FLUTTER APPLICATION WITH WHATSAPP INTEGRATION

Submitted by:

Yogheshwar Surjoo

Edoo Jameel Mohammad Ibraheem

MODULE NAME: PROGRAMMATION MOBILE HYBRIDE

Université des Mascareignes

Faculty of ICT

MARCH 2024

Table of Contents

1 | INTRODUCTION

In today's digital age, instant messaging has become an integral part of our daily communication routines. Among the myriad of messaging platforms available, WhatsApp stands out as one of the most popular and widely used applications globally. With its user-friendly interface, robust security features, and seamless cross-platform functionality, WhatsApp has revolutionized how individuals and businesses communicate.

The purpose of this project is to harness the power of WhatsApp's messaging capabilities and integrate them into a Flutter application, thereby providing users with a convenient and efficient way to engage with their contacts. By leveraging the WhatsApp Business API, we aim to bridge the gap between traditional messaging platforms and modern mobile applications, opening up new possibilities for communication and interaction.

1.1 Project Description

In this project, our main goal is to be able to successfully implement a WhatsApp plugin into an application called "UDM Connect" which is an application for the improvement of communication among students and tutors at UDM.

Our focus is on the practical side of the WhatsApp plugin implementation in terms of the UI representation and ease of use. Our design aims to resemble as much as possible the original design and UI of WhatsApp. This will ensure our WhatsApp integration is intuitive for all users as the vast majority of people already use WhatsApp in their everyday lives.

2 | METHODOLOGY

By following a comprehensive methodology, we were able to successfully integrate WhatsApp into a Flutter application.

Below is an outline of the main steps in the application development.

2.1 Project Planning and Research

- Documentation on the integration of WhatsApp into a flutter application to understand its features, capabilities, and limitations.
- Analyzed existing Flutter packages and libraries for integrating with WhatsApp, evaluating their suitability for our project requirements.
- Defined the project goals, scope, and deliverables based on the project requirements.

2.2 Analysis and Design

- Basic use case definition for the application
- Construct an intuitive wireframe design for the user interface of the application using Figma.

2.3 Environment Setup and Tool Selection

- Installed and configured Flutter SDK, ensuring compatibility with the latest stable release.
- Selected Visual Studio as the primary Integrated Development Environment (IDE) for Flutter development, leveraging its comprehensive suite of tools and plugins.

2.4 Application Development and Implementation

- Implemented features such as chat threads, message composition, call management, and status to provide a seamless messaging experience.
- Optimized UI performance and responsiveness for various screen sizes and orientations, ensuring a consistent user experience across different devices.

2.5 Analysis of Results

- Documented the project architecture, codebase structure, and UI integration details to gain insight into possible future developments for the application.

3 | ANALYSIS AND DESIGN

3.1 Definition of Use Cases

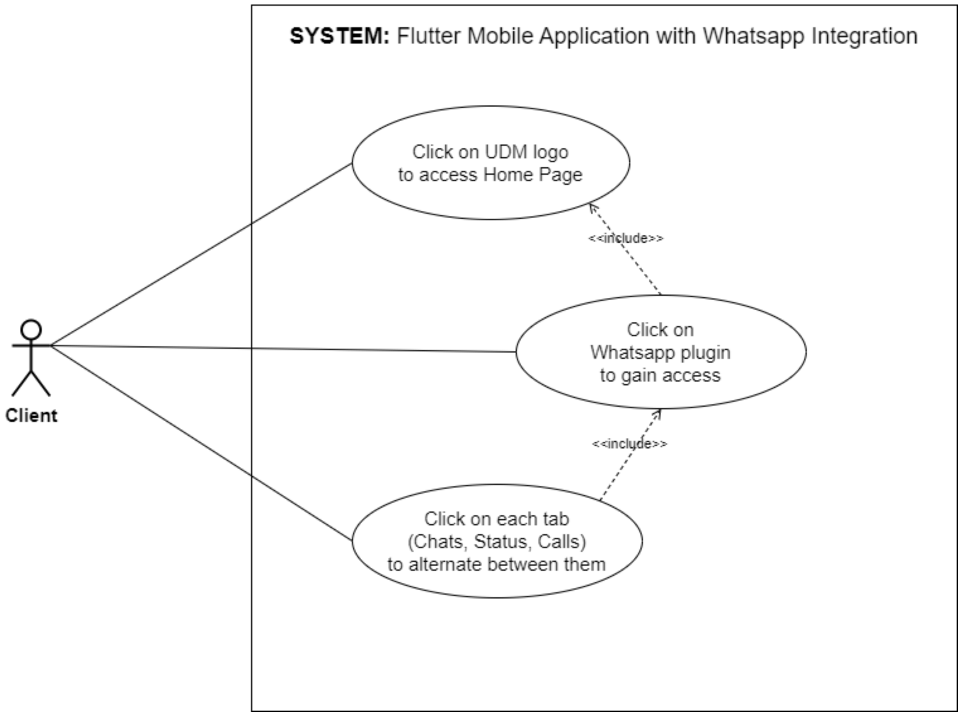
The definition of a use case in light of this project is a written description of a specific task or action which will be performed by the user (actor) of the application.

The list of use cases is as follows:

- Access Home Page by clicking on the UDM logo on the application's Welcome Page
- Access the WhatsApp plugin from the application Home Page by clicking on the WhatsApp icon
- Alternate between the 3 tabs (Chats, Status and Calls) on the WhatsApp UI.

3.2 Use Case Diagram

The following is a simple use case diagram to better represent the flow of the use cases within the application

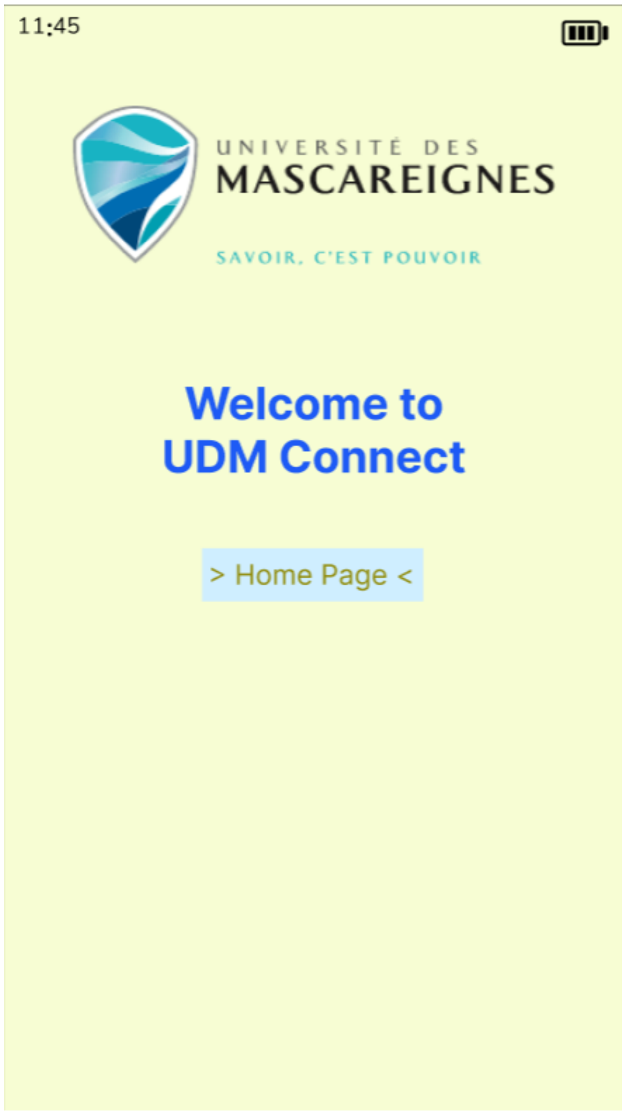

SYSTEM: Flutter Mobile Application with Whatsapp Integration
Click on UDM logo to access Home Page
<<include>>
Click on Whatsapp plugin to gain access
<<include>>
Click on each tab (Chats, Status, Calls) to alternate between them
Client

3.3 <u>Wireframe Design</u>

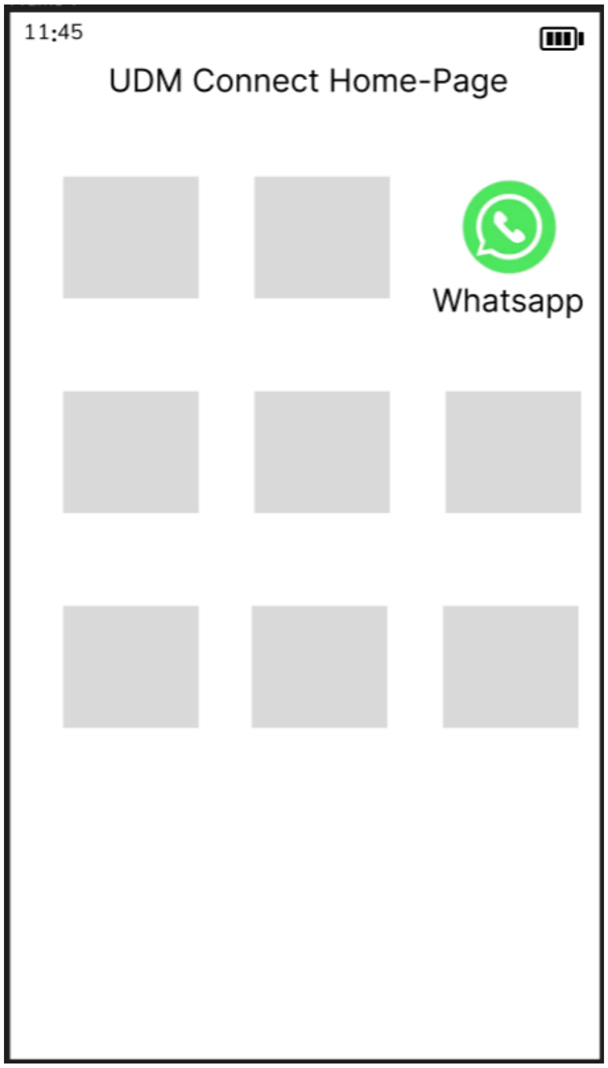
A wireframe design is a visual blueprint or skeletal framework that outlines the basic structure and layout of a user interface (UI) without focusing on visual design elements such as colors, fonts, or graphics. Its purpose is to provide a conceptual representation of the app's layout, navigation flow, and functionality to facilitate the development of the application.

The wireframe for the application's main sections or pages are as follows:

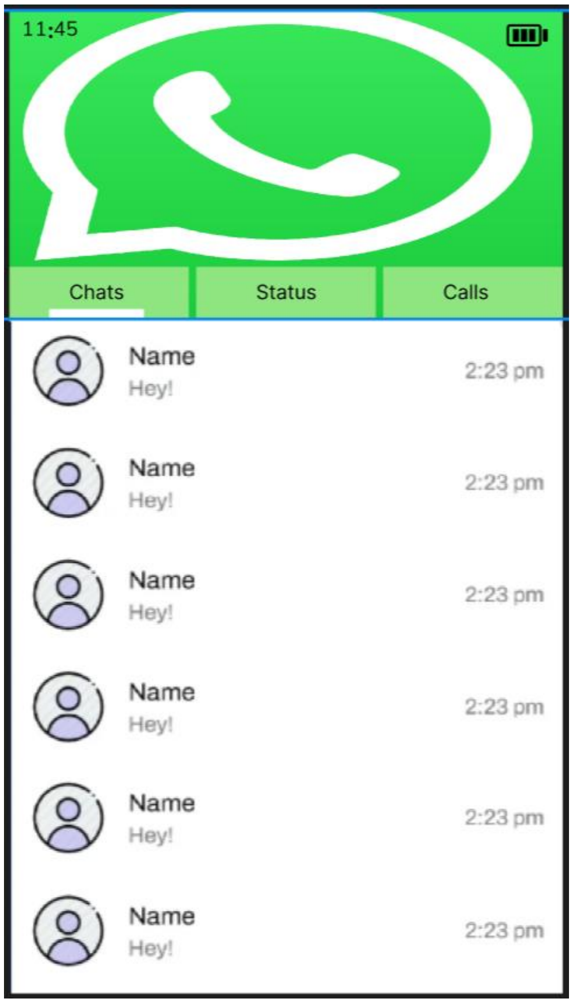
Welcome Page:



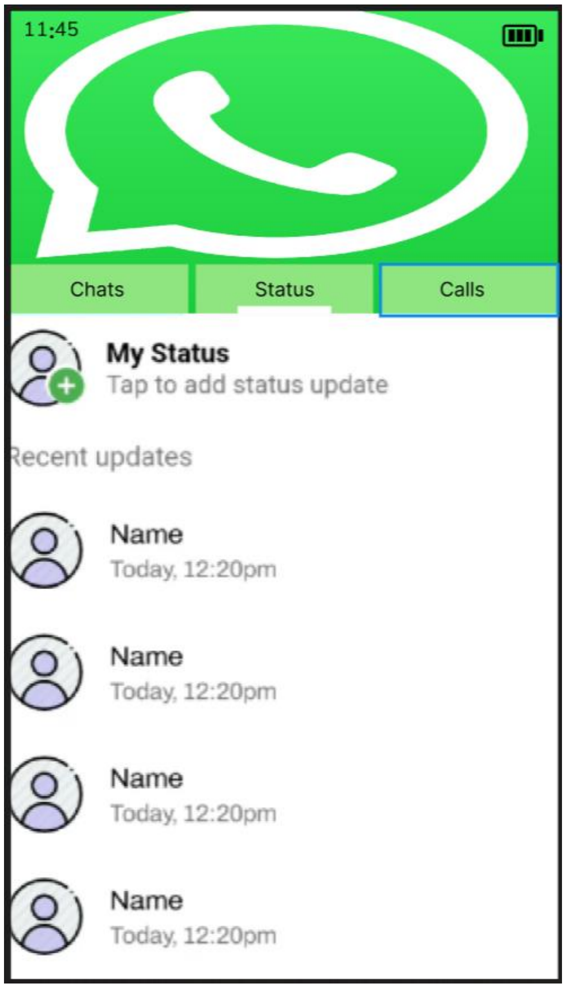
Home Page:



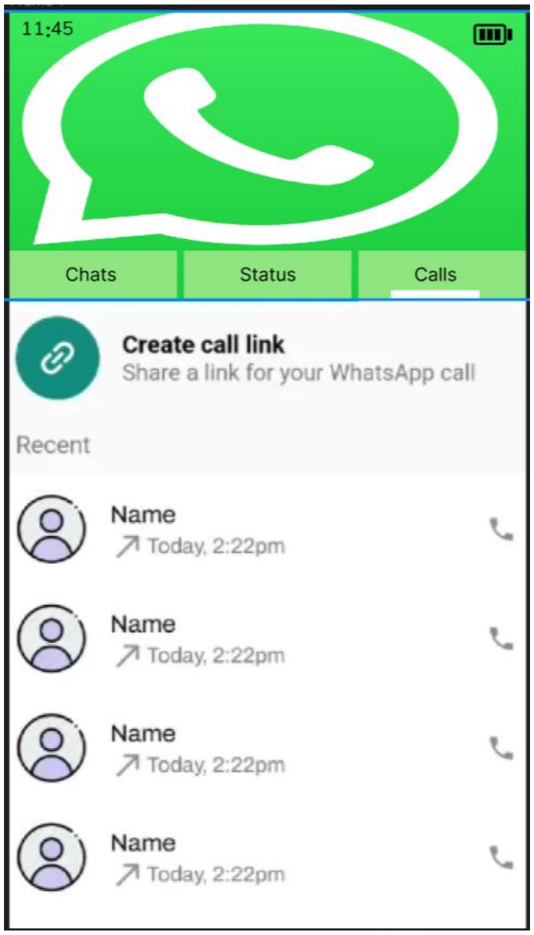
WhatsApp Plugin – Chats:



WhatsApp Plugin – Status:



WhatsApp Plugin – Calls:

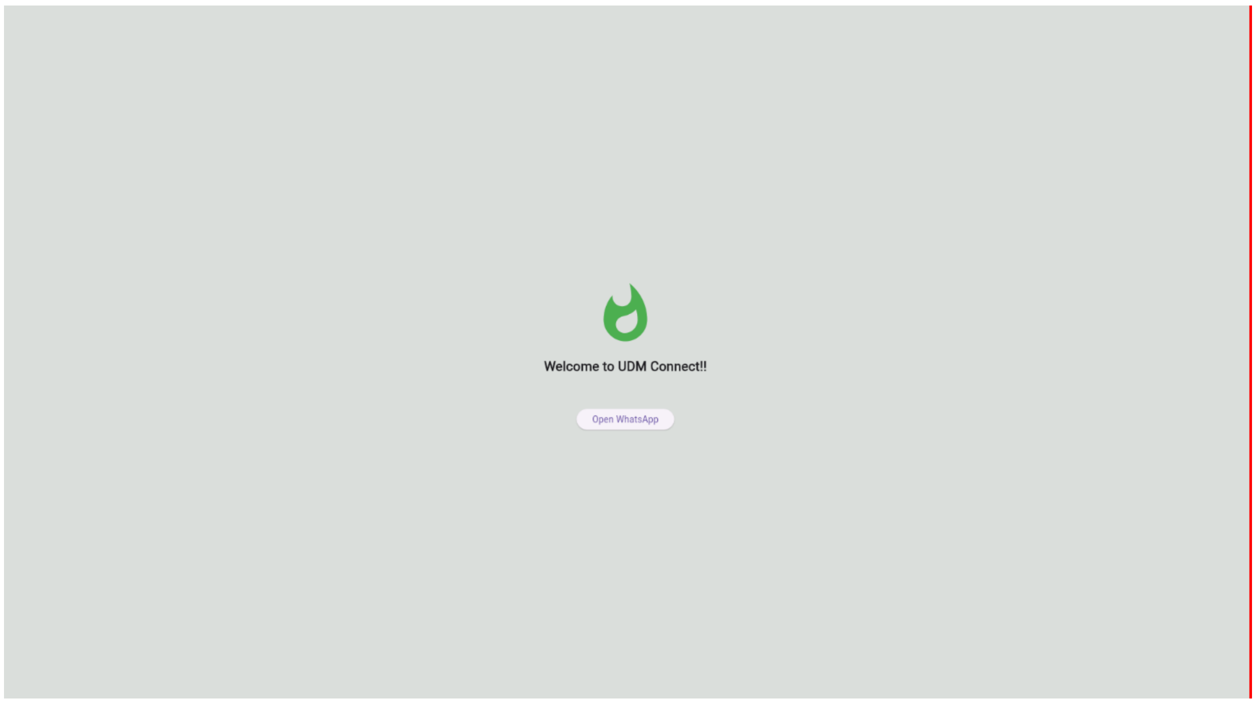


4 | System Development

4.1 <u>Description of Application Functionalities</u>

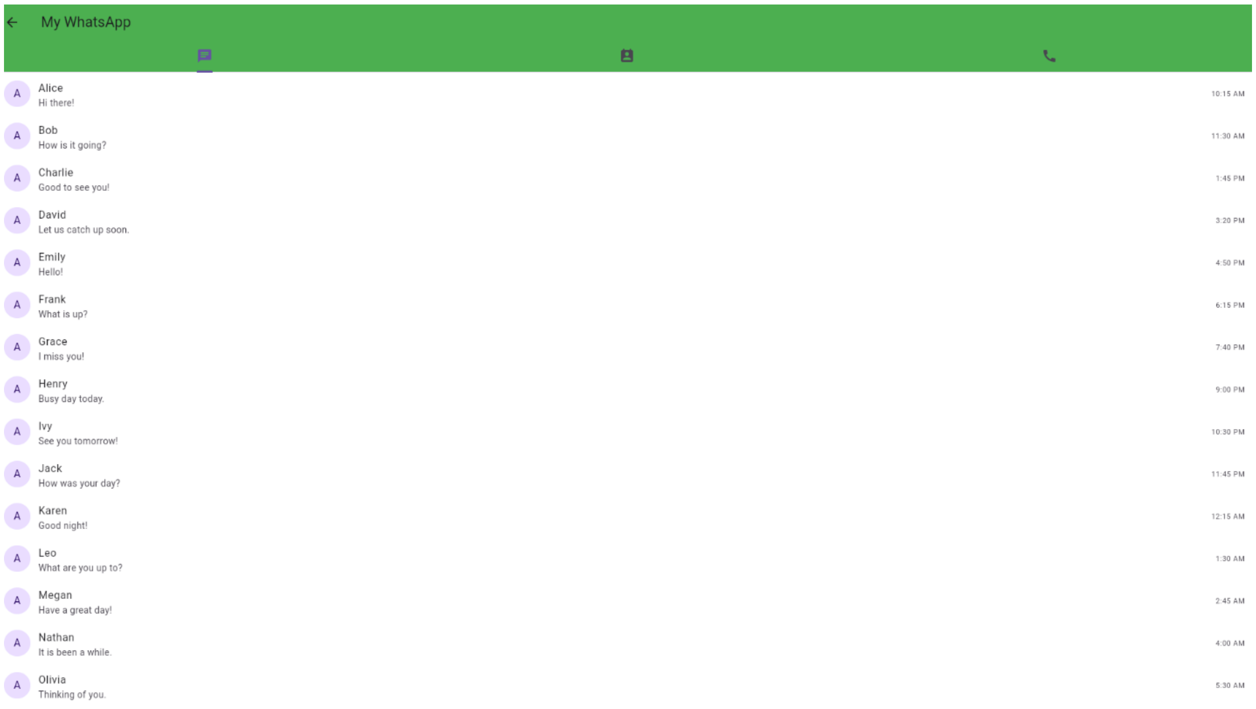
The application consists of a welcome/home page, where the user will have the option to choose to open the WhatsApp plugin within the application.

The welcome page is as follows:



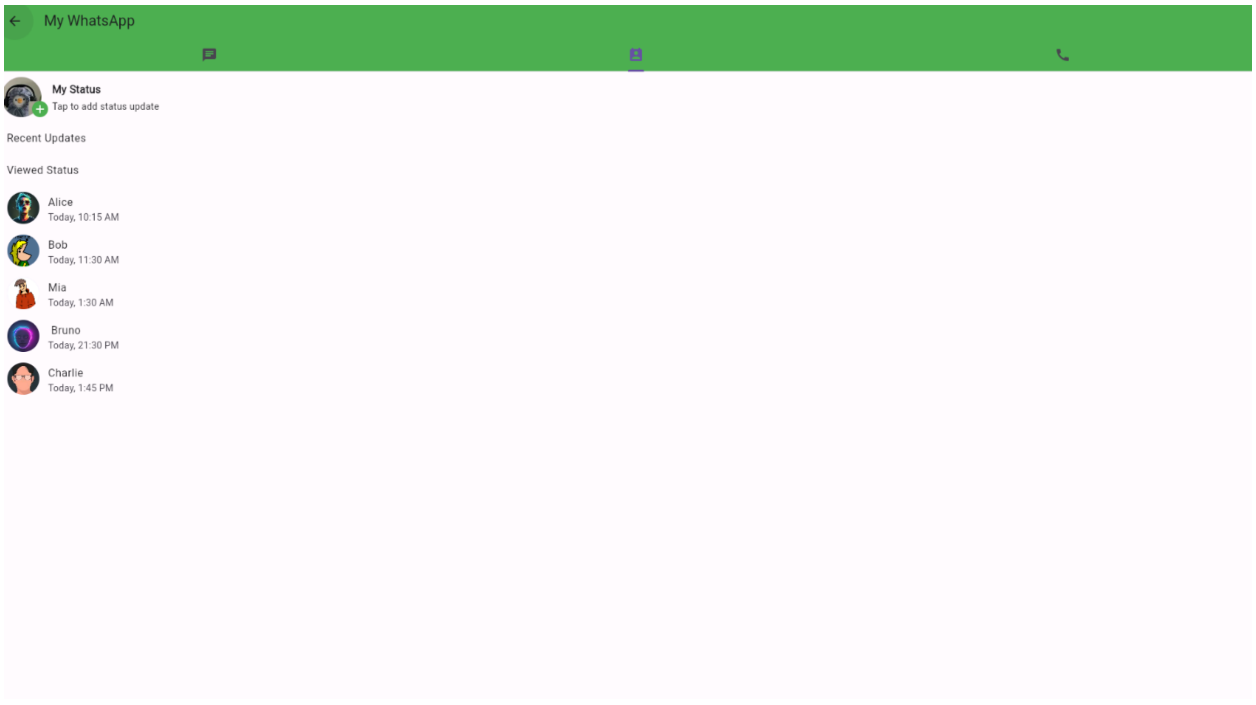
Once the user click on "Open WhatsApp", the application redirects the user to the WhatsApp plugin, where he will have access to his WhatsApp account. Within this user interface, the user can alternate between the 3 tabs, namely "Chats", "Status" and "Calls".

The following images demonstrate the different options available to the user once within the WhatsApp plugin.



My WhatsApp

A	Alice Hi there!	10:15 AM
A	Bob How is it going?	11:30 AM
A	Charlie Good to see you!	1:45 PM
A	David Let us catch up soon.	3:20 PM
A	Emily Hello!	4:50 PM
A	Frank What is up?	6:15 PM
A	Grace I miss you!	7:40 PM
A	Henry Busy day today.	9:00 PM
A	Ivy See you tomorrow!	10:30 PM
A	Jack How was your day?	11:45 PM
A	Karen Good night!	12:15 AM
A	Leo What are you up to?	1:30 AM
A	Megan Have a great day!	2:45 AM
A	Nathan It is been a while.	4:00 AM
A	Olivia Thinking of you.	5:30 AM



My WhatsApp

My Status
Tap to add status update

Recent Updates

Viewed Status

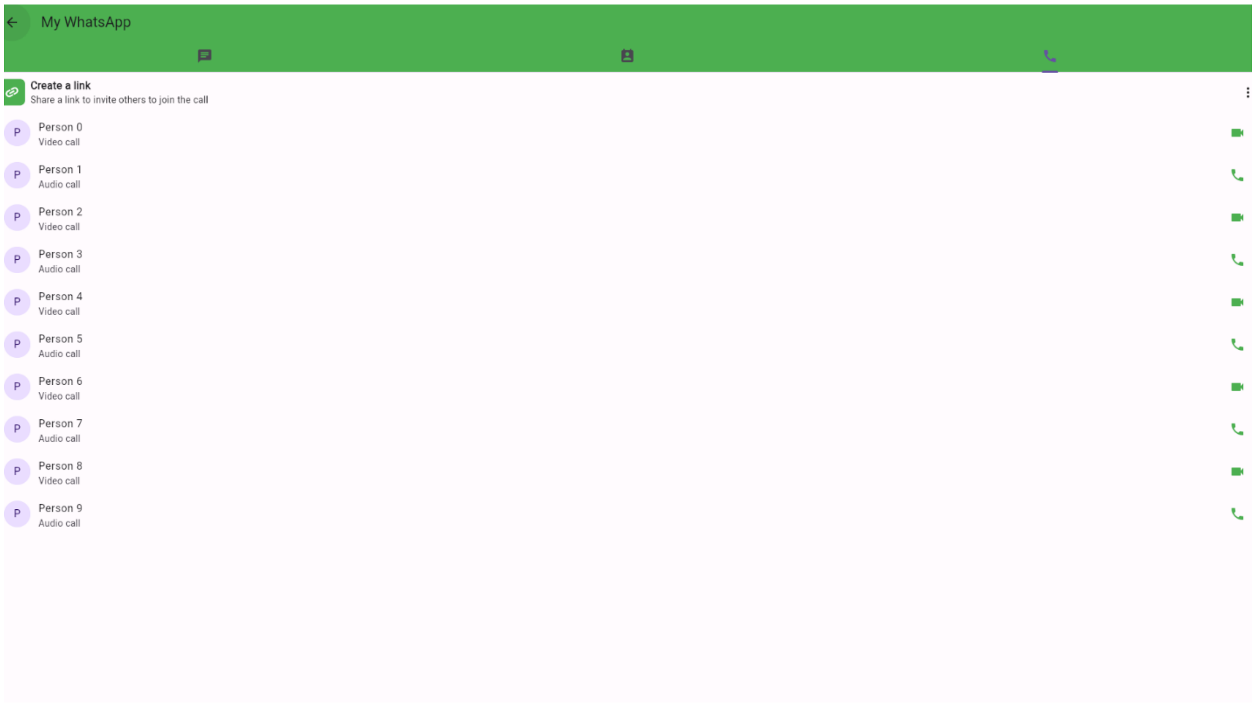
Alice
Today, 10:15 AM

Bob
Today, 11:30 AM

Mia
Today, 1:30 AM

Bruno
Today, 21:30 PM

Charlie
Today, 1:45 PM


My WhatsApp

Create a link
Share a link to invite others to join the call

Person 0
Video call

Person 1
Audio call

Person 2
Video call

Person 3
Audio call

Person 4
Video call

Person 5
Audio call

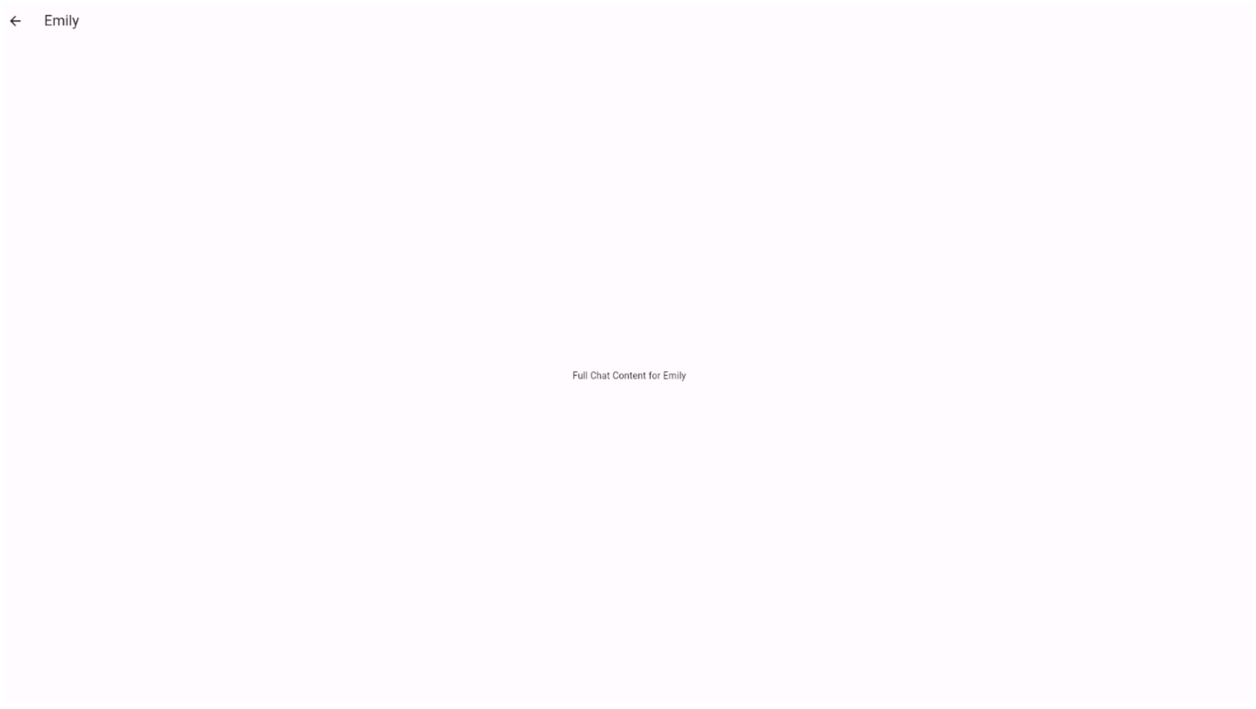
Person 6
Video call

Person 7
Audio call

Person 8
Video call

Person 9
Audio call

This last image shows the UI when the user selects the chat of a specific contact:


Emily

Full Chat Content for Emily

4.2 Development and Coding

Designing UI

So, this section is all about making a UI section of the Whatsapp application using Flutter. We will divide it into different sections like Chat, Status, Calls & Message.

First, we will make a home screen having TabBar in it.

HOME

So, here we have to make titles and icons in the end as we see in the Whatsapp. We will also make TabBar which will have 3 tabs - **Chats, Status** and **Calls**.

- For this, first, make a class and extend it with SingleTickerProviderStateMixin .

```
class _HomePageState extends State<HomePage>  with
SingleTickerProviderStateMixin {}
```

- Then make a TabController and initialize it in the initState() method.

```
TabController? tabController;
@override
void initState() {
 super.initState();
 tabController = TabController(length: 3, vsync:
this);
}
```

- Then coming to the UI part, we will use the SliverAppBar widget, which gives different scrolling effects on the appbar. **SliverAppBar** gives us the means to create an app bar that can change appearance, blend in the background, or even disappear as we scroll. The same feature is seen in the WhatsApp application.

  We will define the title inside it and the icons inside the actions:[] property. For using tabs, we will define it in the bottom property of SliverAppBar.

```
  SliverAppBar(
         title: Text(
           "WhatsApp",
           style: TextStyleConstants.white16,
         ),
         actions: [
           // define icons
         ],
         bottom: TabBar(
             // define tabs and other properties of TabBar
             ),
       ),
```

- So finally, to show the tab's data, we will use the TabBarView widget on the body property of NestedScrollView and will show the UI of all three(Chat, Status, Calls) parts inside it.

  We will make different classes for this purpose to follow the Single Responsibility of the SOLID principle.

```
body: NestedScrollView(
       headerSliverBuilder: (BuildContext context, bool 
innerBoxIsScrolled) {
       return <Widget>[
       SliverAppBar()

       ];

       },
       body: TabBarView(
         controller: tabController,
         children: const [
           ChatList(),
           StatusList(),
           CallList()

         ],
       )
```

So, this is all we need to make Home for Whatsapp UI, which will look like this. You can customize it more based on your needs, like **color, font, icons**, etc.

CHAT

Now, to make chat UI, we have to show the list of users, with whom we have messaged or talked.

- We will make a class named ChatList and we will show data inside it.
- We will return ListView.builder in **build** method and will define different properties of it.

We see a white screen, when we see the list of people in Chat. We will take a container with white color and we will pass **ListTile as a child**.

So, in ListTile, we can define the name, profile, last message, and time. On clicking on any chat, we will open the full chat and will navigate to a new screen. See the code below.

```
ListView.builder(

        itemCount: 20,

        padding: const EdgeInsets.symmetric(vertical: 0),

        itemBuilder: (context, index) {

          return  GestureDetector(

            onTap: (){


Navigator.of(context).push(MaterialPageRoute(builder:
(context)=>ChatScreen()));

            },

            child: Container(

              color: ColorConstants.colorWhite,

              child: ListTile(

                title:
Text("Name",maxLines:1,style:GoogleFonts.archivo(fontSize:
16,color: ColorConstants.colorBlack) ,),

                subtitle:
Text("Hey!",maxLines:1,style:GoogleFonts.archivo(fontSize:
14,color: ColorConstants.color72777A) ,),

                leading:FadeInNetImage( // This is a common
class for images, you can define an icon here.

                    placeholderImage:IMAGE.dummyPic

                ),

                trailing: Column(

                  mainAxisAlignment: MainAxisAlignment.center,

                  children: [
                    Text("2:23
pm",style:GoogleFonts.archivo(fontSize: 14,color:
ColorConstants.color72777A) ,),
                  ],
                ),
              ),
            ),
          );
        });
```

STATUS

This part is where we show the status or story shared by the people. Here are three sections:

1. My status
2. Recent Updates
3. Viewed Status.

As different lists are there, which need to be scrolled. For this, we are going to use CustomScrollView and will pass

- **My Status:**
  For this, you need a Stack widget as you need to show add an icon on the profile picture.

```
  Row(children: [
         Stack(clipBehavior: Clip.none, children: [
           //Profile Pic
           // Add icon
         ]),
         Column(children: [
           Text("My Status"),
           Text("Tap to add status update"),
         ])
       ]),
```

- Recent & Viewed updates:
  As we can see, the UI part is the same for both of these. We can create a common widget and pass it inside the ListView.builder of both updates.

  Write the heading as "Recent updates" and set different properties of ListView.builder. Then you can pass inside statusModule() itemBuilder property. Same way for "Viewed updates".

```
ListView.builder(
                       itemCount: 4,
                       physics: const
NeverScrollableScrollPhysics(),
                       padding: const
EdgeInsets.symmetric(vertical: 0),
                       shrinkWrap: true,
                       itemBuilder: (context, index) {
                         return statusModule();
                       })
```

We see a profile picture along with the name and time. We will make a common widget, statusModule(), and will use it in a different list. You can also pass parameters inside this common widget and pass dynamic data inside Recent and Viewed updates. Like **statusModule({String? profilePic, String? name, String time})** and pass it in place of "Name","Today, 12:20 pm", and "IMAGE.dummyPic".

```
  Widget statusModule() {
    return Column(
      children: [
        Container(
          color: ColorConstants.colorWhite,
          child: ListTile(
            title: Text(
              "Name",
              maxLines: 1,
              style: GoogleFonts.archivo(
                  fontSize: 16, color:
ColorConstants.colorBlack),
            ),
            subtitle: Text(
              "Today, 12:20pm",
              maxLines: 1,
              style: GoogleFonts.archivo(
                  fontSize: 14, color:
ColorConstants.color72777A),
            ),
            leading: FadeInNetImage(placeholderImage:
IMAGE.dummyPic),
          ),
        ),
      ],
    );
  }
```

CALLS

Now going forward, we will learn about making the **"Calls"** section. Here we see 2 parts:

1. Create a link
2. List of people with audio & video calls.

- So to create a link part, we need a Row widget and set Container & Text inside it. To make a tilted icon of the link, we will use Transform.rotate and will pass an angle to it. This way, we are ready with our first part.

```
Container(
         height: 55,
         width: 55,
         decoration: BoxDecoration(
           borderRadius: BorderRadius.circular(28),
           color: ColorConstants.color128C7E,
         ),
         child: Transform.rotate(
             angle: 135 * pi / 180,
             child: const Icon(
               Icons.link_outlined,
               color: Colors.white,
               size: 30,
             )),
       ),
```

- For all the calls received, declined, or dialed, we have to show that list. We will create a ListView.builder and will pass ListTile inside it for showing names, **profile, call, icons**, and **time**.

```
ListView.builder(
             itemCount: 20,
             padding: const EdgeInsets.symmetric(vertical: 0),
             // controller: scrollController1,
             itemBuilder: (context, index) {
              return Container(
                 color: ColorConstants.colorWhite,
                 child: ListTile()
                 );
 }
```

4.3 Analysis of Results

In the scope of the objectives first laid out for the application, the WhatsApp integration has been successful. The WhatsApp plugin functions as intended, and the UI representation is satisfactory, as it reflects very well the original UI and design of WhatsApp.

Some further enhancements to the UI can be made to make it less bulky and more fluid.

4.4 Weaknesses and Strengths of the Application

The things that the application has done right is the general UI representation is very close to that of the original WhatsApp, which will give users a better experience while using the WhatsApp plugin.

However, a key drawback of the current WhatsApp UI implemented is that it cannot be used without obstructing the application completely, which can pose an issue when trying to multitask within the application.

5 | Conclusion

5.1 Conclusion

With WhatsApp being one of the most used messaging apps on the planet for years, it is important to understand its significance in people's social and professional lives. In consequence, the integration of WhatsApp within modern applications such as the one in question in this particular project is a crucial aspect of flutter mobile application development.

To conclude, the objective set at the beginning of the project, which was to integrate a WhatsApp plugin within a flutter application has been achieved, while keeping close to the original UI of WhatsApp to give users a sense of comfortability when using the application.

5.2 Future Developments

- Normal Messaging:
  We can integrate the chat feature in Whatsapp using Flutter. We can use Firebase Cloud Messaging or web sockets to send and receive messages between sender and receiver. You can use firebase_messaging, a Flutter plugin for Firebase Cloud Messaging, a cross-platform messaging solution that lets you reliably deliver messages on Android and iOS. You can also get various docs, blogs, or videos for integrating web sockets in Flutter apps.

- Users List:
  Firebase is very helpful and plays an important role in various features in Whatsapp applications using Flutter. So, we can show different lists of people with chatList, statusList, and callsList using Firebase Firestore Database or using APIs from the backend.

- Media and Gifs:
  We can also try integrating this feature of Whatsapp of sending images, videos, emojis, gifts, stickers, etc. We can use Storage of Firebase for storing these data or some APIs from the backend. For more information, you can check other articles and blogs.

- Multi-platform Support:

You can extend the application's reach by supporting additional platforms like web and desktop. Try to utilize Flutter's platform-specific plugins and adapt the UI to provide a seamless experience across different devices.

- Real-time Updates:
  Implement real-time updates using technologies like Firebase Realtime Database/Firebase CloudFirestore or WebSocket to provide instant delivery of messages, online/offline status updates, and read receipts.

- Localization and Internationalization:
  You can enable localization and internationalization support for users from different regions. You can keep options of multiple languages and the user can select the preferred language and show the application's UI and content accordingly.

- Security Features:
  You can authenticate users using Firebase Authentication or other backend APIs. You can include phone authentication, two-factor authentication (2FA), biometric authentication, and advanced encryption methods to further strengthen the security of user data. On regular intervals, ask the user to enter a PIN to make it attack-free.

- Audio & Video Call:
  This audio and video call feature of Whatsapp is very useful. You can try this feature using different SDKs available like Agora, Videosdk and many others are available. Just explore different SDKs and try to go for any which you can integrate with different platforms.

- Data Backup and Restore:
  Provide users with the ability to backup their chat history, media files, and settings to the cloud. Implement a secure backup solution and allow users to restore their data when switching devices or reinstalling the application.

- Smart Reply and AI Features:
  Also for more advanced features, you can utilize natural language processing (NLP) and machine learning techniques to implement smart reply suggestions, message summarization, or intelligent chatbot capabilities to enhance user convenience and productivity.