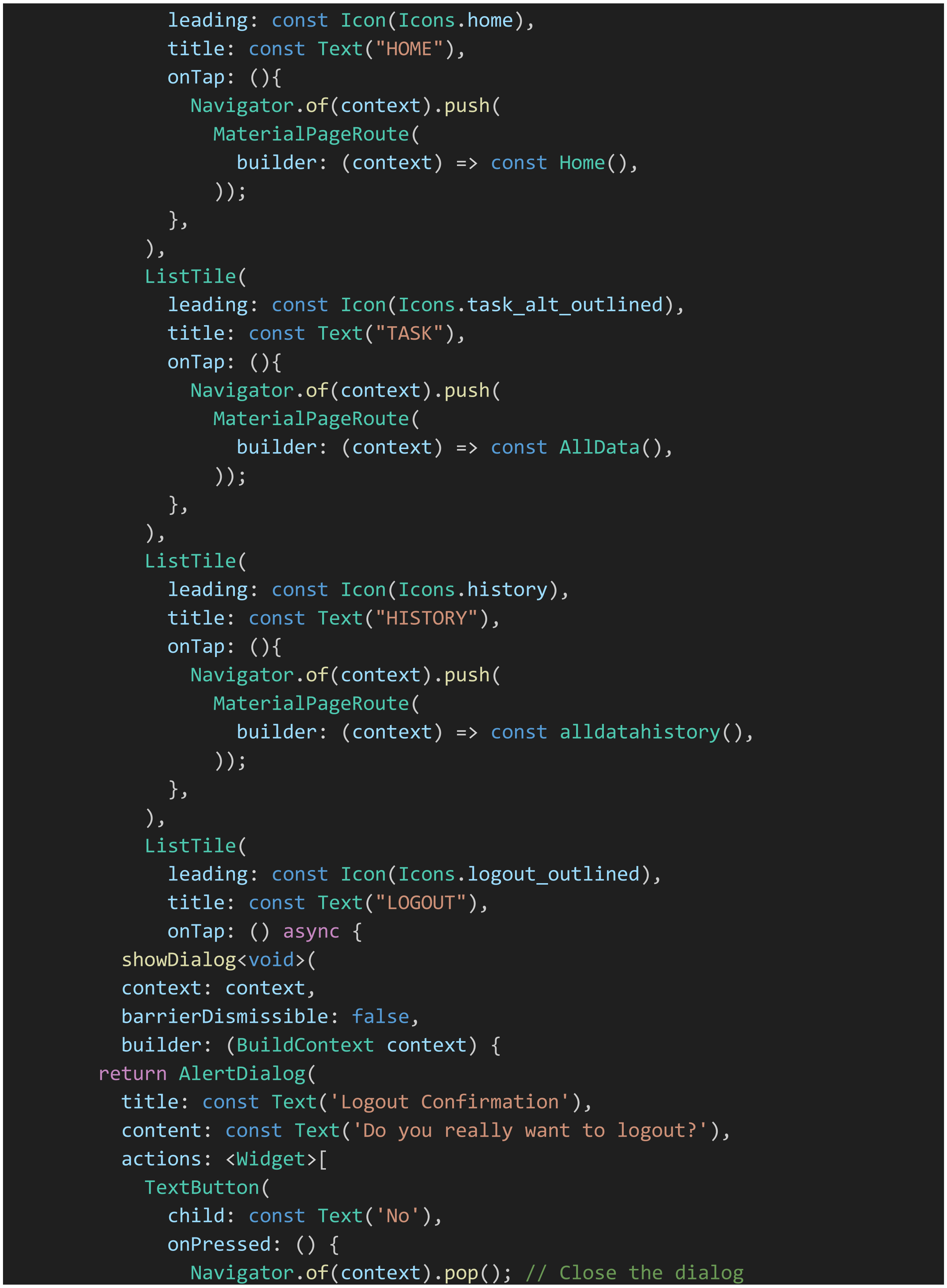


```dart
                  decoration: const InputDecoration(
                    border: OutlineInputBorder(),
                    labelText: 'Enter task description',
                    prefixIcon: Icon(Icons.description),
                  ),
                ),
              const SizedBox(height: 15),
              ElevatedButton(
                style: ElevatedButton.styleFrom(
                  backgroundColor: const Color.fromARGB(255, 108, 183, 245),
                  minimumSize: const Size.fromHeight(50),
                ),
                onPressed: () {
                  createTask();
                },
                child: const Text('SAVE',
                style: TextStyle(
                  color: Colors.black),),
              ),
              const SizedBox(height: 15),
              (isError)
                  ? Text(
                      errormessage,
                      style: errortxtstyle,
                    )
                  : Container(),
            ],
          ),
        ),
      ),
    ),
  );
}

var errortxtstyle = const TextStyle(
  fontWeight: FontWeight.bold,
  color: Colors.red,
  letterSpacing: 1,
  fontSize: 18,
);
var txtstyle = const TextStyle(
  fontWeight: FontWeight.bold,
  letterSpacing: 2,
  fontSize: 38,
);
```

```
}
```

```dart
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:task_manager_app/task.dart';

class UpdateData extends StatefulWidget {
  const UpdateData({
    super.key,
    required this.task,});

    final Task task;

  @override
  State<UpdateData> createState() => _UpdateDataState();
}


class _UpdateDataState extends State<UpdateData> {
  TextEditingController titlecontroller = TextEditingController();
  TextEditingController descriptioncontroller = TextEditingController();
  late String errormessage;
  late bool isError;

  @override
  void initState() {
    errormessage = "This is an error";
    isError = false;
    titlecontroller.text = widget.task.title;
    descriptioncontroller.text = widget.task.description;
    super.initState();
  }

  @override
  void dispose() {
    super.dispose();
  }

  Future updateTask(String id) async{
  final docTask = FirebaseFirestore.instance.collection('Task').doc(id);
  docTask.update({
    'description': descriptioncontroller.text,
    'title': titlecontroller.text,
  });

    Navigator.pop(context);
}
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: const Color.fromARGB(255, 182, 241, 252),
      appBar: AppBar(
        backgroundColor: const Color.fromARGB(255, 11, 79, 134),
        title: const Text('',
        style: TextStyle(color: Colors.white),),
        leading: IconButton(
          color: Colors.white,
          onPressed: (){
            Navigator.pop(context);
          },
          icon: const Icon(Icons.arrow_back),
          ),
      ),
      body: Container(
        decoration: const BoxDecoration(
        image: DecorationImage(
          image: AssetImage('assets/images/finalbg.jpg'),
          fit: BoxFit.cover),),
        child: Center(
          child: Container(
            margin: const EdgeInsets.symmetric(horizontal: 25),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(
                  'UPDATE TASK',
                  style: txtstyle,
                ),
                const SizedBox(height: 15),
                TextField(
                  controller: titlecontroller,
                  decoration: const InputDecoration(
                    border: OutlineInputBorder(),
                    labelText: 'Enter title',
                    prefixIcon: Icon(Icons.person),
                  ),
                ),
                const SizedBox(height: 15),
                TextField(
                  controller: descriptioncontroller,
                   maxLines: null,
                  decoration: const InputDecoration(
                    border: OutlineInputBorder(),
                    labelText: 'Enter description',
                    prefixIcon: Icon(Icons.email),
                  ),
                ),
                const SizedBox(height: 15),
                ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    backgroundColor: const Color.fromARGB(255, 108, 183, 245),
                    minimumSize: const Size.fromHeight(50),
                  ),
                  onPressed: () {
                    updateTask(widget.task.id);
                  },
                  child: const Text('UPDATE',
                  style: TextStyle(color: Colors.black),),
                ),
                const SizedBox(height: 15),
                (isError)
                    ? Text(
                        errormessage,
                        style: errortxtstyle,
                      )
                    : Container(),
              ],
            ),
          ),
        ),
      ),
    );
  }

  var errortxtstyle = const TextStyle(
    fontWeight: FontWeight.bold,
    color: Colors.red,
    letterSpacing: 1,
    fontSize: 18,
  );
  var txtstyle = const TextStyle(
    fontWeight: FontWeight.bold,
    letterSpacing: 2,
    fontSize: 38,
  );

}
```

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:task_manager_app/alldata.dart';
import 'package:task_manager_app/alldatahistory.dart';
import 'package:task_manager_app/home.dart';
import 'package:task_manager_app/login.dart';

class sidebar extends StatelessWidget {
   sidebar({super.key});

final user = FirebaseAuth.instance.currentUser!;
  @override
  Widget build(BuildContext context) {
    return Drawer(
      child: Container(
        decoration: const BoxDecoration(
        image: DecorationImage(
          image: AssetImage('assets/images/finalbg.jpg'),
          fit: BoxFit.cover),),
        child: ListView(
          children: [
            UserAccountsDrawerHeader(accountName:  const
            Text("Welcome!"),
            accountEmail: Text( user.email!,),
                currentAccountPicture: CircleAvatar(
                  child: ClipOval(
                    child:
Image.network('https://bloximages.chicago2.vip.townnews.com/tucson.com/content/tn
cms/assets/v3/editorial/0/80/0802223e-21bb-11ee-8d0c-
b3a2d439312e/64b0727559628.image.jpg?resize=1200%2C800',
                      width: 90,
                      height: 90,
                      fit: BoxFit.cover,
                    ),
                  ),
                ),
                ),
            ListTile(
```



```dart
              leading: const Icon(Icons.home),
              title: const Text("HOME"),
              onTap: (){
                Navigator.of(context).push(
                  MaterialPageRoute(
                    builder: (context) => const Home(),
                  ));
              },
            ),
            ListTile(
              leading: const Icon(Icons.task_alt_outlined),
              title: const Text("TASK"),
              onTap: (){
                Navigator.of(context).push(
                  MaterialPageRoute(
                    builder: (context) => const AllData(),
                  ));
              },
            ),
            ListTile(
              leading: const Icon(Icons.history),
              title: const Text("HISTORY"),
              onTap: (){
                Navigator.of(context).push(
                  MaterialPageRoute(
                    builder: (context) => const alldatahistory(),
                  ));
              },
            ),
            ListTile(
              leading: const Icon(Icons.logout_outlined),
              title: const Text("LOGOUT"),
              onTap: () async {
          showDialog<void>(
          context: context,
          barrierDismissible: false,
          builder: (BuildContext context) {
        return AlertDialog(
          title: const Text('Logout Confirmation'),
          content: const Text('Do you really want to logout?'),
          actions: <Widget>[
            TextButton(
              child: const Text('No'),
              onPressed: () {
                Navigator.of(context).pop(); // Close the dialog
              },
            ),
            TextButton(
              child: const Text('Yes'),
              onPressed: () async {
                Navigator.of(context).pop(); // Close the dialog
                await handleSignOut(context);
                await _showLogoutSuccessDialog(context);
                try {
                  Navigator.of(context).pushReplacement(
                    MaterialPageRoute(
                      builder: (context) => Login(),
                    ),
                  );
                } catch (e) {
                  print('Error signing out: $e');
                }
              },
            ),
          ],
        );
          },
        );
              },
            ),
          ],
        ),
      ),
    );
  }
}



Future<void> handleSignOut(BuildContext context) async {
  try {
    await FirebaseAuth.instance.signOut(); // Sign the user out
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => const Login()), // Navigate back to
the Login screen
    );
  } catch (e) {
    print('Error signing out: $e');
  }
```

```
}
Future<void> _showLogoutConfirmationDialog(BuildContext context) async {
  showDialog<void>(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return AlertDialog(
        title: const Text('Logout Confirmation'),
        content: const Text('Do you really want to logout?'),
        actions: <Widget>[
          TextButton(
            child: const Text('No'),
            onPressed: () {
              Navigator.of(context).pop(); // Close the dialog
            },
          ),
          TextButton(
            child: const Text('Yes'),
            onPressed: () async {
              Navigator.of(context).pop(); // Close the dialog
              await handleSignOut(context);
              await _showLogoutSuccessDialog(context);
              try {
                Navigator.of(context).pushReplacement(
                  MaterialPageRoute(
                    builder: (context) => Login(),
                  ),
                );
              } catch (e) {
                print('Error signing out: $e');
              }
            },
          ),
        ],
      );
    },
  );
}

  Future<void> _showLogoutSuccessDialog(BuildContext context) async {
    return showDialog<void>(
      context: context,
      barrierDismissible: false,
      builder: (BuildContext context) {
        return AlertDialog(
```

```dart
          title: const Text('Logout Successful'),
          content: const Text('You have been successfully logged out.'),
          actions: <Widget>[
            TextButton(
              child: const Text('OK'),
              onPressed: () {
                Navigator.of(context).pop(); // Close the dialog
              },
            ),
          ],
        );
      },
    );
  }


import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:task_manager_app/alldata.dart';
import 'package:task_manager_app/login.dart';
import 'package:task_manager_app/sidebar.dart';
import 'package:task_manager_app/task.dart';

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

  @override
  State<alldatahistory> createState() => _alldatahistoryState();
}
Future<void> _showTaskDetailsDialog(BuildContext context, Task task) async {
  return showDialog<void>(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text(task.title),
        content: SingleChildScrollView(
          child: ListBody(
            children: <Widget>[
              Text('Description:\n${task.description}'),
            ],
          ),
        ),
        actions: <Widget>[
          TextButton(
            child: Text('Close'),
            onPressed: () {
              Navigator.of(context).pop();
            },
          ),
        ],
      );
    },
  );
}
Future<void> handleSignOut(BuildContext context) async {
  try {
    await FirebaseAuth.instance.signOut();
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => const Login()),
    );
  } catch (e) {
    print('Error signing out: $e');
  }
}

Future<void> deleteAllTasks() async {
  try {
    // Fetch all tasks from the collection
    QuerySnapshot querySnapshot = await
FirebaseFirestore.instance.collection('Task').get();

    // Iterate through each task and delete it
    for (QueryDocumentSnapshot documentSnapshot in querySnapshot.docs) {
      await
FirebaseFirestore.instance.collection('Task').doc(documentSnapshot.id).delete();
    }

    print('All tasks deleted successfully.');
  } catch (e) {
    print('Error deleting tasks: $e');
  }
}

Future<void> _showDeleteAllConfirmationDialog(BuildContext context) async {
  showDialog<void>(
    context: context,
    barrierDismissible: false,
```

```dart
    builder: (BuildContext context) {
      return AlertDialog(
        title: const Text('Delete All Task History Confirmation'),
        content: const Text('Are you sure you want to delete all tasks
history?'),
        actions: <Widget>[
          TextButton(
            child: const Text('Cancel'),
            onPressed: () {
              Navigator.of(context).pop(); // Close the dialog
            },
          ),
          TextButton(
            child: const Text('Delete All'),
            onPressed: () async {
              await deleteAllTasks();  // Call deleteAllTasks to delete all tasks
              Navigator.of(context).pop(); // Close the dialog
            },
          ),
        ],
      );
    },
  );
}


Future<void> _showdeleteConfirmationDialog(BuildContext context, String id) async
{
  showDialog<void>(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return AlertDialog(
        title: const Text('Delete Confirmation'),
        content: const Text('Delete file?'),
        actions: <Widget>[
          TextButton(
            child: const Text('No'),
            onPressed: () {
              Navigator.of(context).pop(); // Close the dialog
            },
          ),
          TextButton(
            child: const Text('Yes'),
            onPressed: () async {
              Navigator.of(context).pop();
              await deleteTask(id);  // Call deleteTask to delete the task
            },
          ),
        ],
      );
    },
  );
}

class _alldatahistoryState extends State<alldatahistory> {
  @override
  Widget buildList(Task task) => GestureDetector (
    child: Container(
      decoration: const BoxDecoration(border: Border(bottom: BorderSide())),
      child: ListTile(
        contentPadding: const EdgeInsets.fromLTRB(10, 10, 10, 10),
          leading: const Icon(Icons.task),
          title: Text(
          task.title,
          maxLines: 1,
          overflow: TextOverflow.ellipsis,
          style: const TextStyle(color: Colors.black, fontWeight:
FontWeight.bold),),
          subtitle: Text(task.description,
          maxLines: 1,
          overflow: TextOverflow.ellipsis,),
          dense: true,
          onTap: () {
            _showTaskDetailsDialog(context, task);
          },
          trailing: Row(
            mainAxisSize: MainAxisSize.min,
            children: [
            Checkbox(
                    value: task.status,
                    onChanged: (newValue) {
                    print("Checkbox onChanged: newValue=$newValue");
                    if (newValue != null) {
                      updateTaskStatus(task.id, newValue);
                      setState(() {
                        task.status = !newValue;
                      });
                    }
                  }
```

```
                ),
          IconButton(
              onPressed: () {
                _showdeleteConfirmationDialog(context, task.id);
              },
              icon: const Icon(Icons.delete_outlined),
              ),

        ],
        ),
        ),
  ),
);


@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: const Color.fromARGB(255, 182, 241, 252),
    drawer: sidebar(),
    appBar: AppBar(
      iconTheme: const IconThemeData(color: Colors.white),
      centerTitle: true,
      title: const Text('All Tasks History',
      style: TextStyle(color: Colors.white),),
      backgroundColor: const Color.fromARGB(255, 11, 79, 134),
      actions: <Widget> [
          IconButton(
            onPressed: (){
              _showDeleteAllConfirmationDialog(context);
            },
            icon: const Icon(Icons.delete),),
        ],

    ),
    body: Container(
      decoration: const BoxDecoration(
      image: DecorationImage(
        image: AssetImage('assets/images/finalbg.jpg'),
        fit: BoxFit.cover),),
      child: StreamBuilder<List<Task>>(
        stream: readTasks(),
        builder: (context, snapshot) {
          if (snapshot.hasError) {
            return Text('Something went wrong! ${snapshot.error}');
            } else if (snapshot.hasData) {
              final tasks = snapshot.data!;
              final completedTasks = tasks.where((task) => task.status).toList();

              return ListView(
                children: completedTasks.map(buildList).toList(),
              );
            } else {
              return const Center(
                child: CircularProgressIndicator(),
              );
            }
          },
        ),
      ),
      );
  }
}
```

```
class Employee {
  final String id;
  final String name;
  final String email;

  Employee({
    required this.id,
    required this.name,
    required this.email,
  });

  Map<String, dynamic> toJson() => {
  'id': id,
  'name': name,
  'email': email,
};


  static Employee fromJson(Map<String, dynamic> json) => Employee(
    id: json['id'],
    name: json['name'],
    email: json['email'],
    );

}
```

[redacted]