MCA581: Specialization Project

Email Assist

Shah Naishar Vikram

(1947258)

Under the guidance of

Prof. Nismon Rio

Specialization project report submitted in partial fulfillment of the requirements of V semester Master of computer Applications, CHRIST (Deemed to be University)

November 2021

CERTIFICATE

*This is to certify that the report titled **Email Assist** is a bona fide record of work done by **Manish Bharti (1947235)** of CHRIST (Deemed to be University), Bangalore, in partial fulfillment of the requirements of V Semester MCA (Computer Applications) during the year 2021.*

Head of the Department **Project Guide**

Valued-by:

Name :

1 . Register Number :

Examination Centre : CHRIST (Deemed to be University)

2 . Date of Exam :

ACKNOWLEDGEMENTS

First of all, we thank the lord almighty for his immense grace and blessings showered on me at every stage of this work. We are greatly indebted to our Head of the Department of Computer Science, Dr. Joy Paulose, CHRIST (Deemed to be University) for providing the opportunity to take up this project as part of my curriculum and we wholeheartedly thank Dr.Tulasi B, Coordinator, Department of computer science for being constant support and for helping us to do our best.

We are deeply indebted to our project guide Dr. Saravanan KN, for his assistance and valuable suggestions as a guide. He made this project a reality.

We express our sincere thanks to Dr. Nismon Rio R, and Dr.Ramesh Chandra Poonia lecturers of the Department of Computer Science, CHRIST (Deemed to be University), for their valuable suggestions during the course of this project. Their critical suggestions helped us to improve the project work.

Acknowledging the efforts of everyone, their chivalrous help in the course of the project preparation and their willingness to corroborate with the work, their magnanimity through lucid technical details lead to the successful completion of my project.

We would like to express our sincere thanks to all our friends, colleagues, parents and all those who have directly or indirectly assisted during this work.

ABSTRACT

In this digital age, we all know how emails are essential for communication. But we all face email spamming. Our inbox is full of unwanted emails which we never want to read. Promotional emails, e-commerce mails are the most sent emails. Filtering essential mails from the pile of all mails has been a tedious and annoying task. Sometimes our important emails get lost in the pile of unwanted mail. To tackle this problem, we plan to build a software solution that will filter and segregate emails for us.

Email Assist is a cross-platform application that aims to assist users in segregating mail based on its category like event, lecture, trips, etc., and other essential features considering this current pandemic time where every meeting is online. The use case of this project is far-reaching as we know that we get hundreds of emails every day, and this will help them segregate emails with different parameters, starting with a few and then improving based on user feedback and suggestions as we grow.

Some of the highlighted features of the app will be segregating emails into events, meetings-links, assignments, registration, travel schedule, filtering based on(organization, deadline, etc). Some of the extra features are to star any message, swipe irrelevant emails to delete, take input from the user and update the filtering, deadline-based notification, unsubscribe button.

Table of Contents

LIST OF FIGURES

1. INTRODUCTION

1.1 Purpose

In this digital age, we all know how emails are essential for communication. But we all face email spamming. Our inbox is full of unwanted emails which we never want to read. Promotional emails, e-commerce mails are the most sent emails. Filtering essential mails from the pile of all mails has been a tedious and annoying task. Sometimes our important emails get lost in the pile of unwanted mail. To tackle this problem, we plan to build a software solution that will filter and segregate emails for us.

1.2 Scope

Email Assist will be a cross-platform app that will help solve the issues stated above. It will be used by all kinds of people who use email for various purposes. This tool will be of great importance for saving time in reading emails to people having busy schedules.

1.3 Definitions, Acronyms, and Abbreviations

- Flutter-Flutter is an open-source mobile SDK developer can use to build native-looking Android and iOS applications from the same code base. Flutter has been around since 2015 when Google introduced it and remained in the beta stage before its official launch in December 2018. Since then, the buzz around Flutter has been growing stronger.
  Flutter is now the top 11 software repos based on GitHub stars. Moreover, we've already seen thousands of Flutter apps being published on app stores. One of the most notable examples is the Xianyu app created by the Alibaba team, used by over 50 million people.

- Dart- Dart is a client-optimized language for developing fast apps on any platform. Its goal is to offer the most productive programming language for multi-platform development, paired with a flexible execution runtime platform for app frameworks.
  Languages are defined by their technical envelope — the choices made during development that shape the capabilities and strengths of a language. Dart is designed for a technical

envelope that is particularly suited to client development, prioritizing both development (sub-second stateful hot reload) and high-quality production experiences across a wide variety of compilation targets (web, mobile, and desktop).

- Firebase- Firebase is Google's mobile application development platform that helps you build, improve, and grow your app. Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure.
  Firebase is categorized as a NoSQL database program, which stores data in JSON-like documents.

1.4 Overview

Email Assist is a cross-platform application that aims to assist users in segregating mail based on its category like event, lecture, trips, etc., and other essential features considering this current pandemic time where every meeting is online. The use case of this project is far-reaching as we know that we get hundreds of emails every day, and this will help them segregate emails with different parameters, starting with a few and then improving based on user feedback and suggestions as we grow.

Some of the highlighted features of the app will be segregating emails into events, meetings-links, assignments, registration, travel schedule, filtering based on(organization, deadline, etc). Some of the extra features are star any message, swipe irrelevant emails to delete, take input from the user and update the filtering, deadline-based notification, unsubscribe button.

2. THE OVERALL DESCRIPTION

Mail Assist helps the users to segregate his/her emails into several categories automatically which will, in turn, help users save time. Instead of manually putting emails into different category buckets, the proposed system will group those mails automatically and will show the users only relevant and important information. In this way, this system will work as an abstraction layer for users' emails and will hide unnecessary details.

2.1 System Interfaces

- The system consists of and interacts with the following hardware components and software.
- The system is made of the following hardware components and the following software platforms are used to create the project.
- Mail Assist home page, where the user can see and take action on segregated mails.
- Login page for authentication purposes. It also provides the app with all provisions to access the users' mails.
- Once login, the user will be able to use all the functionalities provided to the user.

2.2 Hardware Interfaces

- Processor: Intel Core i3 8th gen and up or AMD Ryzen 3 3100 and up.
- Processor speed: 1.5 to 3.40 GHz
- RAM: 4 GB and more

2.3 Software Interfaces

- Frontend: Flutter(SDK).
- Language used: Dart.
- Backend: Firebase.
- Code Editor: Any editor is fine, but preferably VS Code.
- Browser: Chrome or Firefox or Internet Explorer or Microsoft Edge

2.4 Communications Interfaces

There are no specific communication interface requirements. Existing OS and network infrastructure will be leveraged for communication.

2.5 Memory Constraints

Memory constraints will come into play when the size of the database grows to a considerable size.

2.6 Operations

The product shall have operations to protect the database from being corrupted or accidentally altered during a system failure.

2.7 Product Functions

The cross-platform application will mainly segregate users' mails into various categories automatically. Other functionalities will be filtering emails based on travel schedule, based on deadline, based on the organization. Sort emails based on priority.

Few more functionality includes,

- Star any message
- Swipe irrelevant emails to delete
- Take input from the user and update the filtering
- Deadline based notification
- Unsubscribe button

2.8 User Characteristics

There is a single type of user in this system. Whoever wants to use this system, that person needs to log in and after successful authentication, the user will have all the functionality available in the app.

2.9 Constraints

- Using this system is fairly simple and intuitive. A user familiar with basic browser navigation skills should be able to understand all functionality provided by the system.
- The system should work on most home desktop and laptop computers which support JavaScript and HTML5
- The system will be intended to run on Firefox 4 and above, Google Chrome 10 and above and Internet Explorer 8 and above.
- The system is limited by its operating server in terms of the maximum number of users it can support at a given time.

3. SPECIFIC REQUIREMENTS

In this digital age, we all know how emails are important for communication. But we all face email spamming. Our inbox is full of unwanted emails which we never want to read. Promotional emails, e-commerce mails are the most sent emails. Filtering important mails from the pile of all mails has been a tedious and annoying task. Sometimes our important mail gets lost in the pile of unwanted mail. To tackle this problem, we are planning to build a software solution that will filter and segregate emails for us. It will be used by all kinds of people who use email for various purposes. This tool will be of great importance for saving time in reading emails to the people having busy schedules.

3.1 External Interfaces

External interfaces used in this project are different APIs such as GMail Api, ML Kit by google, etc. to provide different functionalities offered by the app.

3.2 Functions

- Categorize based on different criteria like:
    - Meet links
    - Trip schedule
    - Upcoming E-commerce Deliveries etc
- Filtering based on the deadline
- Ease of work
- Prioritize unsubscribe links

3.3 Performance Requirements

The performance of environmental control will be always proportional to the network speed available and the throughput will be at maximum if the network speed will be at megabits per second.

For experiencing the better performance of the system we need to have a good internet facility and uninterrupted resource for accessing the application.

3.4 Logical Database Requirements

Database will be used to generate reports. Initially data such as inputs for the prediction and excel formatted data is taken and stored into the database then reports will be generated accordingly along with the predictions and analysis.

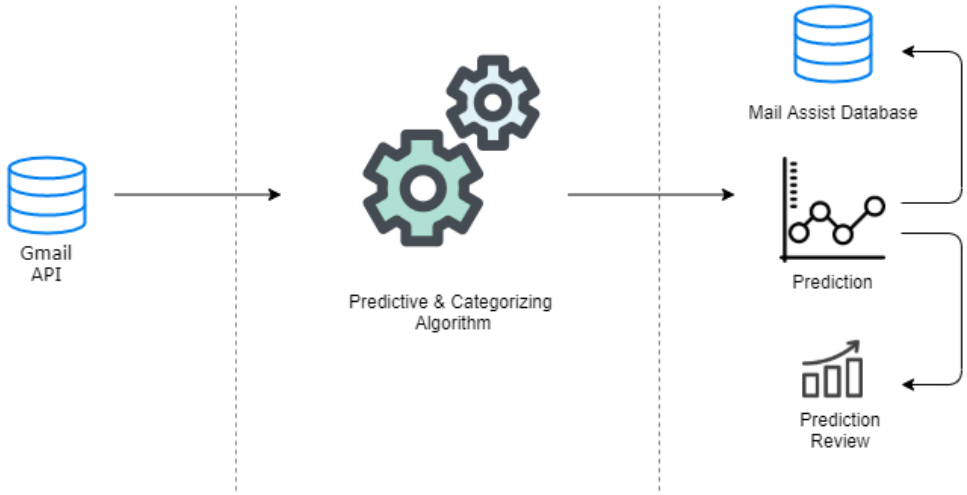

Fig.3.4.1 Logical database

3.5 Design Constraints

The project is going to use reusable controls of Material.IO. System Constraints restrict options of design, behavior, appearance, or operation. They become requirements due to factors outside the normal problem domain. System Constraints describe how the product operates inside various circumstances and limit the options designers have if building the product. This section specifies design constraints imposed by other standards, hardware limitations, communication interface limitations, etc. There are a number of attributes of software that can serve as requirements.

3.5.1 Standards Compliance

- System should have enough memory and processor.
- All the devices and systems should be very reliable.
- All the tools should be compatible with the embedded system on the device.

3.6 Software System Attributes

3.6.1 Reliability

This application is simple and reliable to use. Users have to sign in through their mail account and then every time a new mail will come, if it's relevant to the features of the application, it will categorize it and display it in the corresponding category.

3.6.2 Availability

The system has a high rate of acceptance if there is a power source and a steady internet connection everything will work smoothly.

3.6.3 Security

Use of hash function can also increase security greatly. The application will also be secured with authentication.

3.6.4 Maintainability

Best performance and precision and so the initial establishment cost will be high. But the maintenance cost will be less once setup, afterwards maintenance cost is less.

3.6.5 Portability

The project can be easily implemented in any other operating system as long as the system requirements are met. The application can be accessed from mobile or tablet also; can easily be established and the operations can easily be implemented.

3.7 Organizing the Specific Requirements

3.7.1 System Mode

The systems can be used by different kinds of users, therefore the systems behave differently depending upon the different mail for different users and it provides different categories.

4. CHANGE MANAGEMENT PROCESS

The Change Management Process is the mechanism used to initiate, record, assess, approve and resolve project changes. Project changes are needed when it is deemed necessary to change the scope, time or cost of one or more previously approved project deliverables. People generally dislike change. It disrupts the familiar and the comfortable. Change forces you to rethink what were previously routine actions that required no new thought. It requires familiarizing yourself with new processes and procedures. In many cases, change requires the acquisition of new skills. It is, however, inevitable. Some people may choose to resist a particular transformation, but resistance to change isn't usually successful. In most cases, change is unavoidable because the world is dynamic. Since most change occurs to improve a process, a product, or an outcome, it is critical to identify the focus and to clarify goals. Providing clear and open lines of communication throughout the process is a critical element in all change modalities. The methods advocate transparency and two-way communication structures that provide avenues to vent frustrations, applaud what is working, and seamlessly change what doesn't work. After having a clear communication with the customer through phone calls or by mail, planning should be done. As part of the planning process, resource identification and funding are crucial elements. These can include infrastructure, equipment, and software systems. Resistance is a very normal part of change management, but it can threaten the success of a project. Most resistance occurs due to a fear of the unknown. It also occurs because there is a fair amount of risk associated with change – the risk of impacting dependencies, return on investment risks, and risks associated with allocating budget to something new. Anticipating and preparing for resistance by arming leadership with tools to manage it will aid in a smooth change lifecycle. When managing a change through its lifecycle, it's important to recognize the success of teams and individuals involved. This will help in the adoption of both change management processes as well as adoption of the change itself. Once the change is made, revise and then review it from the customer. And the final project is delivered to the customer as per his/her requirements.

5. DESIGN SPECIFICATION

5.1 ARCHITECTURAL DESIGN

The 3-tier schema is an extension of the 2-tier architecture. the 3-tier architecture has the following layers ; Presentation layer (your PC, Tablet, Mobile, etc.),Application layer (server) and Database Server. This DBMS architecture contains an Application layer between the user and the DBMS, which is responsible for communicating the user's request to the DBMS system and send the response from the DBMS to the user.The application layer (business logic layer) also processes functional logic, constraint, and rules before passing data to the user or down to the DBMS.Three tier architecture is the most popular DBMS architecture. Fig. 5.1.2 Architecture Diagram depicts the architectural design of ASA sales forecasting which contains all the steps.

Fig. 5.1 3-tier Architecture Diagram

5.2 DATA FLOW DIAGRAM

The context diagram also called the Level 1 DFD depicts the overall data flow in the system.

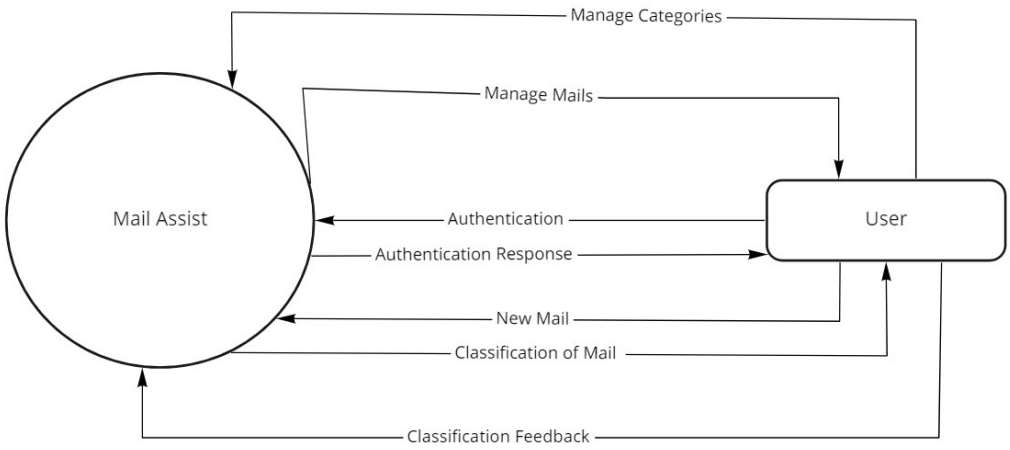

Fig. 5.2 level 0 DFD

5.3 ENTITY RELATIONSHIP DIAGRAM

The entity relationship diagram describes interrelated things in the project. A basic ER model is composed of entity types and specifies relationships that can exist between instances of those entity types. An entity relationship diagram shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

The entity relationship diagram (Fig. 5.3 ER Diagram), represents the inter relativity of the different project components and how it is related together. the different modules of the project. the entity relationship diagram helps in the design and development of the database.

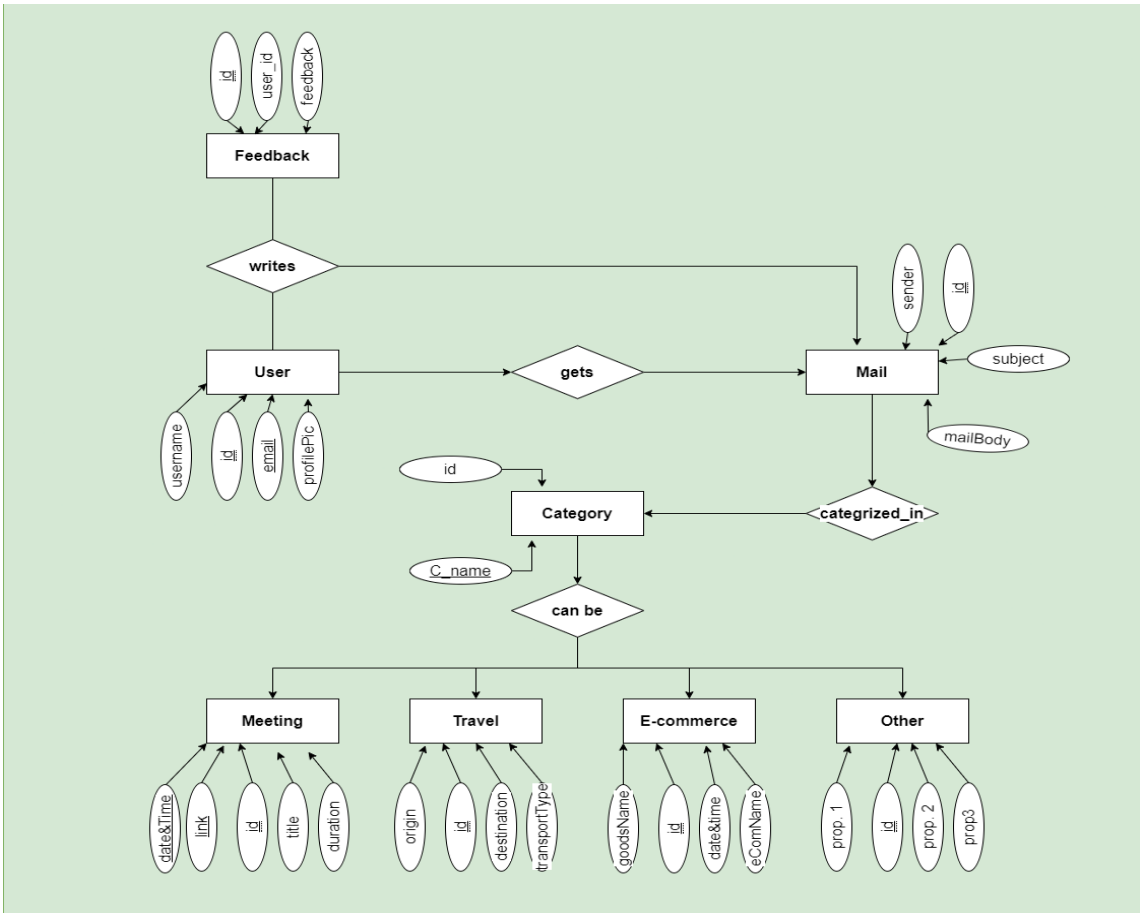

Fig. 5.3 ER Diagram

5.4 CLASS DIAGRAM

Database design illustrates a detailed data model of a database also known as the database schema. It shows the various tables that are in the system and the relationships between those tables.

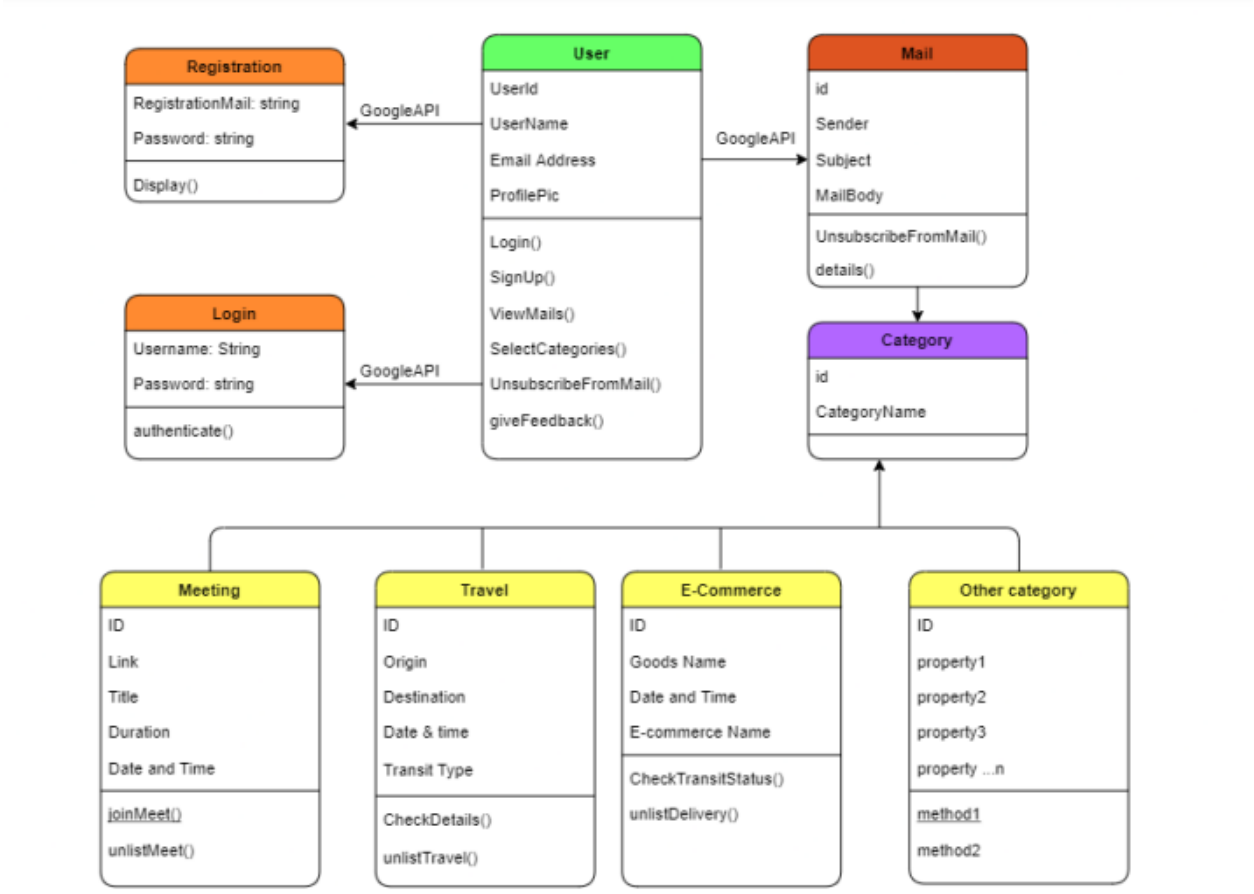

Fig.5.4 Class Diagram

5.5 State Diagram

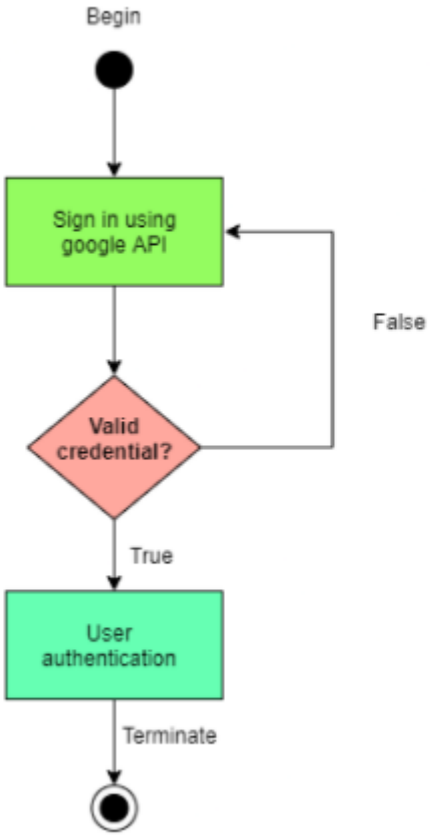

Fig 5.5.4 csv datafile

5.6 Use Case Diagram

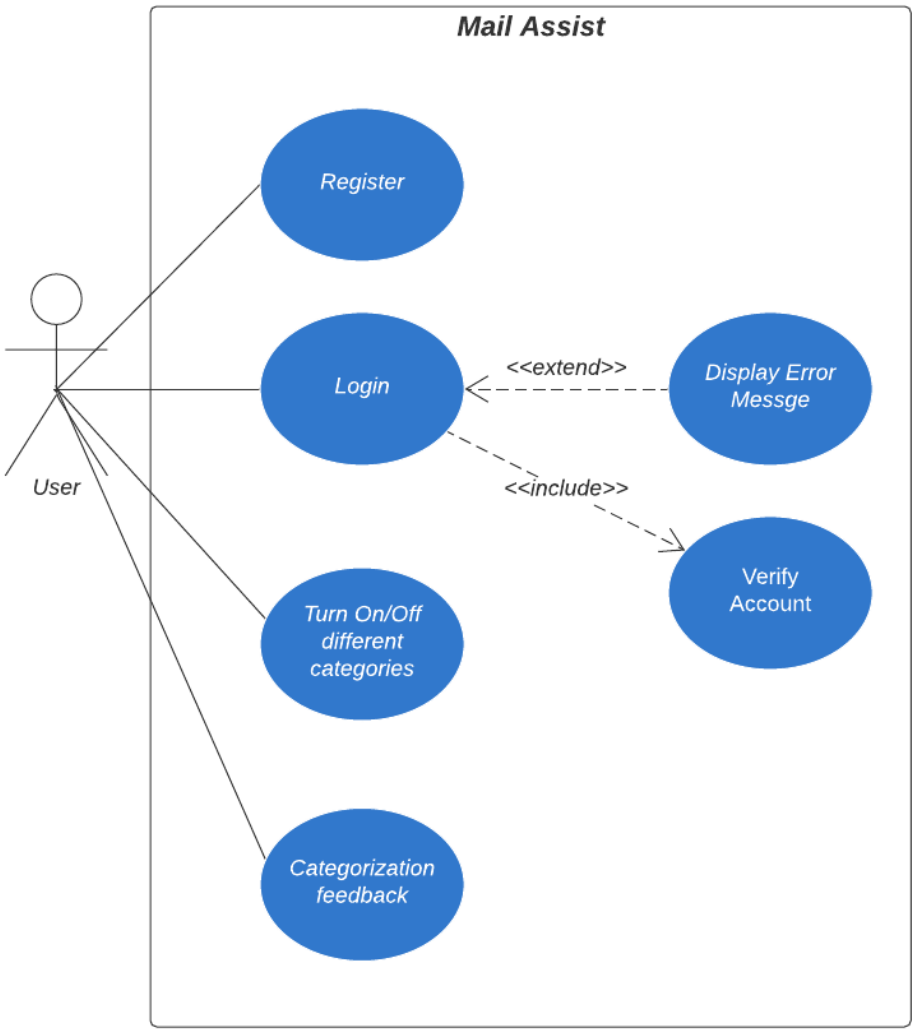

Fig 5.6 csv datafile

6. IMPLEMENTATION DETAILS

6.1 Code

6.1.1 main.dart

```
import 'dart:async';

import 'dart:io';

import 'package:collection/collection.dart' show IterableExtension;

import 'package:enough_mail_app/routes.dart';

import 'package:enough_mail_app/screens/all_screens.dart';

import 'package:enough_mail_app/services/app_service.dart';

import 'package:enough_mail_app/services/background_service.dart';

import 'package:enough_mail_app/services/biometrics_service.dart';

import 'package:enough_mail_app/services/i18n_service.dart';

import 'package:enough_mail_app/services/key_service.dart';

import 'package:enough_mail_app/services/mail_service.dart';

import 'package:enough_mail_app/services/navigation_service.dart';

import 'package:enough_mail_app/services/notification_service.dart';

import 'package:enough_mail_app/services/scaffold_messenger_service.dart';

import 'package:enough_mail_app/services/settings_service.dart';

import 'package:enough_mail_app/services/theme_service.dart';

import 'package:enough_mail_app/widgets/inherited_widgets.dart';
```

```
import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:enough_platform_widgets/enough_platform_widgets.dart';

import 'locator.dart';

import 'package:flutter_gen/gen_l10n/app_localizations.dart';

// AppStyles appStyles = AppStyles.instance;


void main() {

  setupLocator();

  runApp(MyApp());

}


class MyApp extends StatefulWidget {

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


  @override

  _MyAppState createState() => _MyAppState();

}


class _MyAppState extends State<MyApp> with WidgetsBindingObserver {
```

```
late Future<MailService> _appInitialization;
ThemeMode _themeMode = ThemeMode.system;
ThemeService? _themeService;
Locale? _locale;
bool _isInitialized = false;

@override
void initState() {
  WidgetsBinding.instance!.addObserver(this);
  _appInitialization = _initApp();
  super.initState();
}

@override
void dispose() {
  WidgetsBinding.instance!.removeObserver(this);
  super.dispose();
}

@override
```

```
 void didChangeAppLifecycleState(AppLifecycleState state) {
  if (_isInitialized) {
    locator<AppService>().didChangeAppLifecycleState(state);
  }
 }

 Future<MailService> _initApp() async {
  final settings = await locator<SettingsService>().init();
  final themeService = locator<ThemeService>();
  _themeService = themeService;
  themeService.addListener(() => setState(() {
        _themeMode = themeService.themeMode;
      }));
  themeService.init(settings);
  final i18nService = locator<I18nService>();
  final languageTag = settings.languageTag;
  if (languageTag != null) {
    final settingsLocale = AppLocalizations.supportedLocales
        .firstWhereOrNull((l) => l.toLanguageTag() == languageTag);
    if (settingsLocale != null) {
```

```
    final settingsLocalizations =
        await AppLocalizations.delegate.load(settingsLocale);
    i18nService.init(settingsLocalizations, settingsLocale);
    setState(() {
      _locale = settingsLocale;
    });
  }
}
final mailService = locator<MailService>();
// key service is required before mail service due to Oauth configs
await locator<KeyService>().init();
await mailService.init(i18nService.localizations);

if (mailService.messageSource != null) {
  final state = MailServiceWidget.of(context);
  if (state != null) {
    state.account = mailService.currentAccount;
    state.accounts = mailService.accounts;
  }
  // on ios show the app drawer:
```

```
if (Platform.isIOS) {
  locator<NavigationService>().push(Routes.appDrawer, replace: true);
}

/// the app has at least one configured account
locator<NavigationService>().push(Routes.messageSource,
    arguments: mailService.messageSource,
    fade: true,
    replace: !Platform.isIOS);
// check for a tapped notification that started the app:
final notificationInitResult =
    await locator<NotificationService>().init();
if (notificationInitResult !=
    NotificationServiceInitResult.appLaunchedByNotification) {
  // the app has not been launched by a notification
  await locator<AppService>().checkForShare();
}
if (settings.enableBiometricLock) {
  locator<NavigationService>().push(Routes.lockScreen);
  final didAuthenticate =
```

```
          await locator<BiometricsService>().authenticate();
      if (didAuthenticate) {
        locator<NavigationService>().pop();
      }
    }
  } else {
    // this app has no mail accounts yet, so switch to welcome screen:
    locator<NavigationService>()
        .push(Routes.welcome, fade: true, replace: true);
  }
  await locator<BackgroundService>().init();
  _isInitialized = true;
  return mailService;
}

@override
Widget build(BuildContext context) {
  return PlatformSnackApp(
    supportedLocales: AppLocalizations.supportedLocales,
    localizationsDelegates: AppLocalizations.localizationsDelegates,
```

```
locale: _locale,
debugShowCheckedModeBanner: false,
title: 'MailAssist',
onGenerateRoute: AppRouter.generateRoute,
initialRoute: Routes.splash,
navigatorKey: locator<NavigationService>().navigatorKey,
scaffoldMessengerKey:
    locator<ScaffoldMessengerService>().scaffoldMessengerKey,
builder: (context, child) {
  locator<I18nService>().init(
      AppLocalizations.of(context)!, Localizations.localeOf(context));
  child ??= FutureBuilder<MailService>(
    future: _appInitialization,
    builder: (context, snapshot) {
      switch (snapshot.connectionState) {
        case ConnectionState.none:
        case ConnectionState.waiting:
        case ConnectionState.active:
          return SplashScreen();
        case ConnectionState.done:
```

```
            // in the meantime the app has navigated away
            break;
          }
          return Container();
        },
      );
      final mailService = locator<MailService>();
      return MailServiceWidget(
        child: child,
        account: mailService.currentAccount,
        accounts: mailService.accounts,
        messageSource: mailService.messageSource,
      );
    },
    // home: Builder(
    //   builder: (context) {
    //     locator<I18nService>().init(
    //         AppLocalizations.of(context)!, Localizations.localeOf(context));
    //     return FutureBuilder<MailService>(
    //       future: _appInitialization,
```

```
//       builder: (context, snapshot) {
//         switch (snapshot.connectionState) {
//           case ConnectionState.none:
//           case ConnectionState.waiting:
//           case ConnectionState.active:
//             return SplashScreen();
//           case ConnectionState.done:
//             // in the meantime the app has navigated away
//             break;
//         }
//         return Container();
//       },
//     );
//   },
// ),
materialTheme:
    _themeService?.lightTheme ?? ThemeService.defaultLightTheme,
materialDarkTheme:
    _themeService?.darkTheme ?? ThemeService.defaultDarkTheme,
materialThemeMode: _themeMode,
```

```
      cupertinoTheme: CupertinoThemeData(
        brightness: Brightness.light,
      ),
    );
  }
}
```

6.1.2. App_Drawer.dart

```
import 'dart:io';


import 'package:badges/badges.dart';
import 'package:enough_mail/enough_mail.dart';
import 'package:enough_mail_app/extensions/extension_action_tile.dart';
import 'package:enough_mail_app/locator.dart';
import 'package:enough_mail_app/models/account.dart';
import 'package:enough_mail_app/services/icon_service.dart';
import 'package:enough_mail_app/util/localized_dialog_helper.dart';
import 'package:enough_mail_app/services/mail_service.dart';
import 'package:enough_mail_app/services/navigation_service.dart';
import 'package:enough_mail_app/widgets/inherited_widgets.dart';
```

```
import 'package:enough_mail_app/widgets/mailbox_tree.dart';
import 'package:enough_platform_widgets/enough_platform_widgets.dart';
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';

import '../routes.dart';

class AppDrawer extends StatelessWidget {
  AppDrawer({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final mailService = locator<MailService>();
    final theme = Theme.of(context);
    final localizations = AppLocalizations.of(context)!;
    final iconService = locator<IconService>();
    final mailState = MailServiceWidget.of(context)!;
    final currentAccount = mailState.account ?? mailService.currentAccount;
    var accounts = mailState.accounts ?? mailService.accounts;
    if (mailService.hasUnifiedAccount) {
```

```
    accounts = accounts.toList();
    accounts.insert(0, mailService.unifiedAccount!);
  }
  return PlatformDrawer(
    child: SafeArea(
      child: Column(
        children: [
          Material(
            elevation: 18,
            child: Padding(
              padding: const EdgeInsets.all(8.0),
              child: _buildAccountHeader(
                  currentAccount, mailService.accounts, theme),
            ),
          ),
          Expanded(
            child: SingleChildScrollView(
              child: Material(
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
```

```
          children: [
            _buildAccountSelection(context, mailService, accounts,
                currentAccount, localizations),
            _buildFolderTree(currentAccount),
            ExtensionActionTile.buildSideMenuForAccount(
                context, currentAccount),
            Divider(),
            PlatformListTile(
              leading: Icon(iconService.about),
              title: Text(localizations.drawerEntryAbout),
              onTap: () {
                LocalizedDialogHelper.showAbout(context);
              },
            ),
          ],
        ),
      ),
    ),
  ),
Material(
```

```
            elevation: 18,
            // child: PlatformListTile(
            //   leading: Icon(iconSservice.settings),
            //   title: Text(localizations.drawerEntrySettings),
            //   onTap: () {
            //     final navService = locator<NavigationService>();
            //     navService.push(Routes.settings);
            //   },
            // ),
          )
        ],
      ),
    ),
  );
}


Widget _buildAccountHeader(
  Account? currentAccount,
  List<Account> accounts,
  ThemeData theme,
```

```
) {
  if (currentAccount == null) {
    return Container();
  }
  final avatarAccount =
      currentAccount.isVirtual ? accounts.first : currentAccount;
  final userName = currentAccount.userName;
  final accountName = Text(
    currentAccount.name,
    style: TextStyle(fontWeight: FontWeight.bold),
  );
  final accountNameWithBadge = locator<MailService>().hasError(currentAccount)
      ? Badge(child: accountName)
      : accountName;

  return PlatformListTile(
    onTap: () {
      final NavigationService navService = locator<NavigationService>();
      if (currentAccount is UnifiedAccount) {
        navService.push(Routes.settingsAccounts, fade: true);
```

```
    } else {
      navService.push(Routes.accountEdit,
          arguments: currentAccount, fade: true);
    }
  },
  title: Row(
    children: [
      CircleAvatar(
        backgroundColor: theme.secondaryHeaderColor,
        backgroundImage: NetworkImage(
          avatarAccount.imageUrlGravator!,
        ),
        radius: 30,
      ),
      Padding(
        padding: EdgeInsets.only(left: 8),
      ),
      Expanded(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
```

```
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            accountNameWithBadge,
            if (userName != null) ...{
              Text(
                userName,
                style: TextStyle(fontStyle: FontStyle.italic, fontSize: 14),
              ),
            },
            Text(
              currentAccount is UnifiedAccount
                  ? currentAccount.accounts.map((a) => a.name).join(', ')
                  : currentAccount.email,
              style: TextStyle(fontStyle: FontStyle.italic, fontSize: 14),
            ),
          ],
        ),
      ),
    ],
  ),
```

```
  );
}

Widget _buildAccountSelection(
    BuildContext context,
    MailService mailService,
    List<Account> accounts,
    Account? currentAccount,
    AppLocalizations localizations) {
  if (accounts.length > 1) {
    return ExpansionTile(
      leading: mailService.hasAccountsWithErrors() ? Badge() : null,
      title: Text(localizations
          .drawerAccountsSectionTitle(mailService.accounts.length)),
      children: [
        for (final account in accounts) ...{
          PlatformListTile(
            leading: mailService.hasError(account)
                ? Icon(Icons.error_outline)
                : null,
```

```
tileColor: mailService.hasError(account) ? Colors.red : null,
title: Text(account.name),
selected: account == currentAccount,
onTap: () async {
  final navService = locator<NavigationService>();
  if (!Platform.isIOS) {
    // close drawer
    navService.pop();
  }
  if (mailService.hasError(account)) {
    navService.push(Routes.accountEdit, arguments: account);
  } else {
    final accountWidgetState = MailServiceWidget.of(context);
    if (accountWidgetState != null) {
      accountWidgetState.account = account;
    }
    final messageSource = await locator<MailService>()
        .getMessageSourceFor(account, switchToAccount: true);
    navService.push(Routes.messageSource,
        arguments: messageSource,
```

```
                replace: !Platform.isIOS,
                fade: true);
            }
          },
          onLongPress: () {
            final navService = locator<NavigationService>();
            if (account is UnifiedAccount) {
              navService.push(Routes.settingsAccounts, fade: true);
            } else {
              navService.push(Routes.accountEdit,
                arguments: account, fade: true);
            }
          },
        ),
      },
      _buildAddAccountTile(localizations),
    ],
  );
} else {
  return _buildAddAccountTile(localizations);
```

```
  }
}

Widget _buildAddAccountTile(AppLocalizations localizations) {
  return PlatformListTile(
    leading: Icon(Icons.add),
    title: Text(localizations.drawerEntryAddAccount),
    onTap: () {
      final navService = locator<NavigationService>();
      if (!Platform.isIOS) {
        navService.pop();
      }
      navService.push(Routes.accountAdd);
    },
  );
}

Widget _buildFolderTree(Account? account) {
  if (account == null) {
    return Container();
```

```
    }
    return MailboxTree(account: account, onSelected: _navigateToMailbox);
  }

  void _navigateToMailbox(Mailbox mailbox) async {
    final mailService = locator<MailService>();
    final account = mailService.currentAccount!;
    final messageSource =
        await mailService.getMessageSourceFor(account, mailbox: mailbox);
    locator<NavigationService>().push(Routes.messageSource,
        arguments: messageSource, replace: !Platform.isIOS, fade: true);
  }
}
```

6.1.3. oauth.dart:

```
import 'package:enough_mail/enough_mail.dart';
import 'package:enough_mail_app/locator.dart';
import 'package:enough_mail_app/services/key_service.dart';
import 'package:enough_mail_app/util/http_helper.dart';
import 'package:flutter_web_auth/flutter_web_auth.dart';
```

```
class OauthClientId {
  final String id;
  final String? secret;

  const OauthClientId(this.id, this.secret);
}

abstract class OauthClient {
  final String incomingHostName;
  bool get isEnabled => (oauthClientId != null);
  OauthClientId? get oauthClientId =>
      locator<KeyService>().oauth[incomingHostName];
  OauthClient(this.incomingHostName);

  Future<OauthToken?> authenticate(String email) async {
    try {
      final token = await _authenticate(email);
      token.provider = incomingHostName;
      return token;
```

```
    } catch (e, s) {
      print('Unable to authenticate: $e $s');
      return Future.value();
    }
  }

  Future<OauthToken?> refresh(OauthToken token) async {
    try {
      final refreshedToken = await _refresh(token);
      refreshedToken.provider = incomingHostName;
      return refreshedToken;
    } catch (e, s) {
      print('Unable to refresh tokens: $e $s');
      return Future.value();
    }
  }

  Future<OauthToken> _authenticate(String email);
  Future<OauthToken> _refresh(OauthToken token);
}
```

```
class GmailOAuthClient extends OauthClient {
  GmailOAuthClient() : super('imap.gmail.com');

  @override
  Future<OauthToken> _authenticate(String email) async {
    final clientId = oauthClientId!.id;
    final callbackUrlScheme = clientId.split('.').reversed.join('.');

    // Construct the url
    final uri = Uri.https('accounts.google.com', '/o/oauth2/v2/auth', {
      'response_type': 'code',
      'client_id': clientId,
      'redirect_uri': '$callbackUrlScheme:/',
      'scope': 'https://mail.google.com/',
      'login_hint': email,
    });

    // Present the dialog to the user
    final result = await FlutterWebAuth.authenticate(
```

```
    url: uri.toString(), callbackUrlScheme: callbackUrlScheme);

// Extract code from resulting url
final code = Uri.parse(result).queryParameters['code'];

// Use this code to get an access token
final response = await HttpHelper.httpPost(
  'https://oauth2.googleapis.com/token',
  body: {
    'client_id': clientId,
    'redirect_uri': '$callbackUrlScheme:/',
    'grant_type': 'authorization_code',
    'code': code,
  },
);

// Get the access token from the response
final text = response.text;
if (response.statusCode != 200 || text == null) {
  throw new StateError(
```

```
                'Unable to get Google OAuth token with code $code, status code=${response.statusCode}, response=$text');
    }
    return OauthToken.fromText(text);
  }

  @override
  Future<OauthToken> _refresh(OauthToken token) async {
    final clientId = oauthClientId!.id;
    final callbackUrlScheme = clientId.split('.').reversed.join('.');
    final response =
        await HttpHelper.httpPost('https://oauth2.googleapis.com/token', body: {
      'client_id': clientId,
      'redirect_uri': '$callbackUrlScheme:/',
      'refresh_token': token.refreshToken,
      'grant_type': 'refresh_token',
    });
    final text = response.text;
    if (response.statusCode != 200 || text == null) {
      throw new StateError(
```

```
            'Unable to refresh Google OAuth token $token, status code=${response.statusCode},
response=$text');

    }

    return OauthToken.fromText(text);

  }

}



class OutlookOAuthClient extends OauthClient {

                                                                    //                                  source:
https://docs.microsoft.com/en-us/exchange/client-developer/legacy-protocols/how-to-authenti
cate-an-imap-pop-smtp-application-by-using-oauth

  static const String _scope =

      'https://outlook.office.com/IMAP.AccessAsUser.All https://outlook.office.com/SMTP.Send
offline_access';

  OutlookOAuthClient() : super('outlook.office365.com');



  @override

  Future<OauthToken> _authenticate(String email) async {

    final clientId = oauthClientId!.id;

    final clientSecret = oauthClientId!.secret;

    final callbackUrlScheme =

        //'https://login.microsoftonline.com/common/oauth2/nativeclient';
```

```
    'maily://oauth';

// Construct the url
final uri = Uri.https(
    'login.microsoftonline.com', '/common/oauth2/v2.0/authorize', {
  'response_type': 'code',
  'client_id': clientId,
  'client_secret': clientSecret,
  'redirect_uri': callbackUrlScheme,
  'scope': _scope,
  'login_hint': email,
});
// print('authenticate URL: $uri');

// Present the dialog to the user
final result = await FlutterWebAuth.authenticate(
    url: uri.toString(), callbackUrlScheme: 'maily'); // callbackUrlScheme);

// Extract code from resulting url
final code = Uri.parse(result).queryParameters['code'];
```

```
  // print('result: $result');
  // print('got code: "$code"');
  // Use this code to get an access token
  final response = await HttpHelper.httpPost(
      'https://login.microsoftonline.com/common/oauth2/v2.0/token',
      body: {
        'client_id': clientId,
        'redirect_uri': callbackUrlScheme,
        'grant_type': 'authorization_code',
        'code': code,
      });

  // Get the access token from the response
  // print('authorization code token:');
  // print(response.text);
  // print('response.code=${response.statusCode}');
  return OauthToken.fromText(response.text!);
}

@override
```

```
  Future<OauthToken> _refresh(OauthToken token) async {
    final clientId = oauthClientId!.id;
    final response = await HttpHelper.httpPost(
        'https://login.microsoftonline.com/common/oauth2/v2.0/token',
        body: {
          'client_id': clientId,
          'scope': _scope,
          'refresh_token': token.refreshToken,
          'grant_type': 'refresh_token',
        });
    final text = response.text;
    if (response.statusCode != 200 || text == null) {
      throw new StateError(
          'Unable to refresh Outlook OAuth token $token, status code=${response.statusCode},
response=$text');
    }
    return OauthToken.fromText(text);
  }
}
```

7.RESULTS AND DISCUSSION

7.1 USER INTERFACE DESIGN

In the user interface module, we included the screenshots of various interfaces in which different stakeholders can view and interact via that area. We provided the authorization to them, according to that stakeholders could interact with them. Along with the screenshot, the specific figure names indicate its unique functionality. User interface screenshots are attached below.

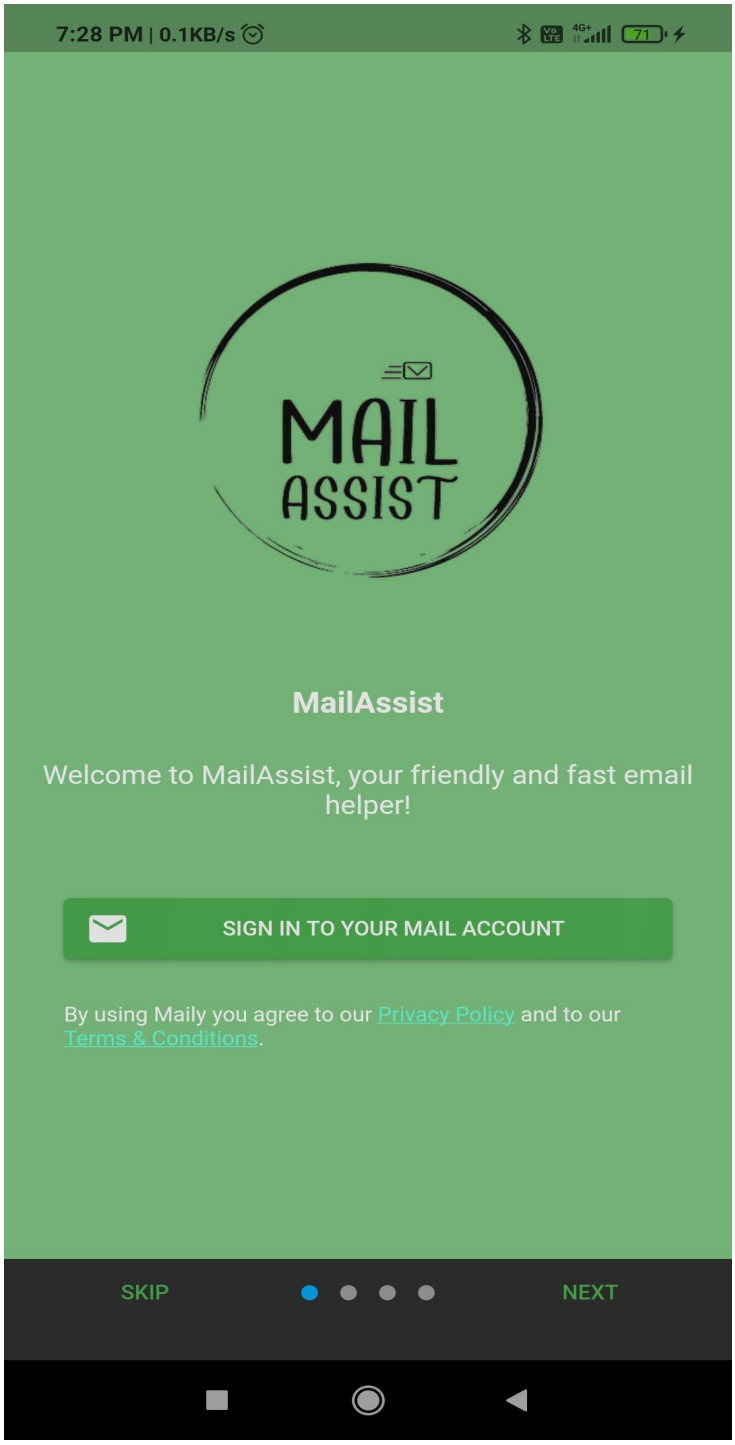

Fig. 7.1.1 splash screen

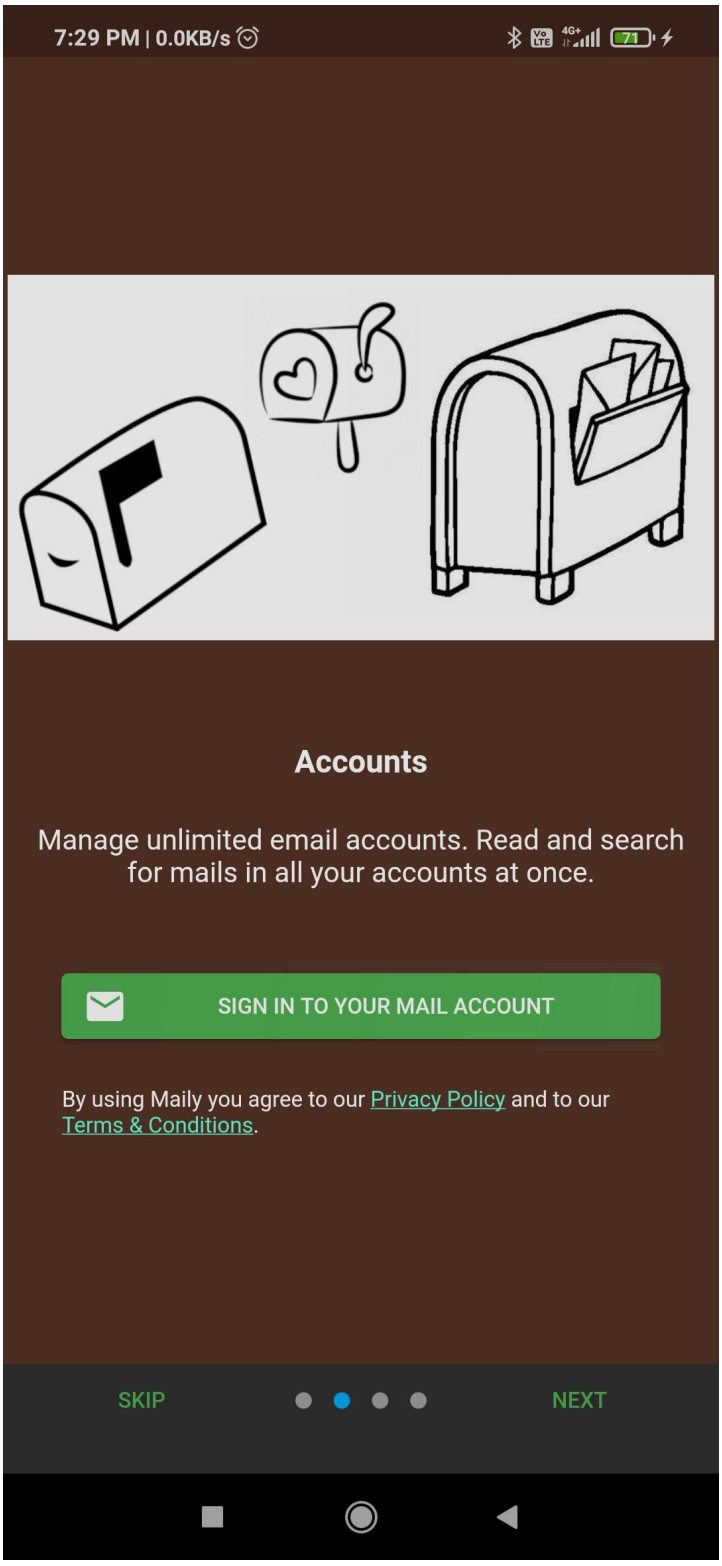

Fig. 7.1.2 Instruction 1

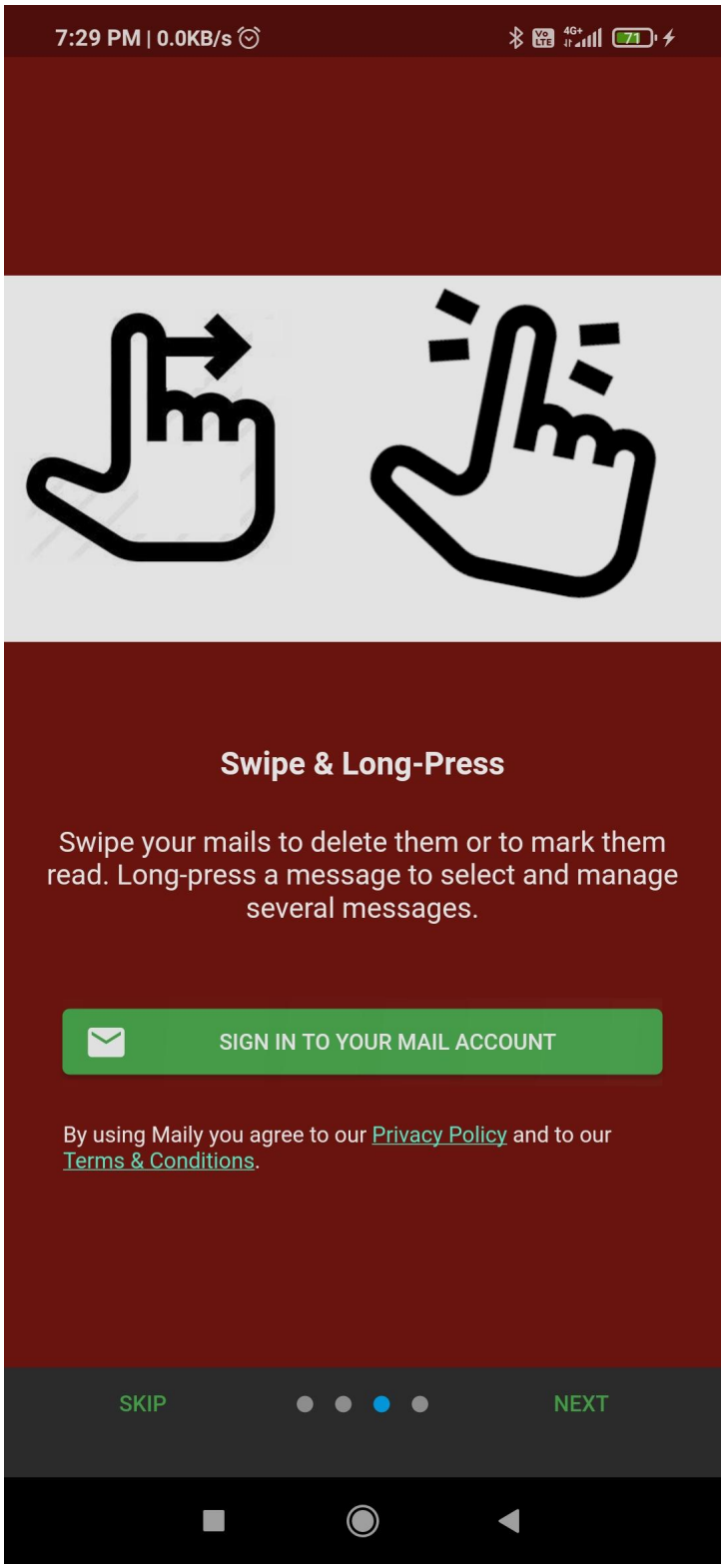

Fig. 7.1.3 Instruction 2

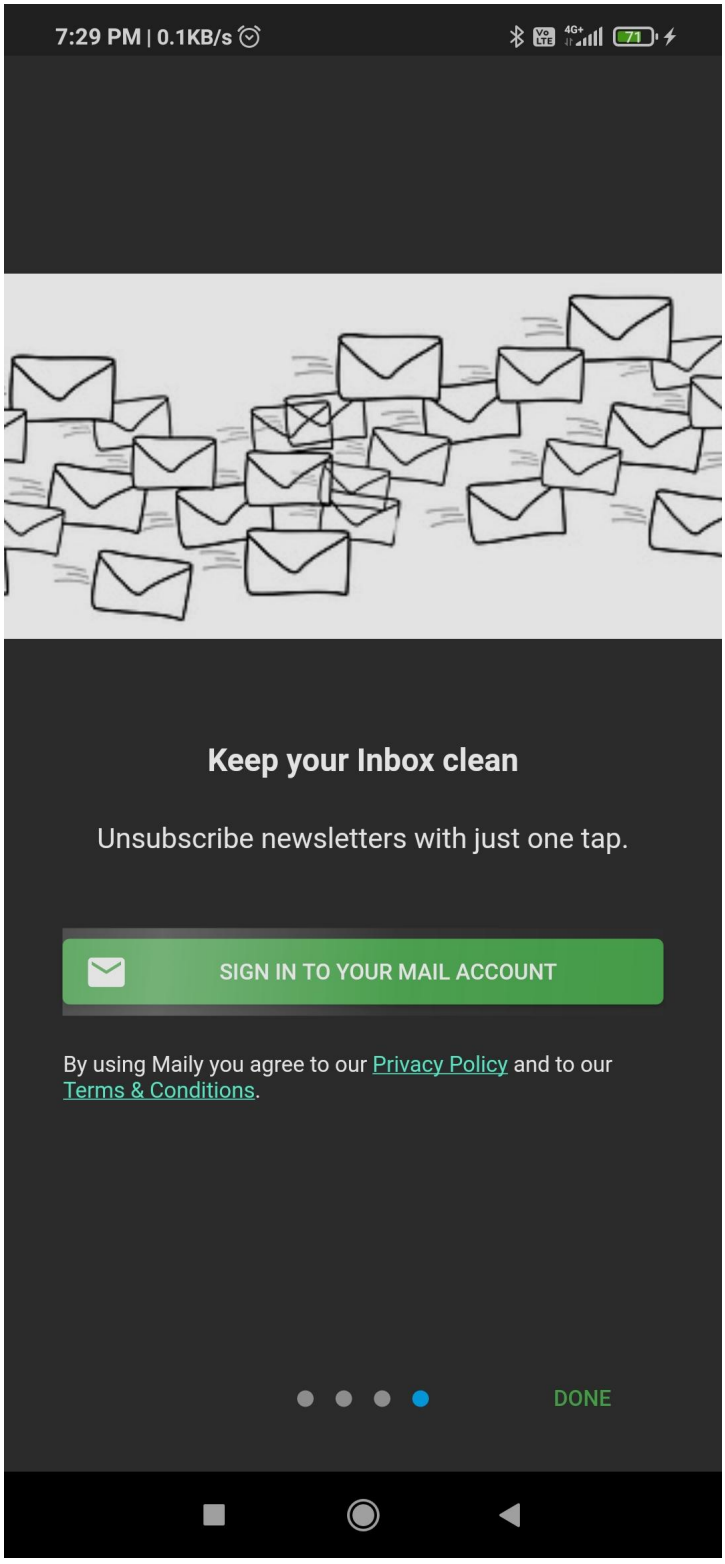

Fig. 7.1.4 Instruction 3

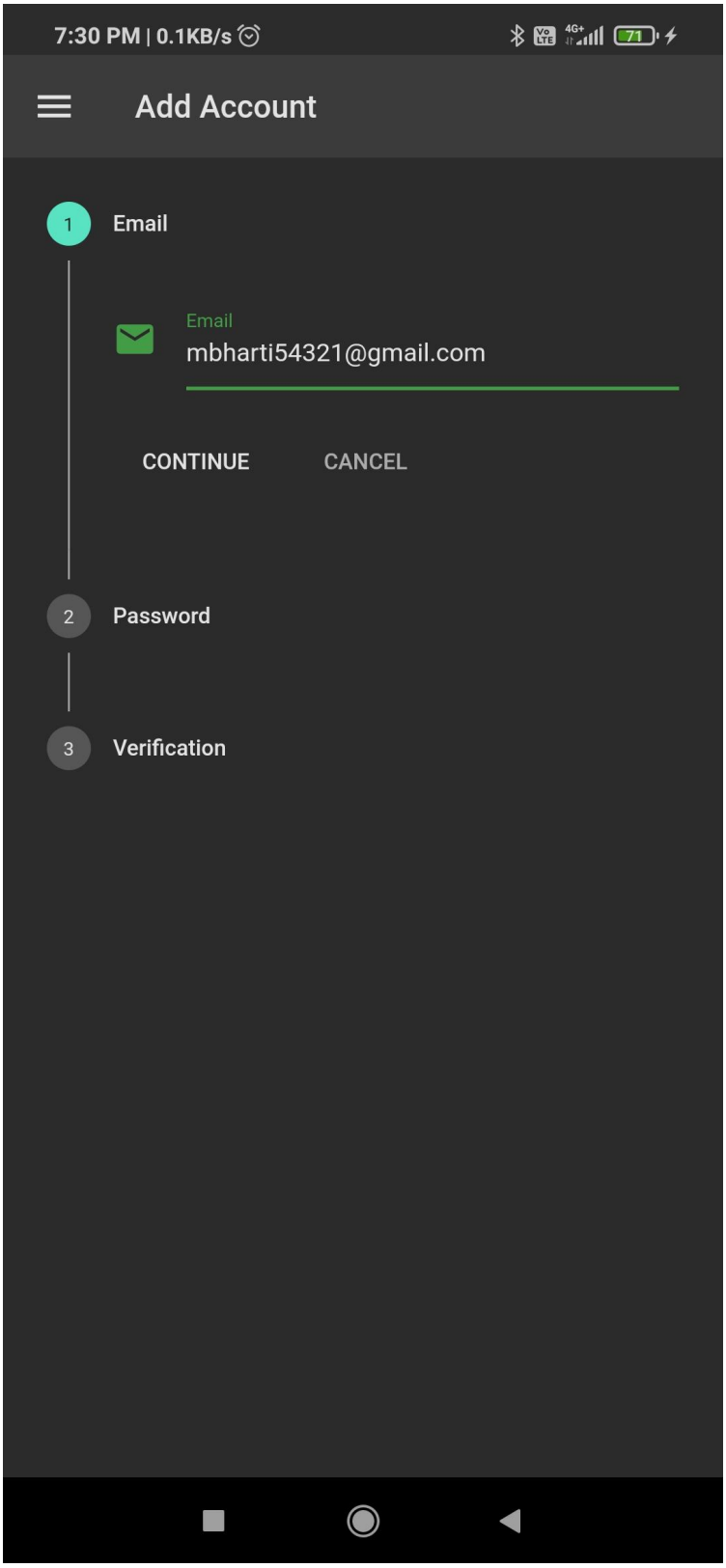

Fig. 7.1.5 login- email

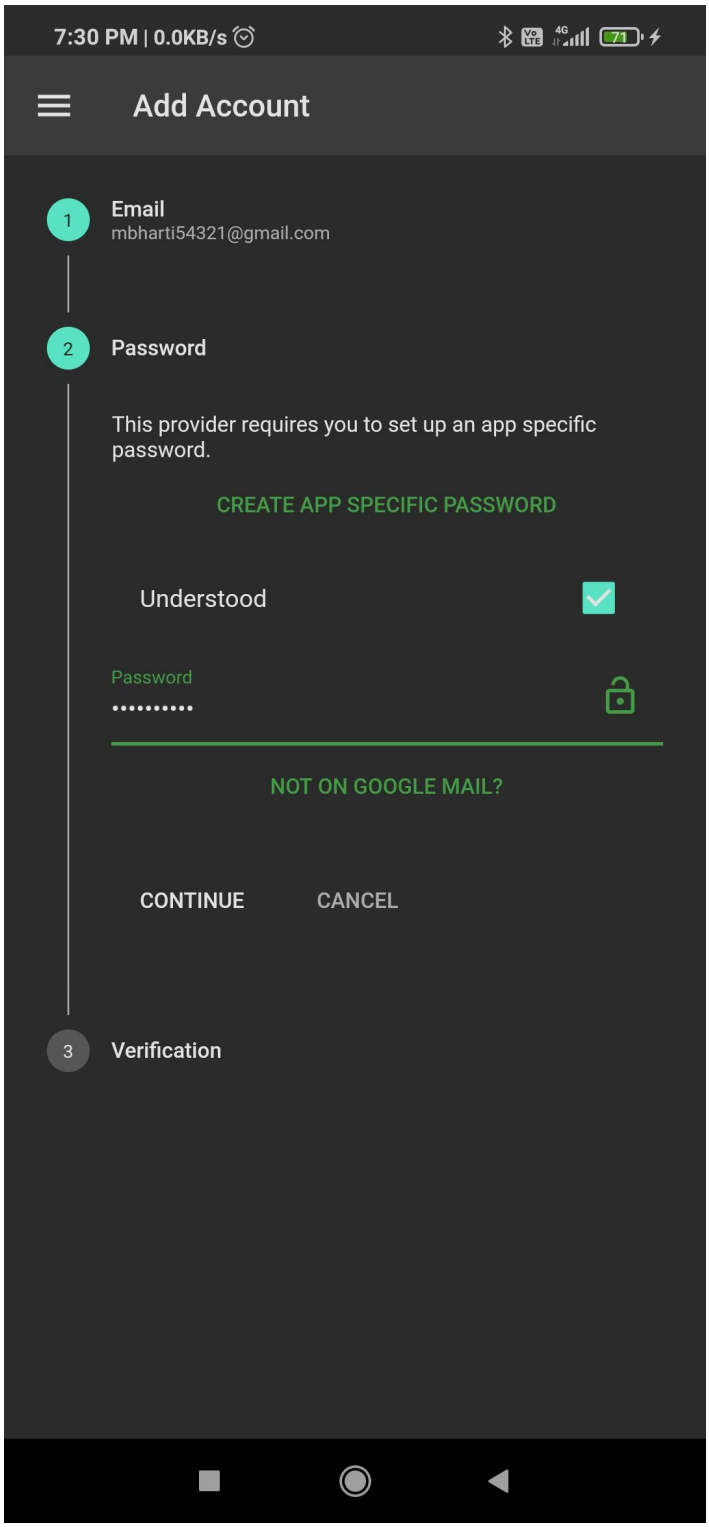

Fig. 7.1.6 login-password

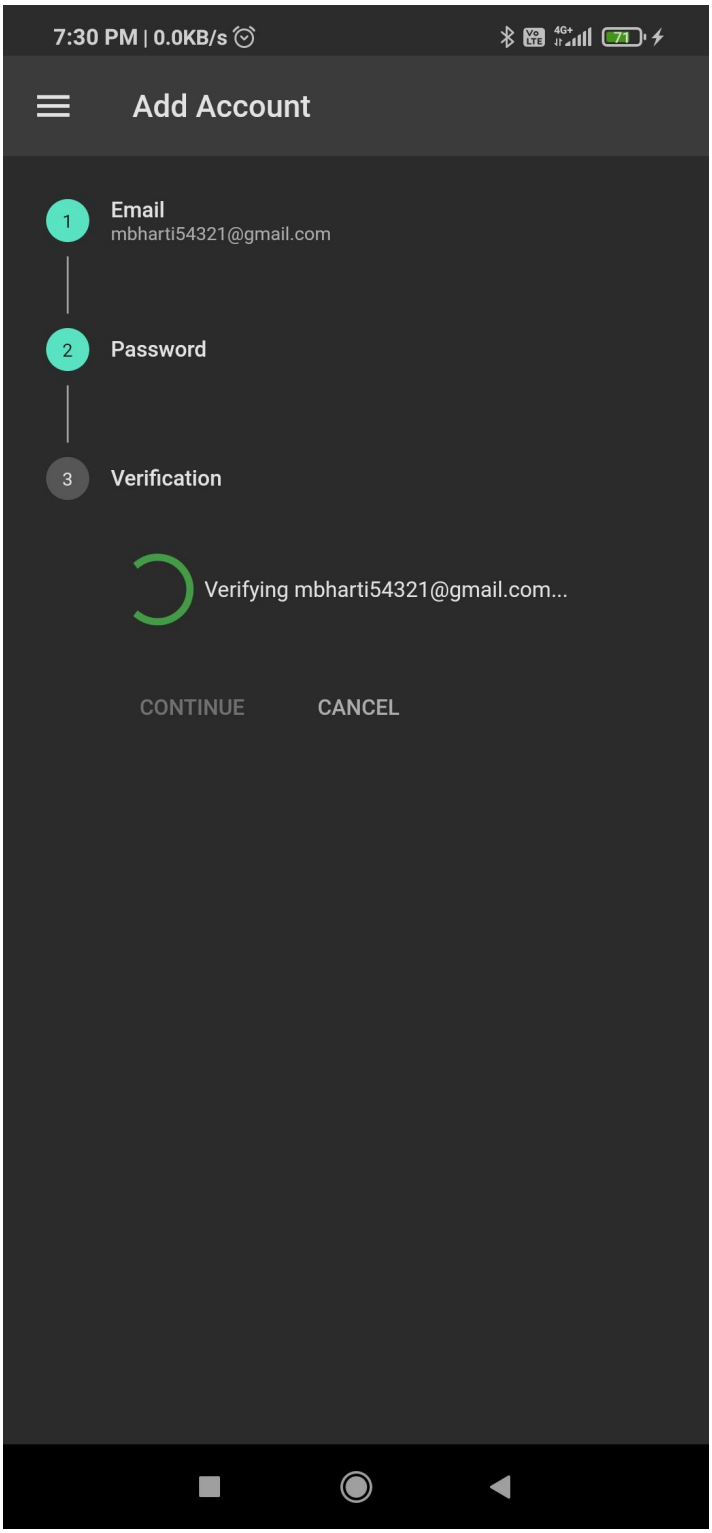

Fig. 7.1.7 Login- verification

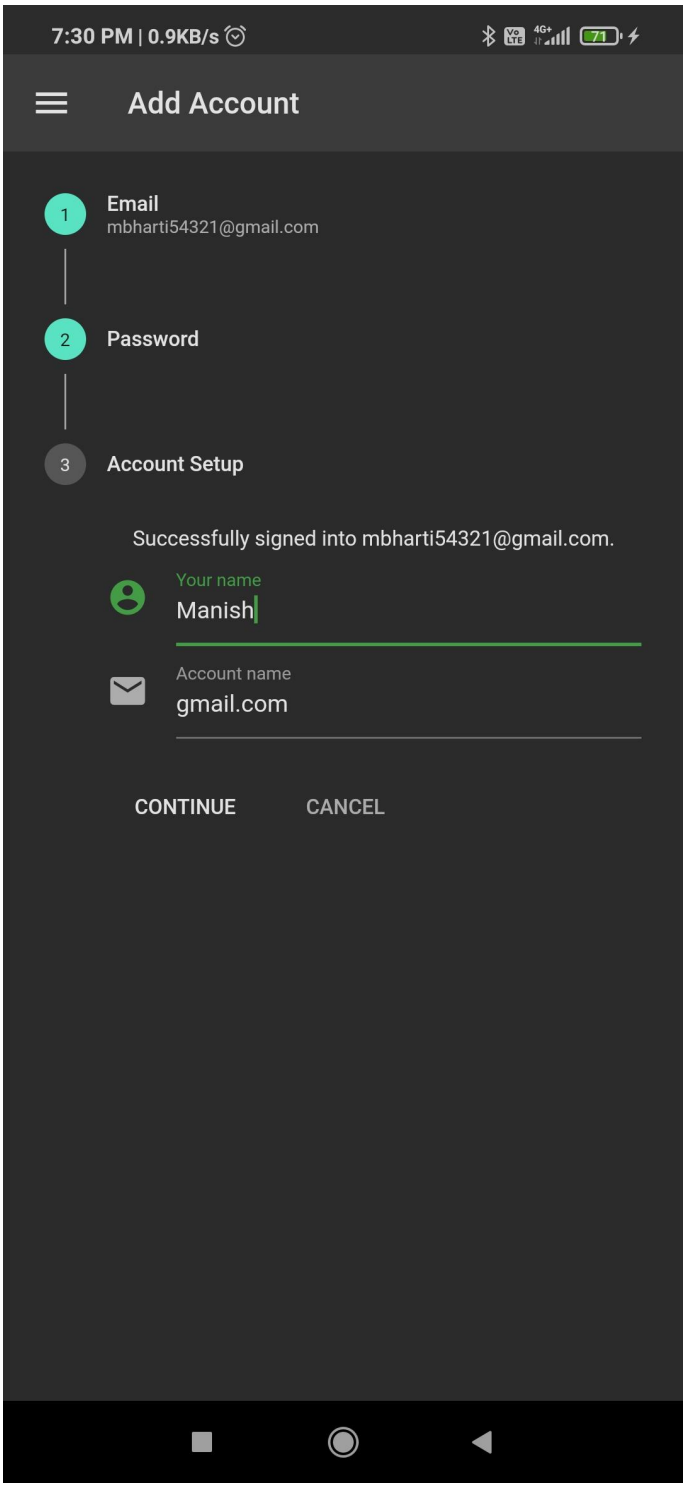

Fig.7.1.8 Login-User name

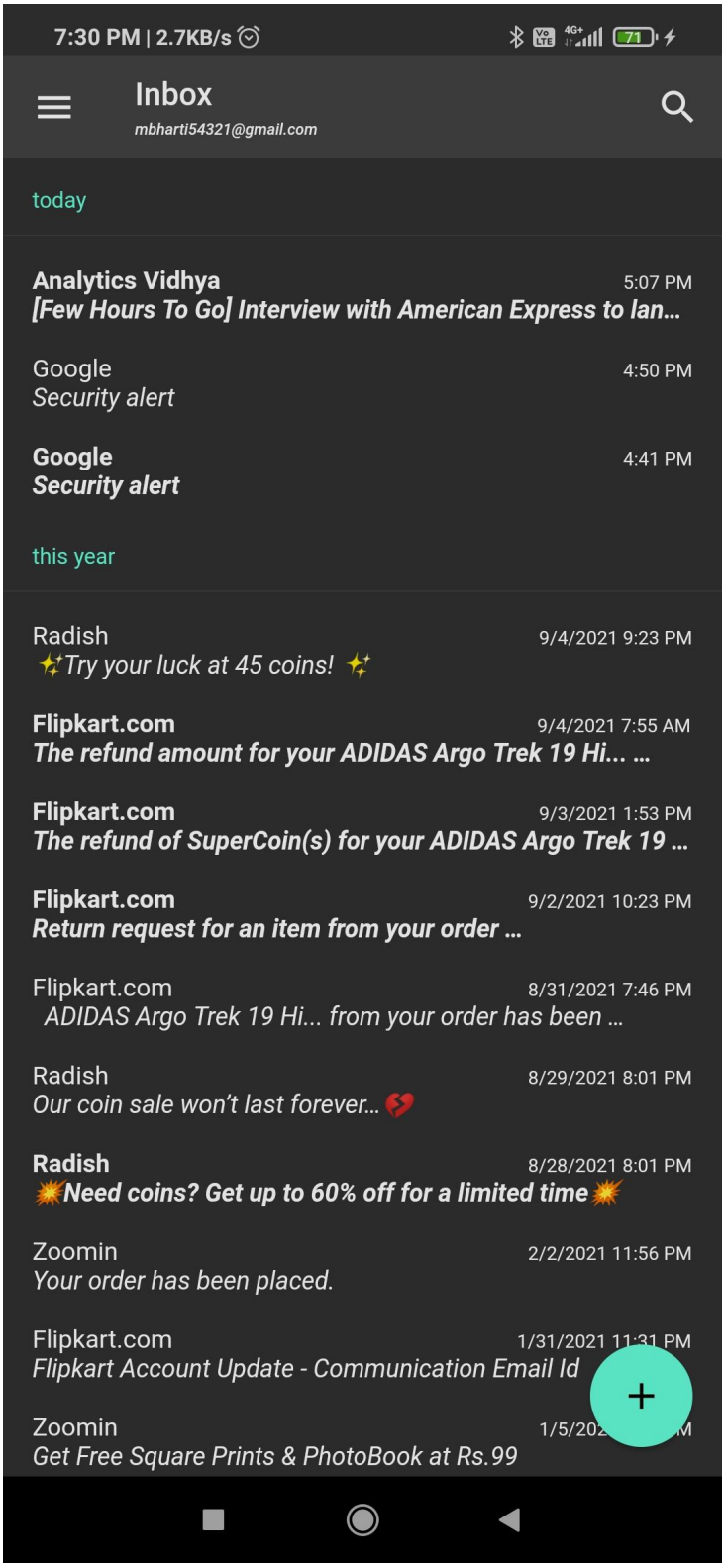

Fig. 7.1.9 User Emails

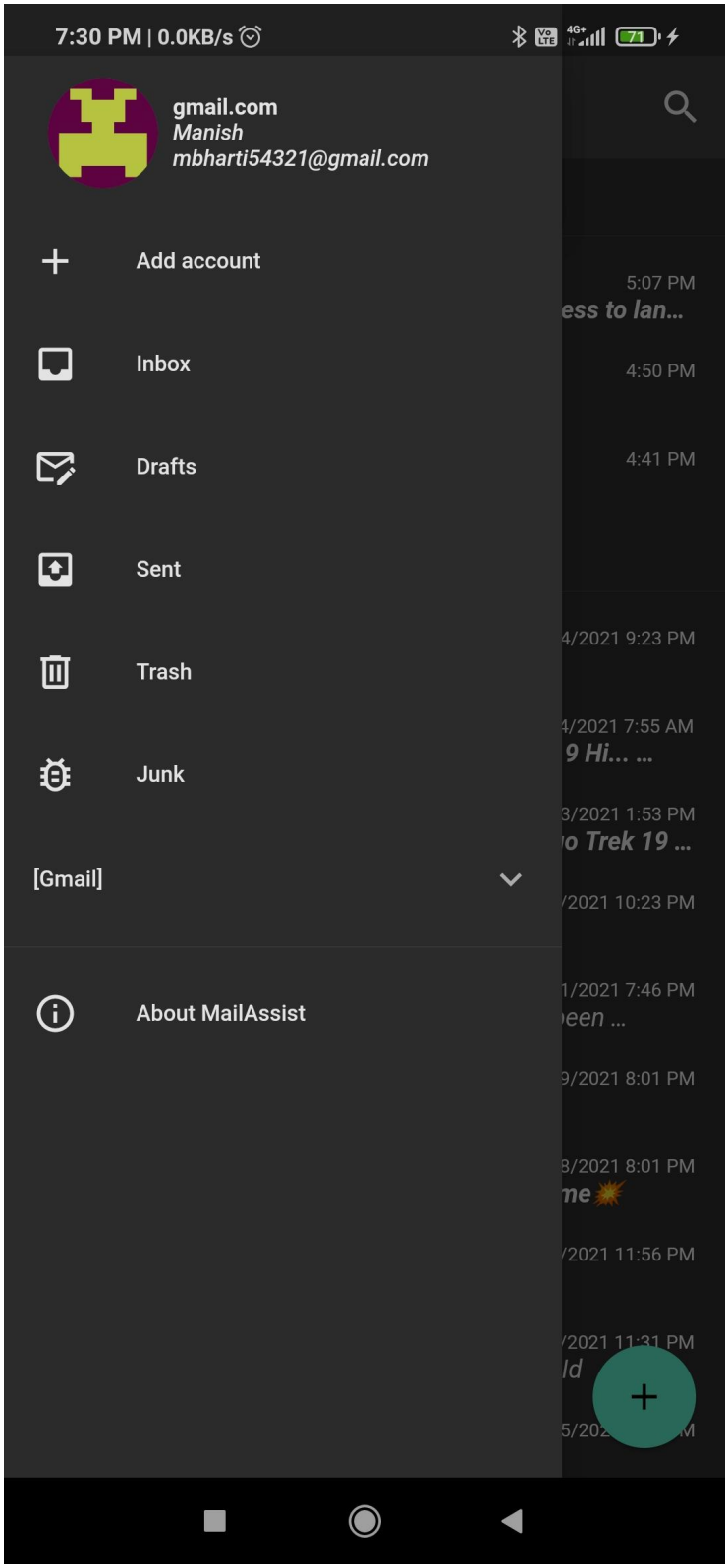

Fig. 7.1.10 drawer with features

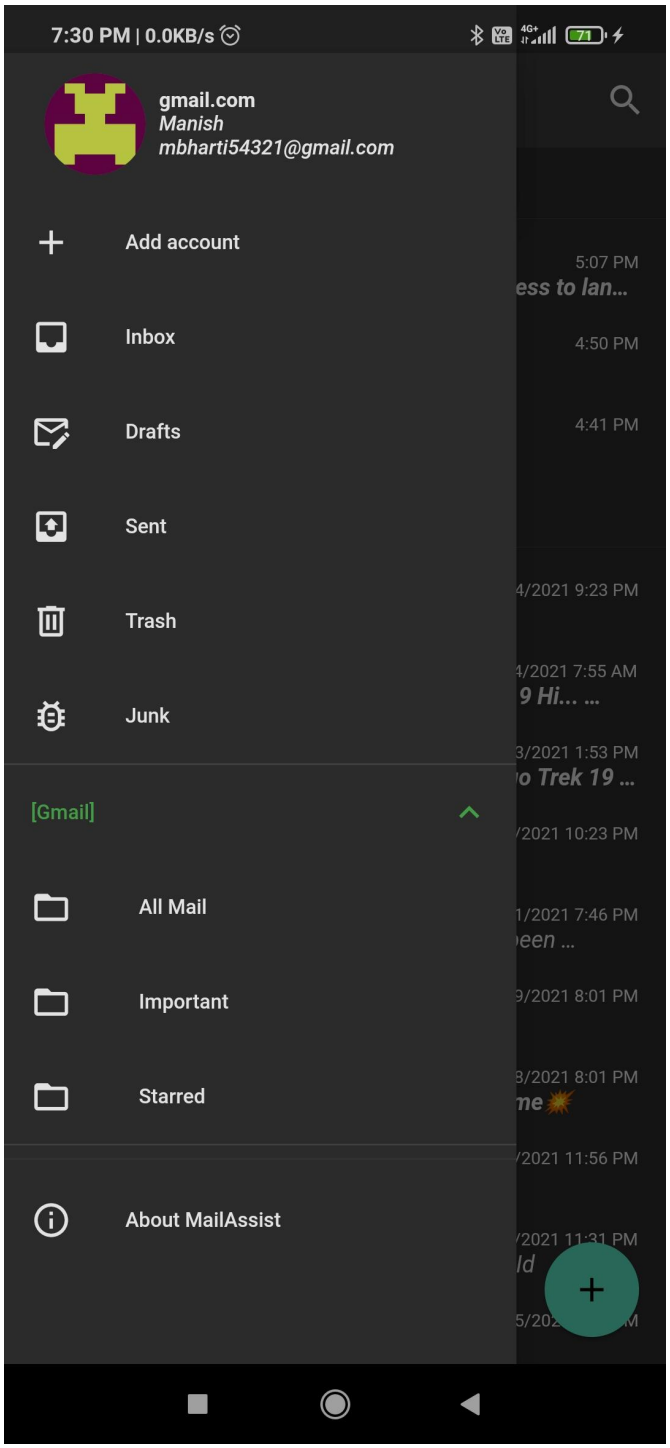

Fig. 7.1.11 Drawer with extra features

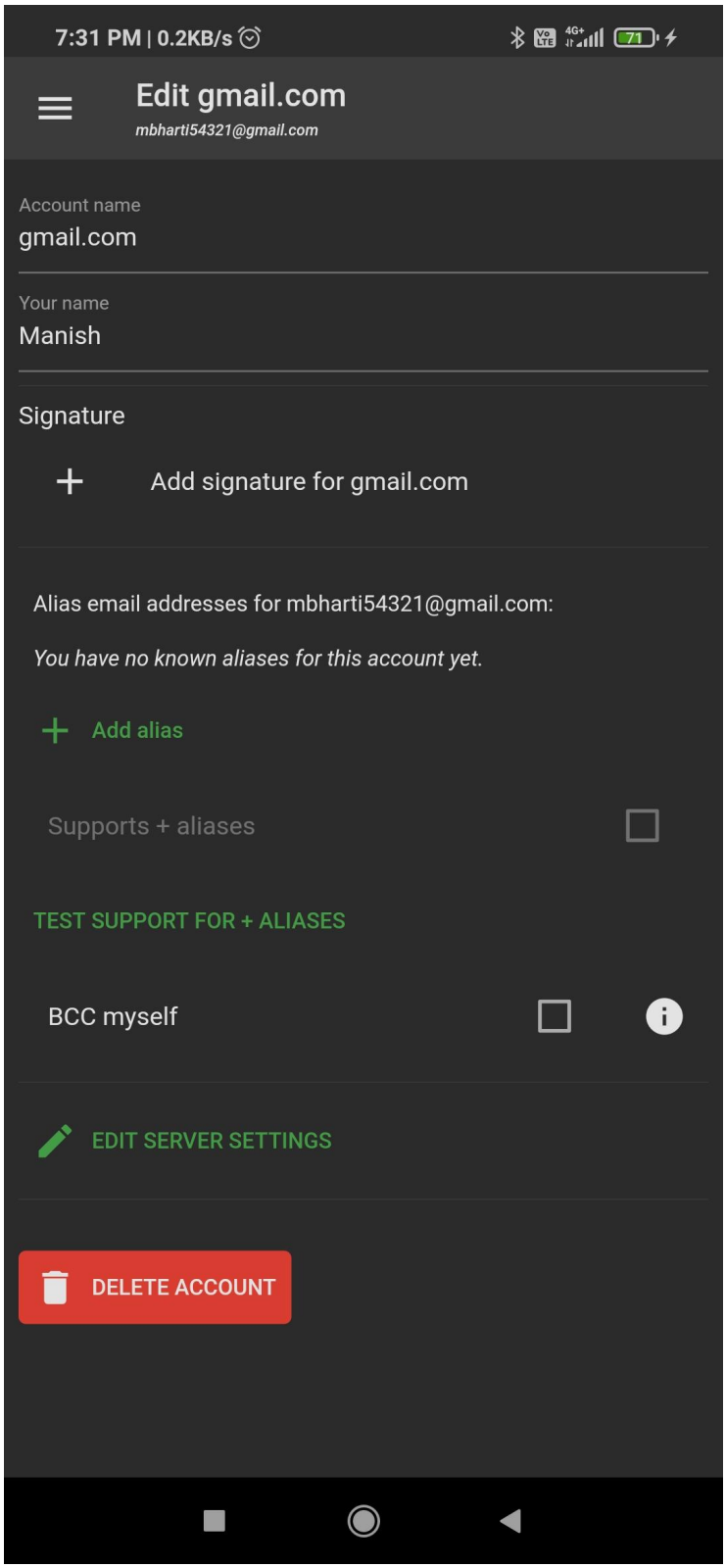

Fig. 7.1.12 Edit profile with delete account option

Fig. 7.1.13 Unsubscribe- button

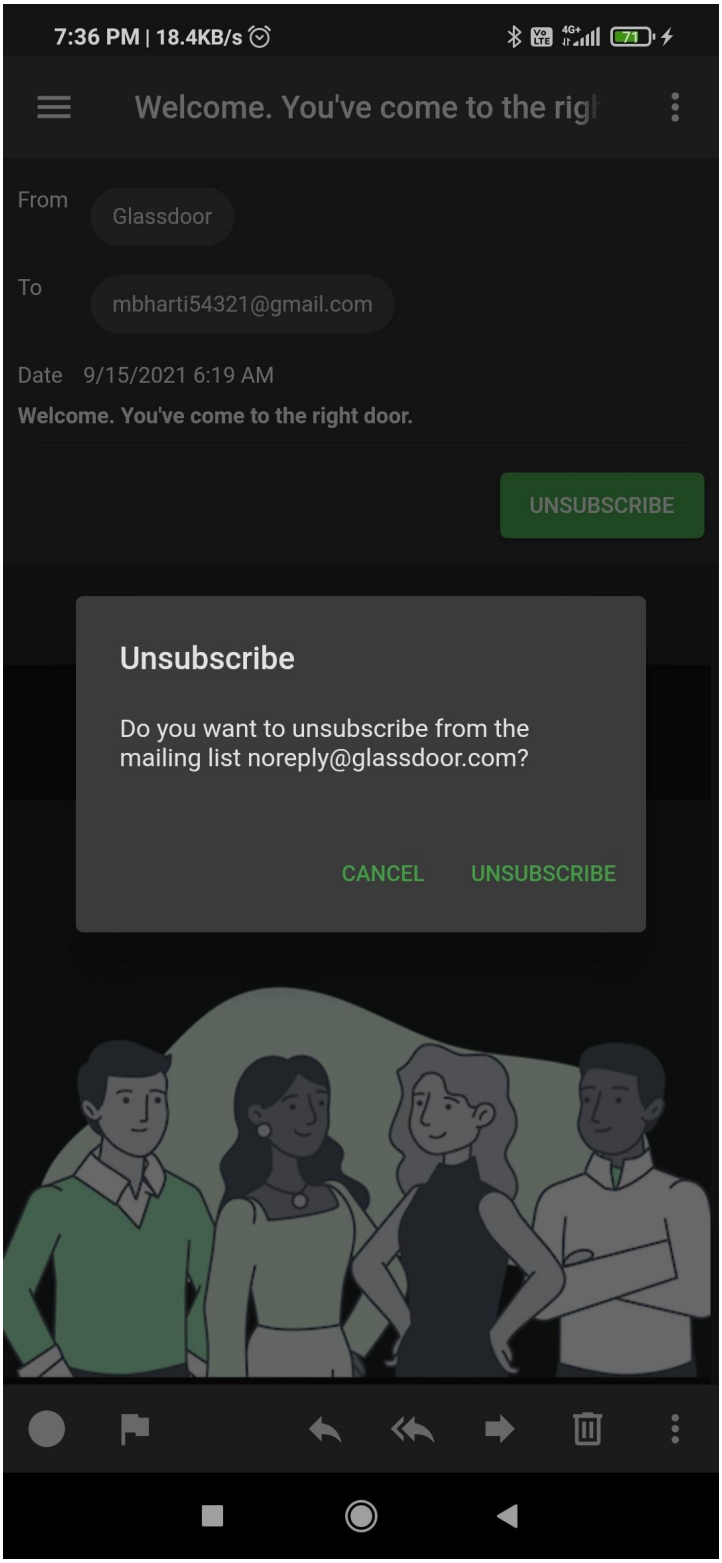

Fig. 7.1.14 Unsubscribe- asking confirmation

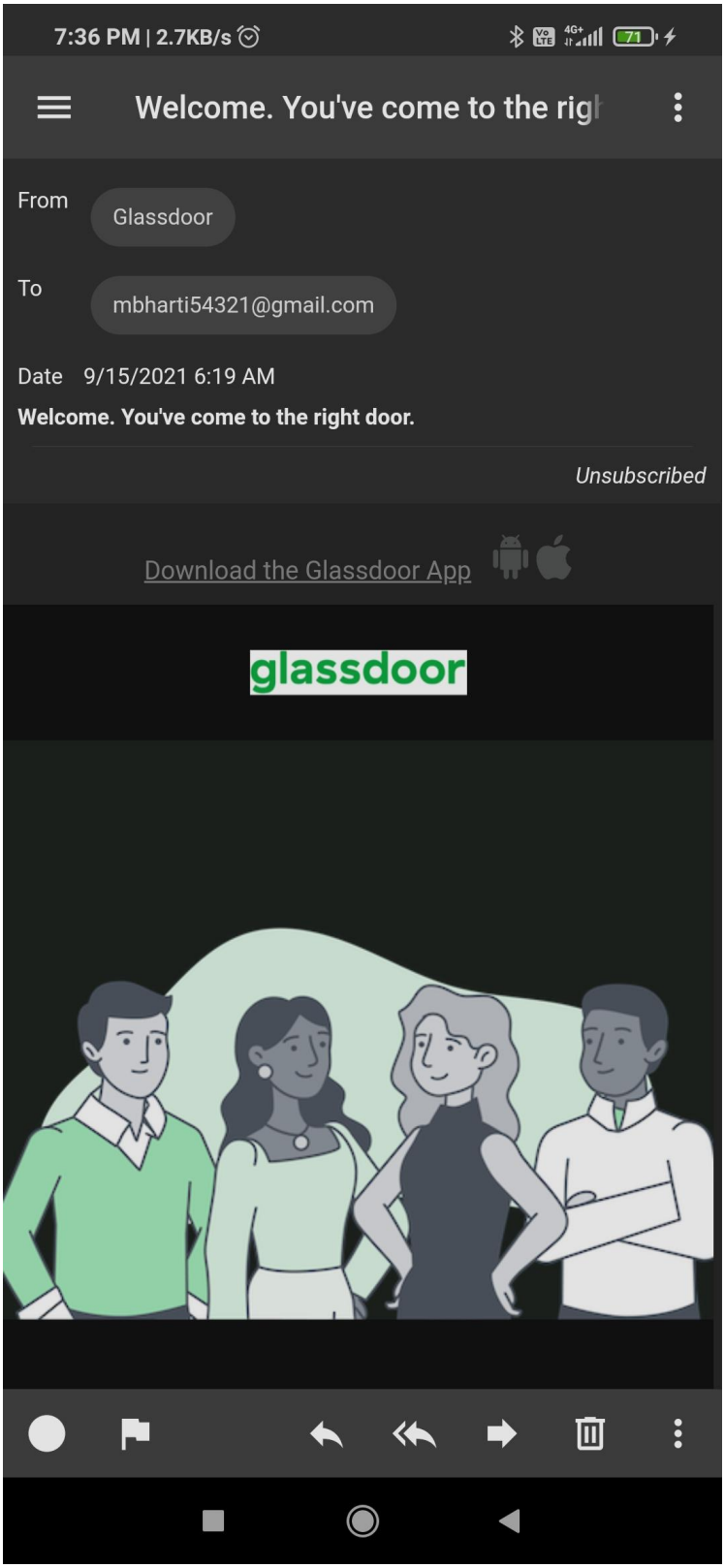

Fig. 7.1.15 After Unsubscribe

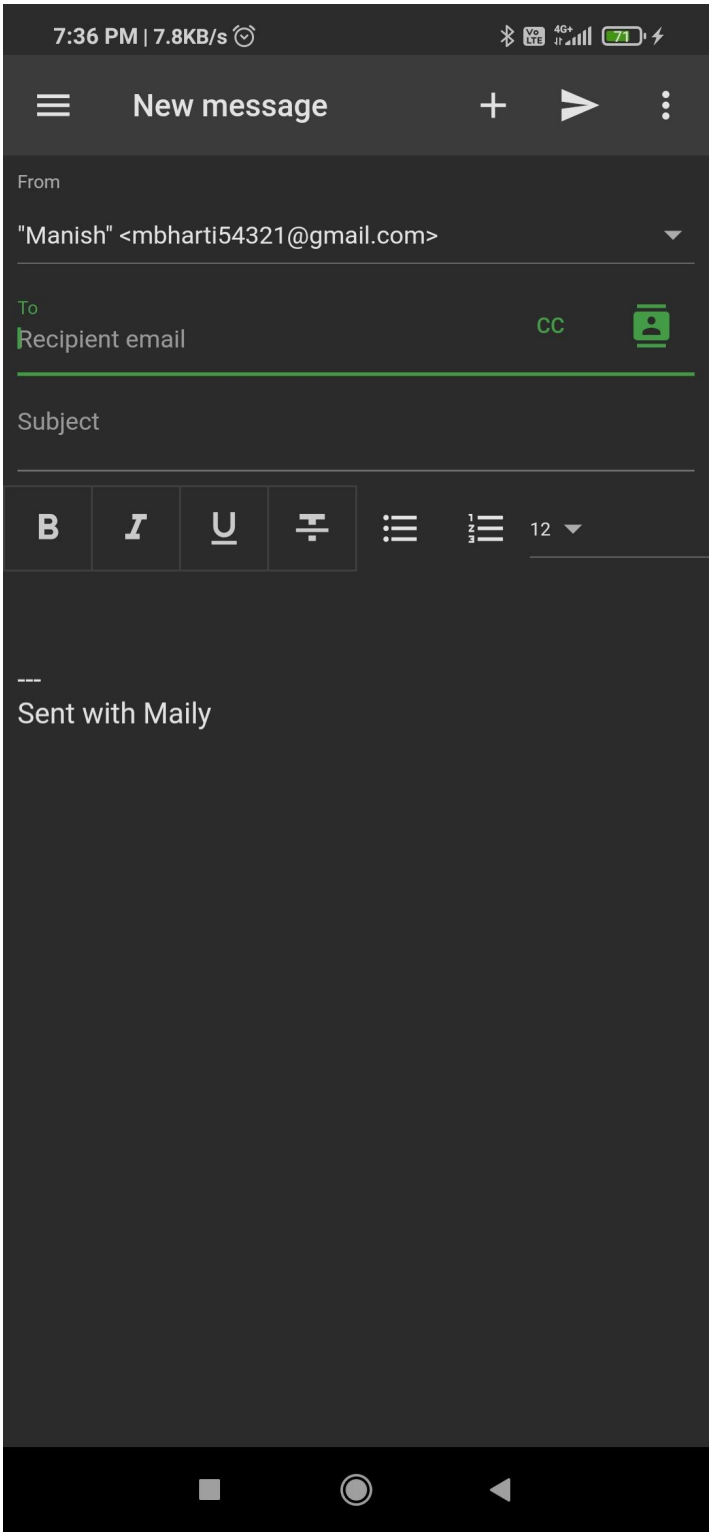

Fig. 7.1.16 Email send box

8. CONCLUSION

We all know how important emails are for communication in this digital age. However, we are all subjected to email spam. We have a lot of unwanted emails in our inbox that we don't want to read. The most commonly sent emails are promotional and e-commerce communications. Filtering important emails from a pile of all emails has been a time-consuming and inconvenient chore. Our critical communications might sometimes get lost in the stream of junk mail. To address this issue, we intend to develop a software solution that can filter and sort our emails. We want to facilitate users with simplification, so they don’t have to go and search tons of mails that they receive everyday. We are Categorizing mail based on meeting, travel etc and with one click they can view all the most important details relevant to that subject.

9. REFERENCES

1. https://docs.flutter.io/
2. https://developers.google.com/ml-kit
3. https://developers.google.com/gmail/api
4. https://www.youtube.com/watch?v=KJ_o-JUgGo0
5. https://medium.com/flutterdevs/roadmap-to-become-a-flutter-developer-resources-for-beginners-ccb68718c84b
6. https://material.io/design
7. Erker, Justus-Jonas, and Merlin Koehler. "Combining Unsupervised and Supervised Machine Learning for Semi-automatic E-Mail Classification." (2021).
8. Bermejo, Pablo, Jose A. Gámez, and Jose M. Puerta. "Improving the performance of Naive Bayes multinomial in e-mail foldering by introducing distribution-based balance of datasets." Expert Systems with Applications 38.3 (2011): 2072-2080.