POLITECNICO
MILANO 1863

Design and Implementation of Mobile Applications

Design Document

MeetOUT

Sebastiano Piantanida

Professor:
Luciano Baresi.

A.Y. 2023/2024

1. Introduction
    1.1. Purpuse
    1.2. Requirements
2. Application architetture
    2.1. Overview
    2.2. External services
        2.2.1. Firebase authentication
        2.2.2. Tencent IM
        2.2.3. Google maps
        2.2.4. Geocoding
    2.3. Data management
        2.3.1. Firestore
        2.3.2. Firebase database
            2.3.2.1. Users
            2.3.2.2. Events
    2.4. Dependencies
    2.5. Sequence diagrams
        2.5.1. Signin
        2.5.2. Create event
        2.5.3. Search events
        2.5.4. Send a message in chat
3. UI
    3.1. Structure
    3.2. Smarthphone UI
        3.2.1. Sign in and registration page
        3.2.2. Find page
        3.2.3. Event list view
        3.2.4. Create event page
        3.2.5. Event details page
        3.2.6. My profile page
        3.2.7. Profile page
        3.2.8. Chat page
    3.3. Tablet UI
        3.3.1. Sign in and registration page
        3.3.2. Find Page
        3.3.3. My profile page

3.3.4. Create event Page
3.3.5. Chat page
4. Testing
4.1. User testing
4.2. Unit testing
4.3. Widget testing
5. Future developments

1 Introduction

MeetOUT is a social networking platform focused on events, designed to assist users in discovering activities tailored to their interests within their vicinity. The concept stemmed from the challenges often faced in locating local events, particularly when exploring new cities or areas. MeetOUT addresses this need by providing a user-friendly interface for creating and discovering events nearby.

Users can easily create events or join ones created by others, fostering community engagement and social interaction. Each user has a personalized profile where they can see their created and joined events, as well as view their friends' profiles to discover additional events. Furthermore, the application automatically generates group chats for each event, facilitating seamless communication and media sharing among attendees for further event coordination.

In essence, MeetOUT aims to enhance the social experience by simplifying the process of discovering and participating in local events while fostering connections within the community.

1.1 Requirements

The following list contains the requirements that the application should satisfy.

- R1: User should be able to sigin/signup in the application.
- R2: User should be able to logout.
- R3: Registered user should be able to delete their account.
- R4: Registered user should be able to personalize their account with a profile-pic, personal bio, and username.
- R5: Registered user should be able to create a new event.
- R6: Registered user should be able to see events on a map.
- R7: Registered user should be able to see events nearby on a list.
- R8: Registered user should be able to search events at a specific location.
- R9: Registered user should be able to see details (like description, location, date-time, ecc…) of an event.
- R10: Registered user should be able to join an event.
- R11: Registered user should be able to quit an event.
- R12: Registered user should be able to see all the events they have crated and the ones they have joined.

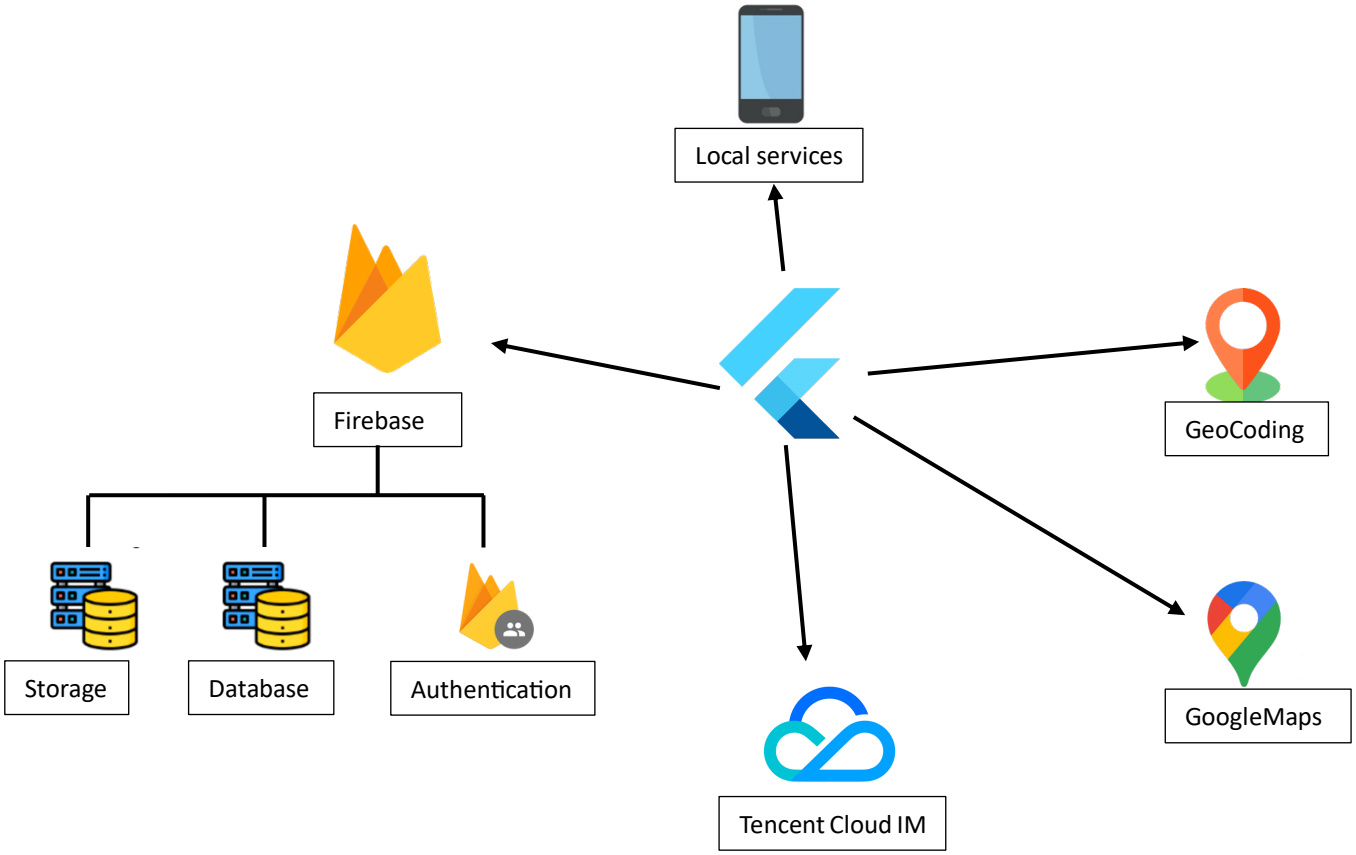
- R13: registered user should be able to access their friend profile.
- R14: Registered user should be able to search other user and add/remove them to the friends list.
- R15: Registered user should be able to exchange message with other participants of an event.

2 Application Architecture

2.1 Overview

To be able to develop a multiplatform application with a single code base, we decide to use the open based Framework Flutter to build `MeetOUT`.

To build the application we relied on different external and local services. To maintain the multiplatform characteristics of flutter we used only services compatible with both IOS and Android platform.
The access to the external services was done, when possible, through asynchronous communication to allow the application to run smoothly even when the services are not responsive.



Internal structure of the application reflects the MVC (Model-View-Controller) architectural pattern, where widgets and pages compose the View. The UserManager and the EventManager act as the Controller, responsible for updating the View based on the content of the Model. The Model, to ensure constantly updated content, is entirely stored on the server side.

2.2 External services

2.2.1 Firebase Authentication

Firebase Authentication provides backend services to authenticate users to our app. It supports authentication using email and password, where user credentials are securely stored upon completion of the registration phase, along with a unique generated ID.

2.2.2 Tencent IM

Tencent IM (Instant Messaging) is a backend service that enables seamless communication among users, facilitating the exchange of messages at no cost. It provides real-time messaging capabilities, allowing users to send text, images, files, and multimedia content instantly. Additionally, Tencent IM enables the delivery of push notifications to users, ensuring that they receive timely updates and messages even when the application is not active.

2.2.3 Google Maps

Google Maps API is a powerful backend service that provides access to and navigation through Google maps. This service allows users to view detailed Google maps, explore points of interest through Google Places, and add custom markers at specific locations.

2.2.4 Geocoding

Geocoding is a service provided by Google that allows the conversion of addresses and place names into geographic coordinates (latitude and longitude), and vice versa. This tool is essential for applications that require mapping of physical addresses or displaying points of interest on an interactive map.

2.3 Data management

The data is stored on a non-relational database located on the server. This database returns structured data in the form of maps, which are then passed to the widgets to display the correct content to the user.

2.3.1 Firebase storage

Firebase Storage is a versatile service provided by Firebase, empowering developers to store a wide range of multimedia files including images, videos, and various other file formats. Within our application, Firebase Storage serves as the foundational repository for housing essential media assets. Specifically, it has been instrumental in securely storing images pertaining to events and user profile pictures, ensuring seamless access and efficient management of these critical visual resources.
The structure of the main folders is the following:

profilePic/
- *<user-id>*
  - *<profilePic-id>.png*
- *<user-id>*
  - *<profilePic-id>.png*
- *….*

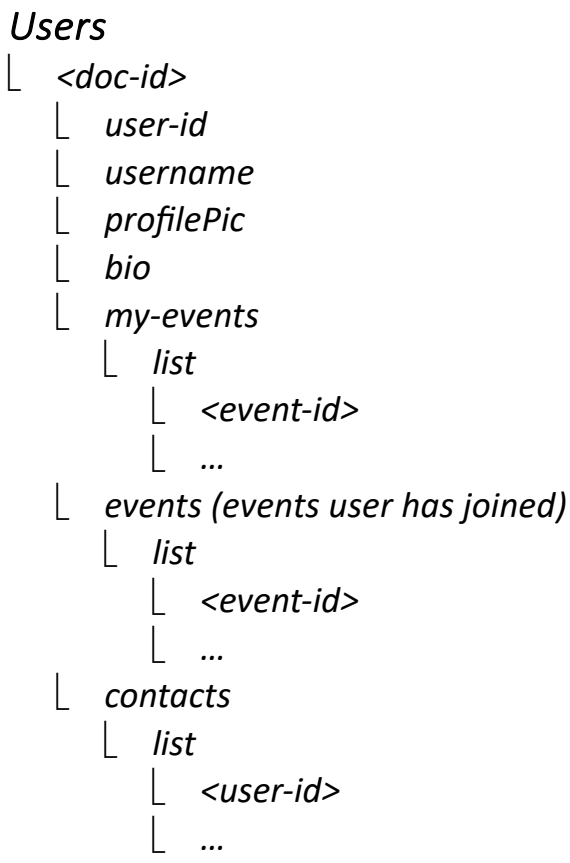
Images/
- *<event-id>*
  - *<image_1>.png*
  - *<image_2>.png*
  - *….*
- *<event-id>*
  - *<image_1>.png*
  - *<image_2>.png*
  - *….*
- *….*

2.3.3 Cloud Firestore

Cloud Firestore enables to store structured data in collections of documents, providing seamless querying, indexing, and real-time updates. It supports various data types and structures, making it ideal for storing diverse datasets like user profiles and event details. Additionally, its integration with Firebase services ensures seamless synchronization and interoperability within the application.

2.3.3.1 Users

This collection allows storing user details, including the username, reference to the profile picture, and biography. It provides a structured organization to efficiently manage information associated with each user within the Firebase database.



2.3.3.2 Events

This collection allows storing event details such as location, name, images, etc. It provides a structured organization to efficiently manage information associated with each event within the Firebase database.

events
- *<doc-id>*
  - *id*
  - *name*
  - *owner*
  - *description*
  - *icon*
  - *date-time*
  - *all-day (boolean to specify if the events has no starting time)*
  - *location*
    - *lat*
    - *lng*
  - *images*
    - *list*
      - *<image-url>*
      - *…*

2.4 Dependencies

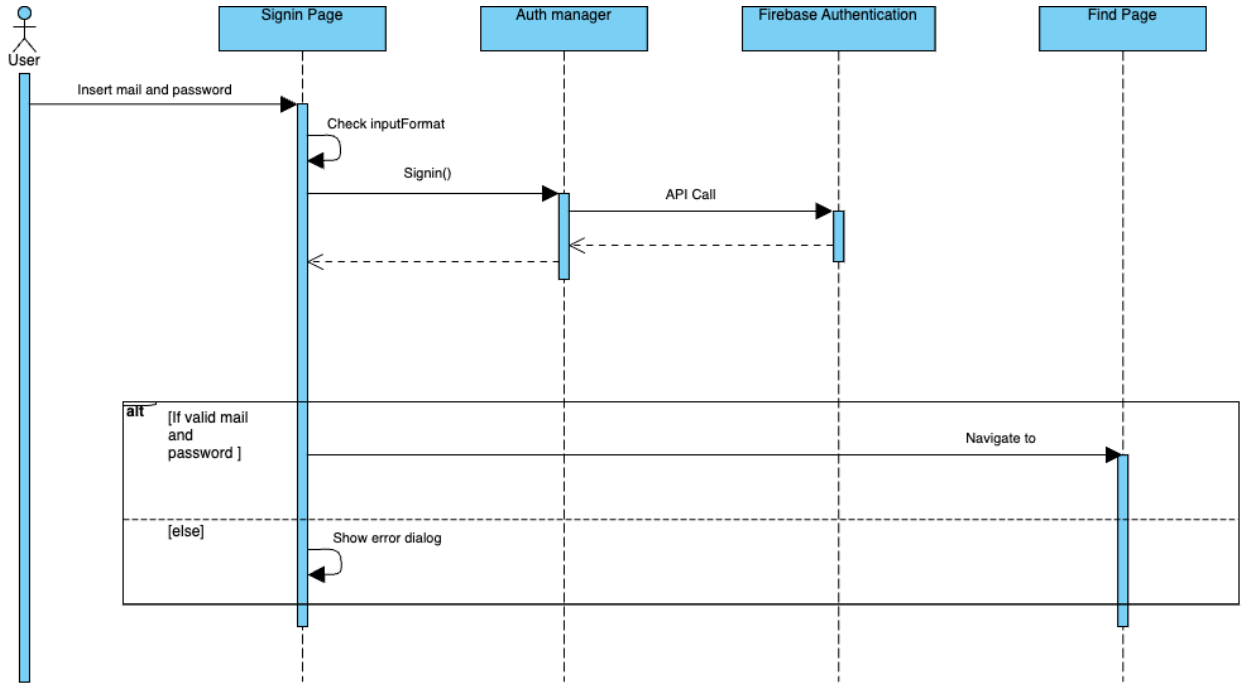
Packages included in the application:

- google_maps_flutter: Allows to add google map.
- geocoding: decode location into addresses ad vice versa.
- firebase_core: allows firebase services to work.
- firebase_auth: used for authentication.
- cloud_firesotre: allow to use Firestore API
- firebase_storage: allow to use
- image_picker: allow to select images from device internal storage

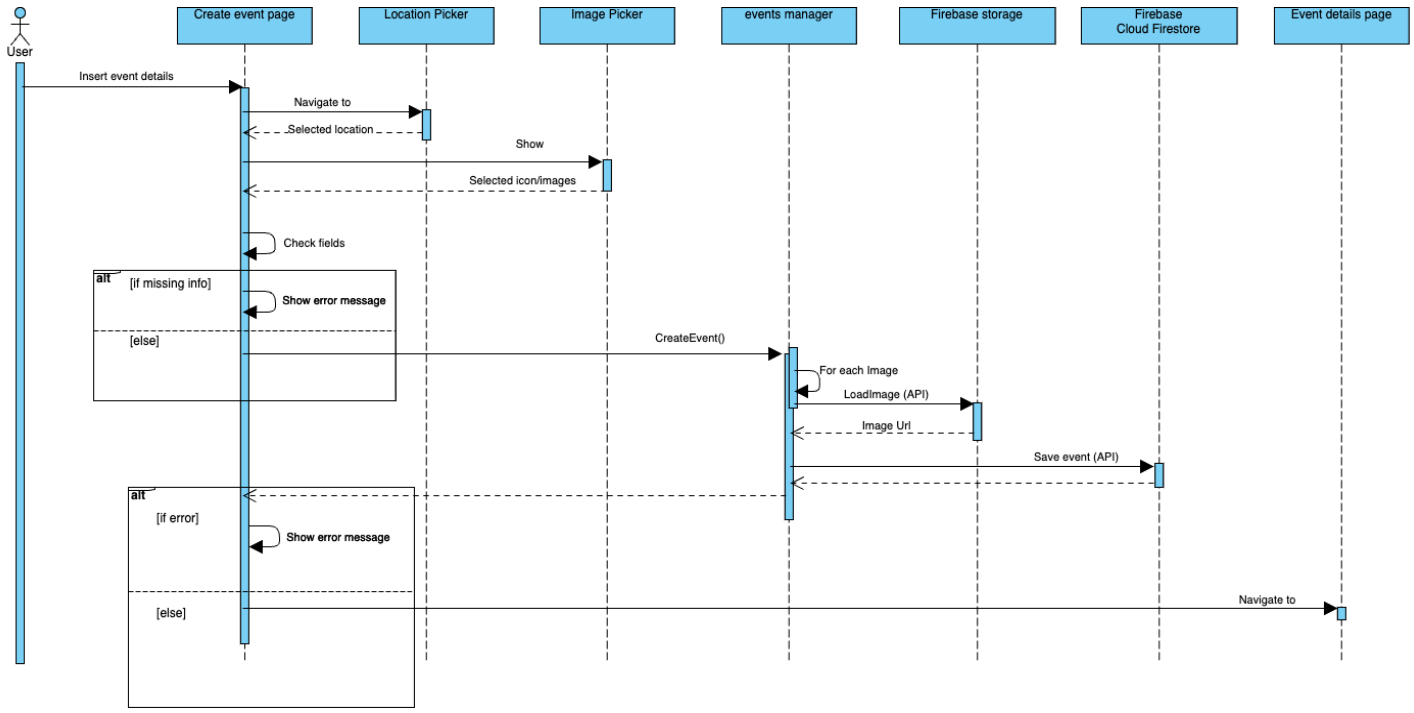
2.5 Sequence diagrams

Sequence diagrams visually represent the interactions and collaborations among various components or objects within our system. They offer clarity regarding the system's behavior, communication flow, and the roles and responsibilities of each element involved. Therefore, the sequence diagram illustrating the key actions of our application is provided for better comprehension.

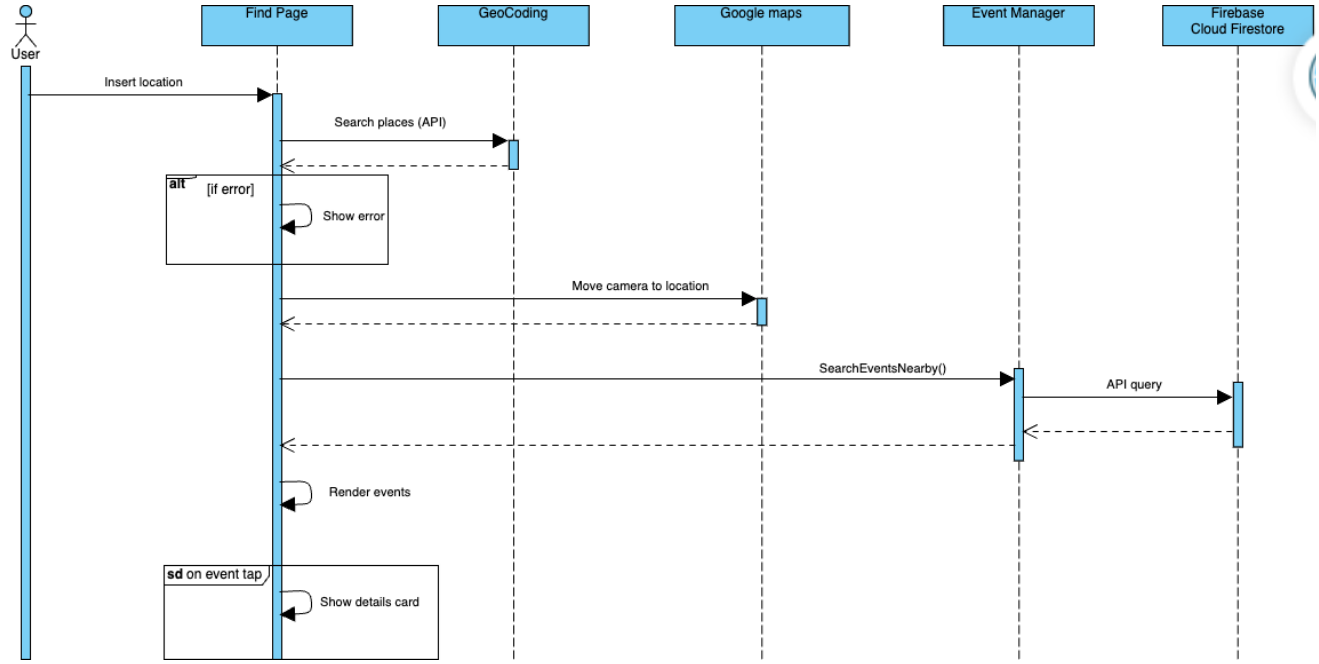
2.5.1 Signin



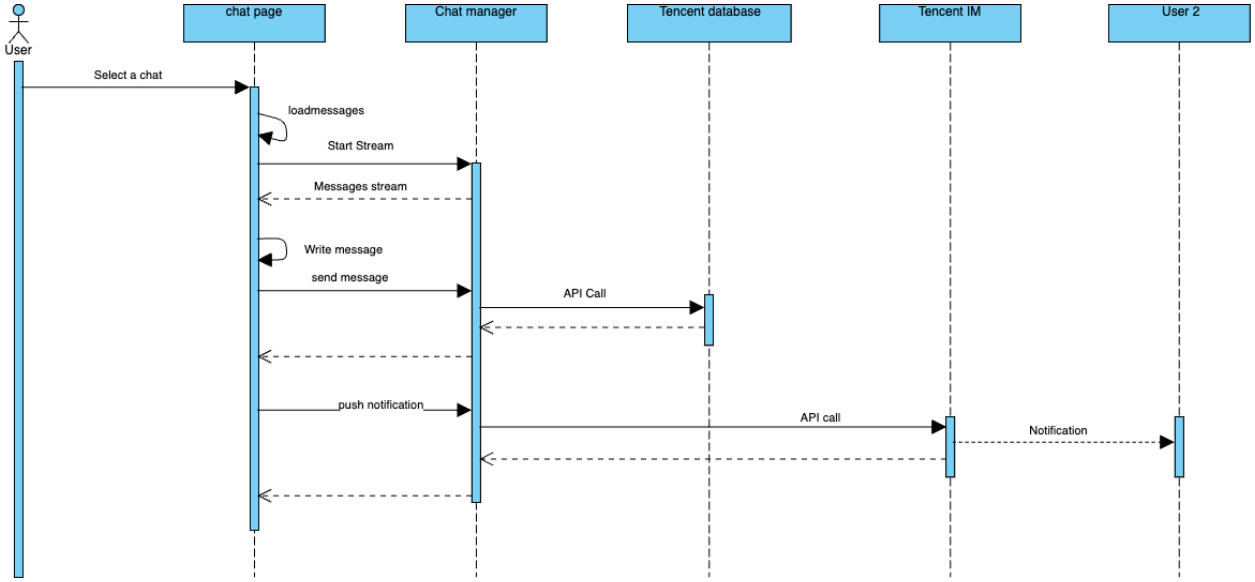
2.5.2 Create event



2.5.3 Search events



2.5.4 Send a message in chat



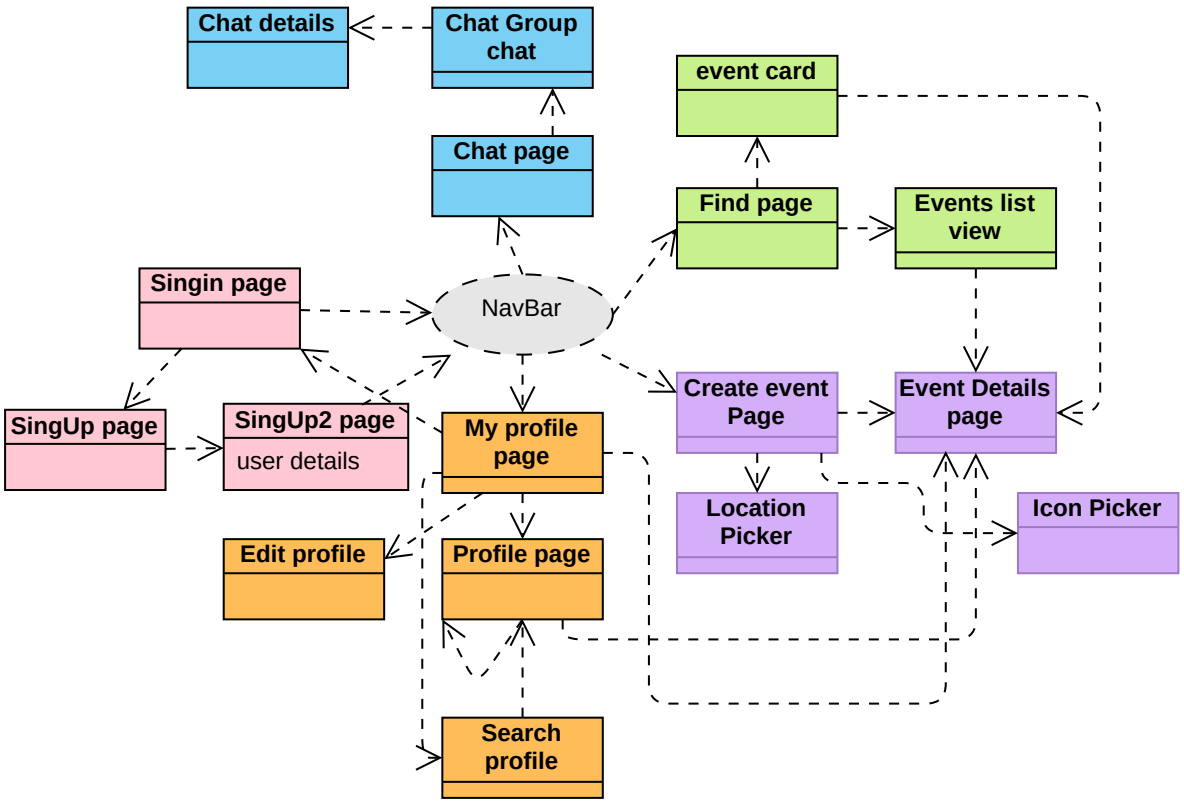
3 UI

This section presents the User Interface for the main pages of the application. The layout is responsive, adjusting according to the device's screen dimensions to optimize user experience. Most screens are showcased in both light and dark themes to illustrate their versatility.

3.1 Structure

To provide a modern and convenient experience, the application has been designed with a navigation bar that allows quick access to the four main sections of the app.

The following diagram illustrates the main interactions and navigations that users can have with the application.

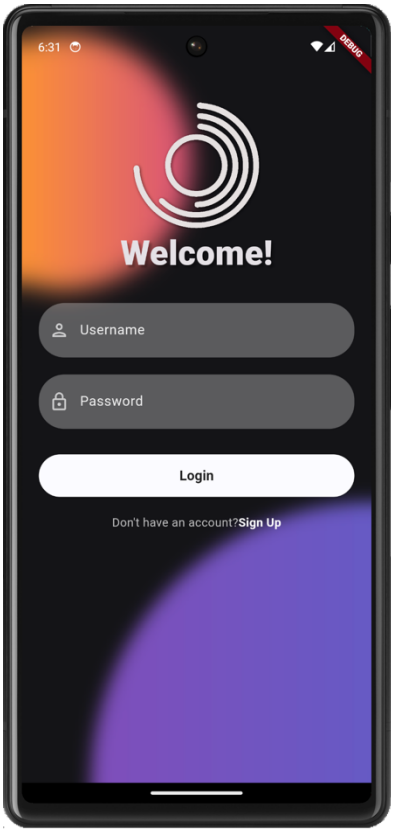


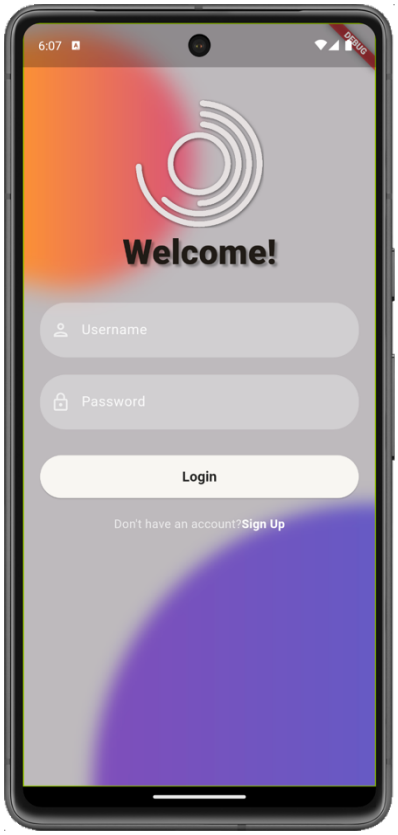
On tablet devices, the interaction is slightly different but still consistent with the diagram.

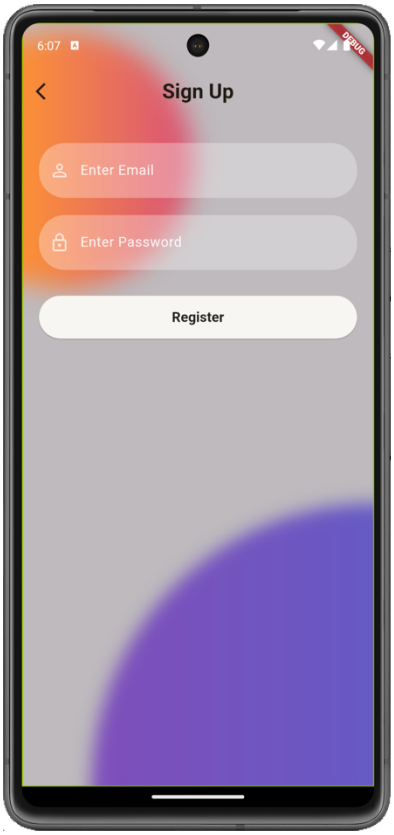
3.2 Smartphone UI

3.2.1 Sign in and registration pages

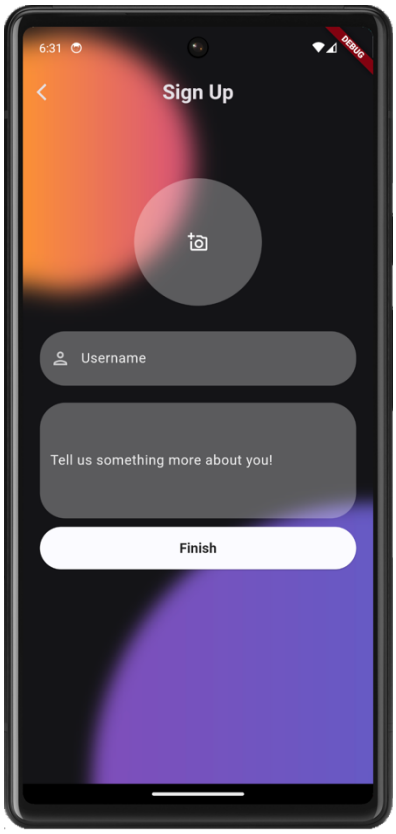
In these screens, registered users can proceed to log in, while non-registered users can proceed to the first step of the registration process. In the registration step, they will be asked to enter an email and a password.

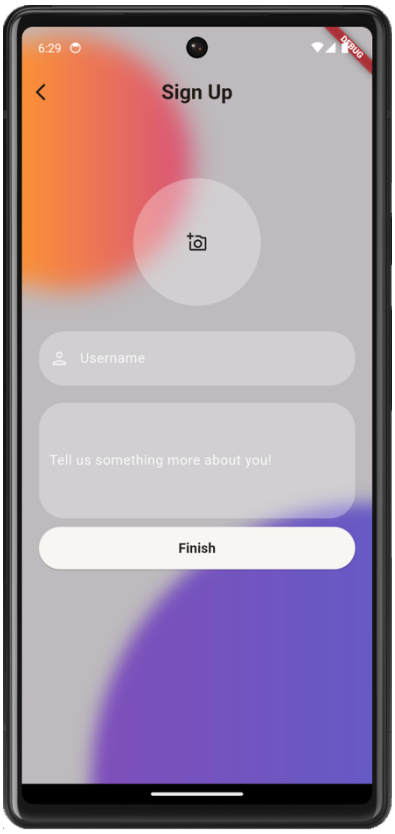






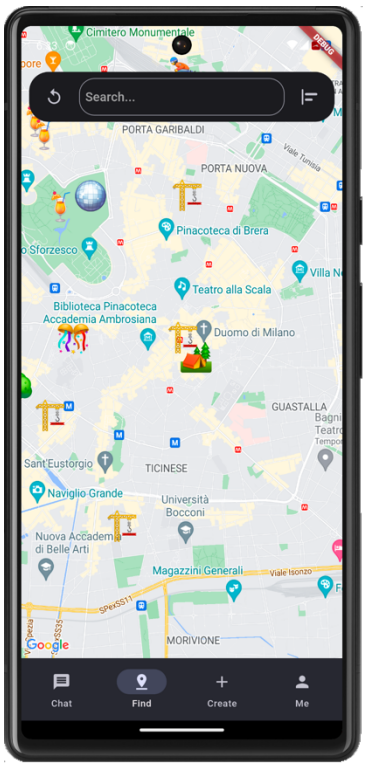
After entering their email and password, users are prompted to enter a username, a profile picture, and write a description of themselves. The latter two are optional.

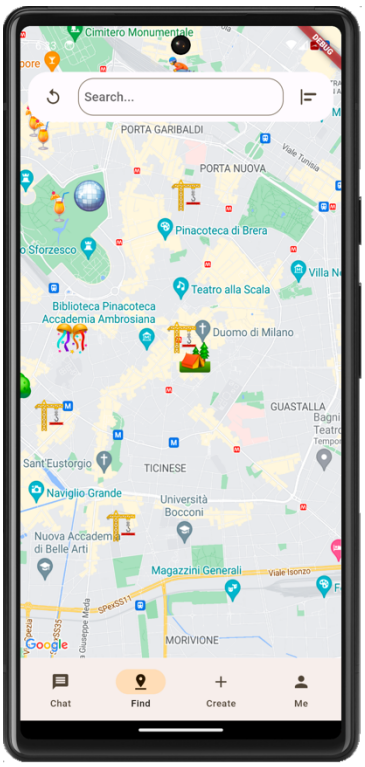




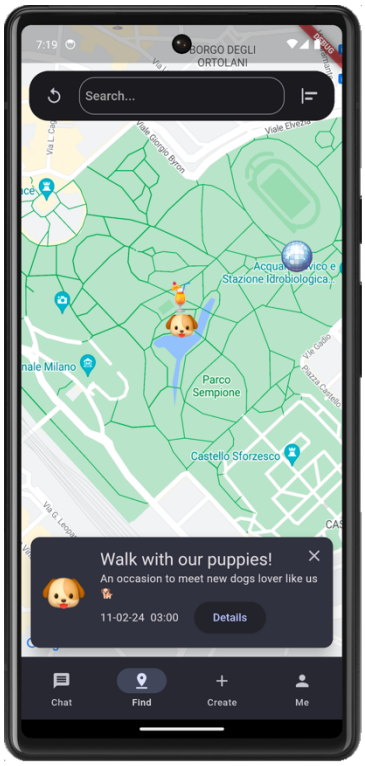
3.2.2 Find Page

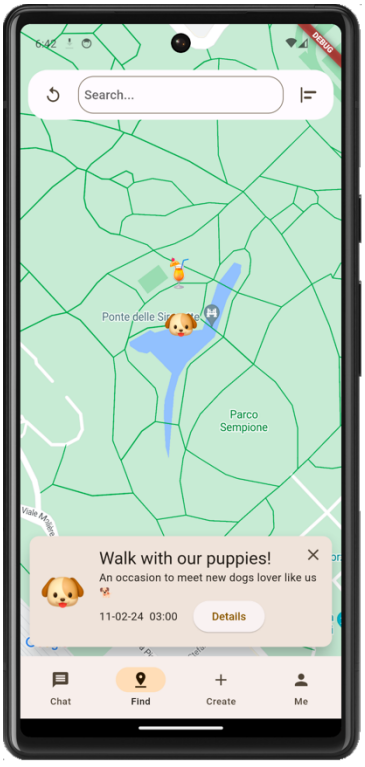
in the find page user can navigate the map to look for events or search for a location





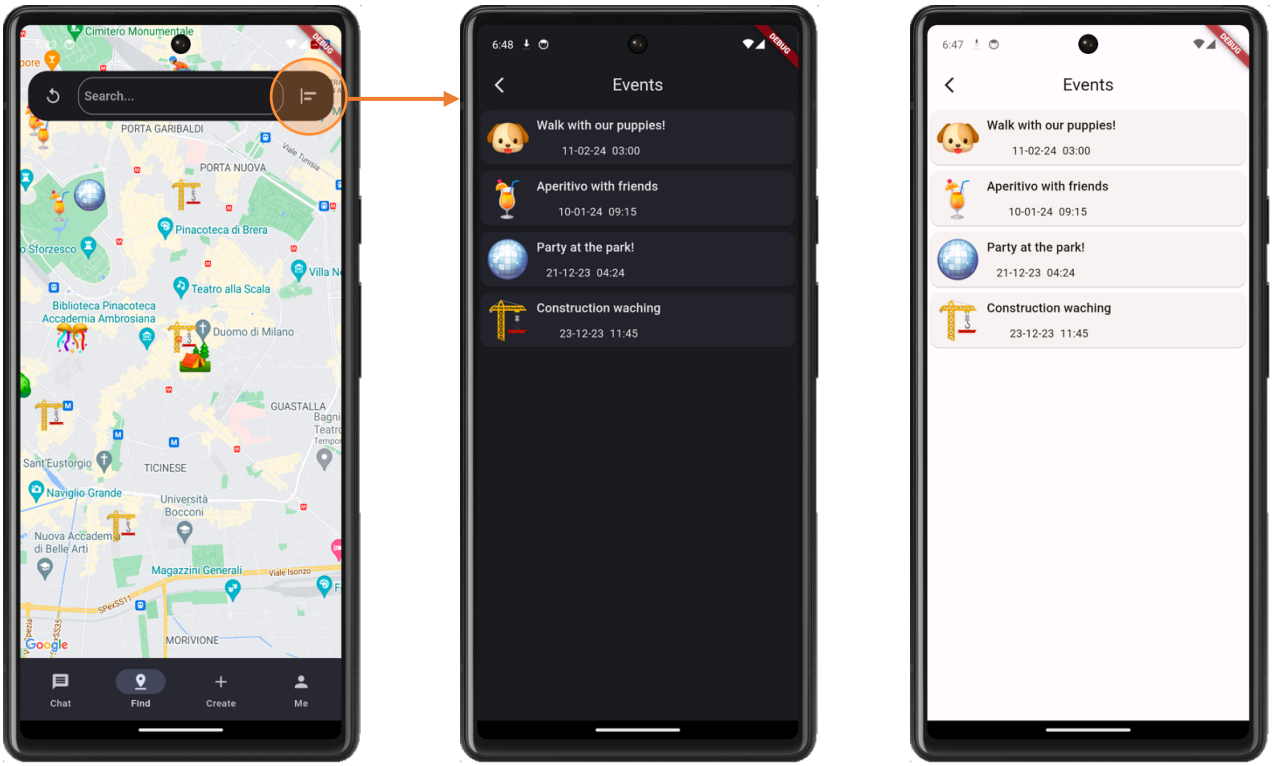
Tapping on an event shows a details card that provides few information about it ad allow to navigate to the details page.





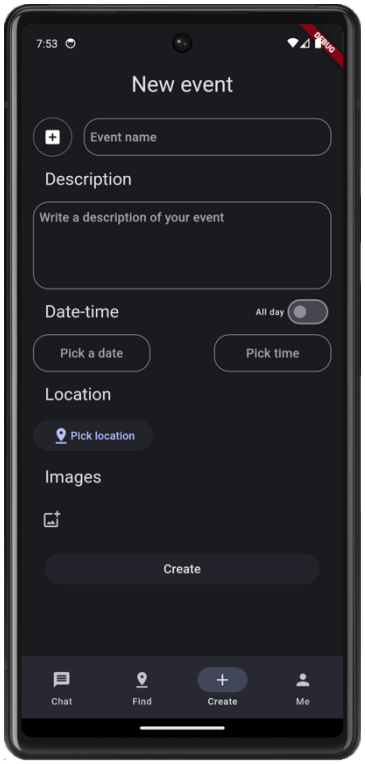
3.2.3 Event list view

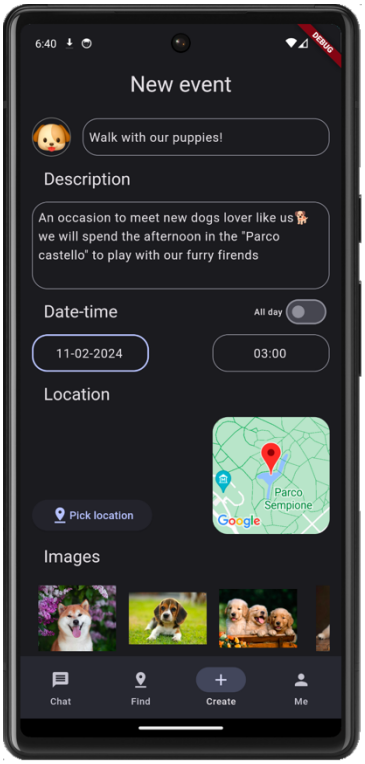
From the Find page, users can navigate to a list view of events by tapping the icon located in the top right corner.

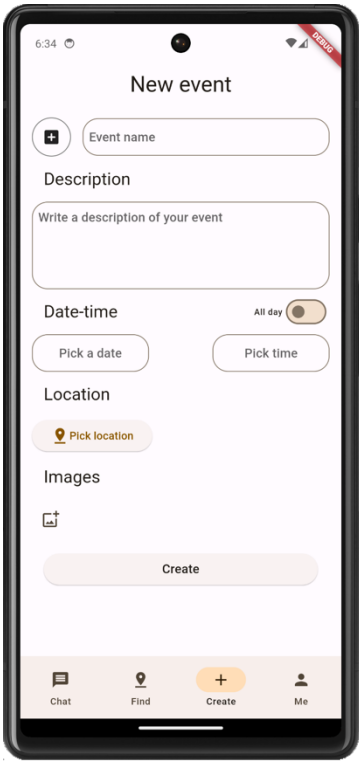


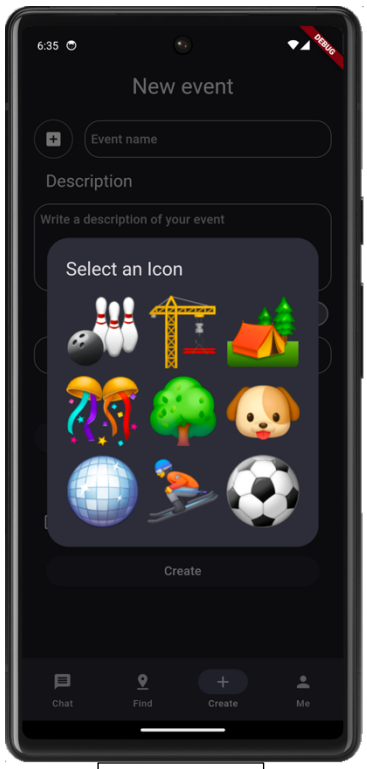
3.2.4 Create event page

This page allows users to create new events by entering an icon, title, description, date, and time, or selecting "All Day" to indicate an event that lasts the entire day. Users can also add images and a location. Tapping the "+" icon opens an icon picker to select the event's icon, which will appear on the map. Pressing "Pick Location" opens a page where users can select a location by long-pressing on the map or searching for a location.

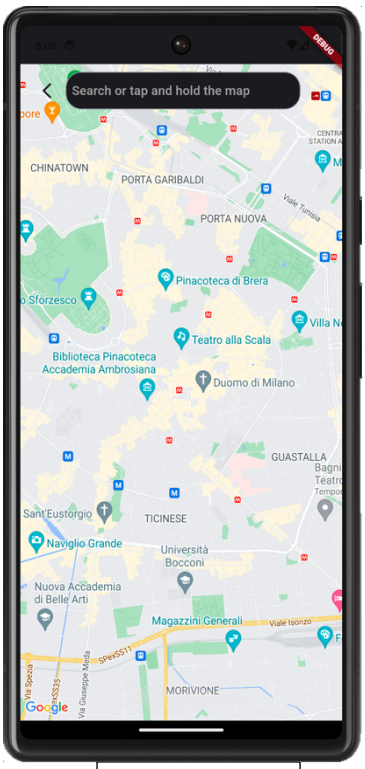








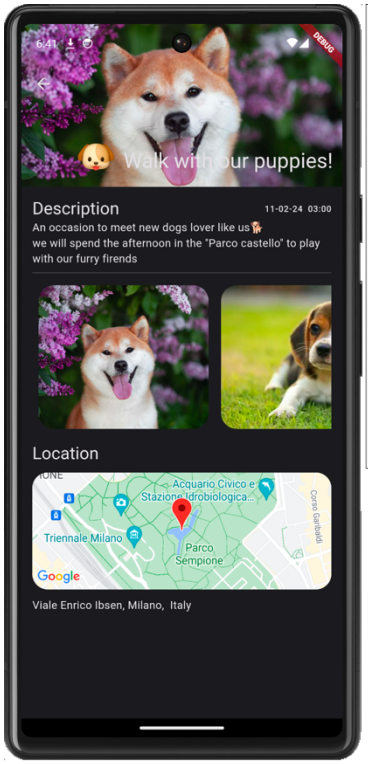
Icon picker

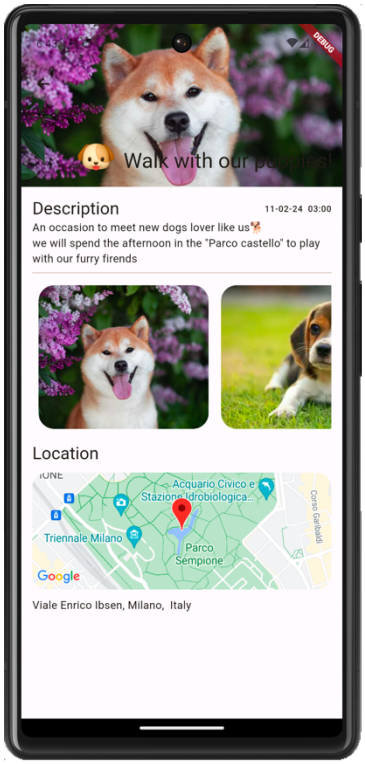


Location picker

3.2.5 Event details page

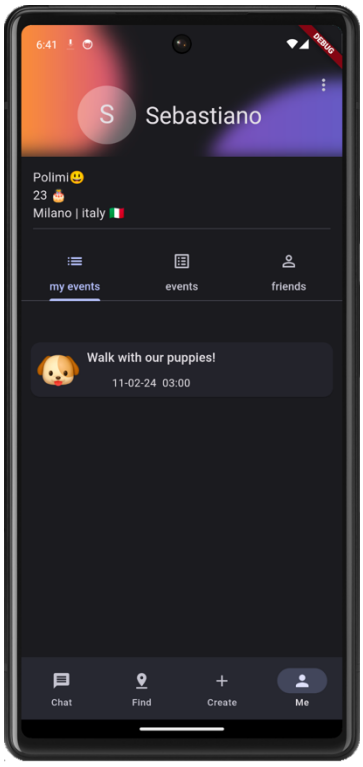
On this page, users can view the complete details of a selected event and potentially join it. They have access to all the information related to the event, including the title, description, date and time, location, and any attached images.

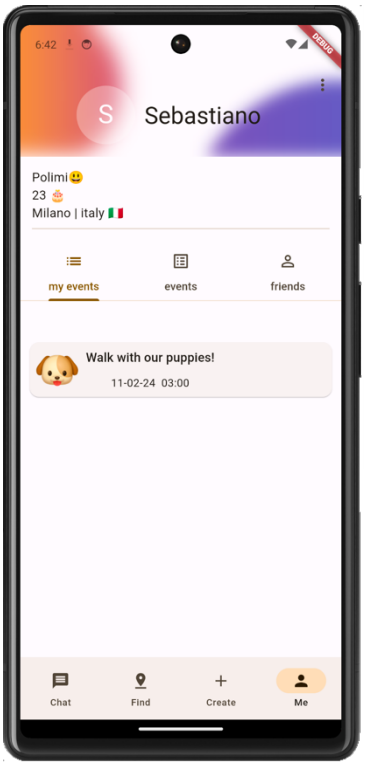




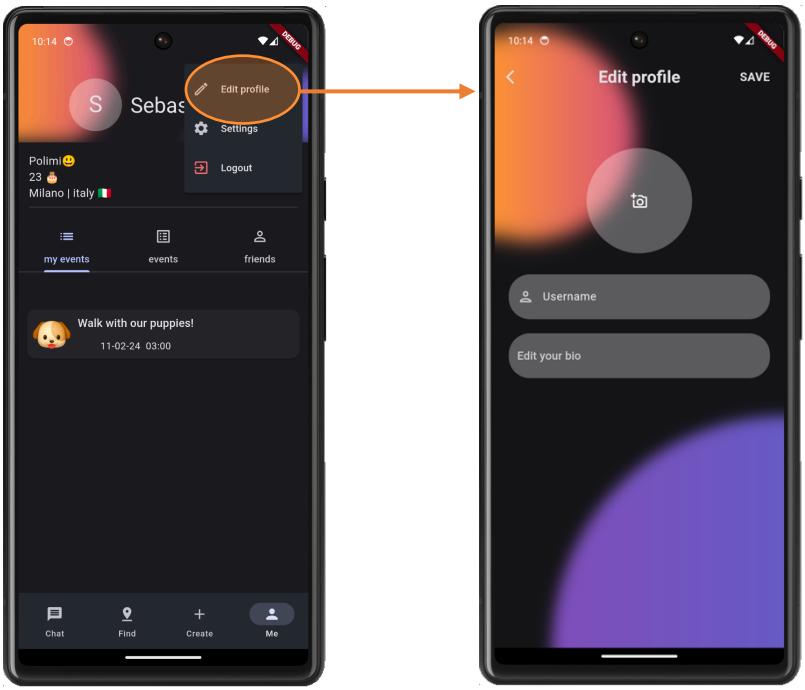
3.2.6 My profile page

On this page, the user can view their profile, access events they have created, view events they have joined, and see or search for their friends.



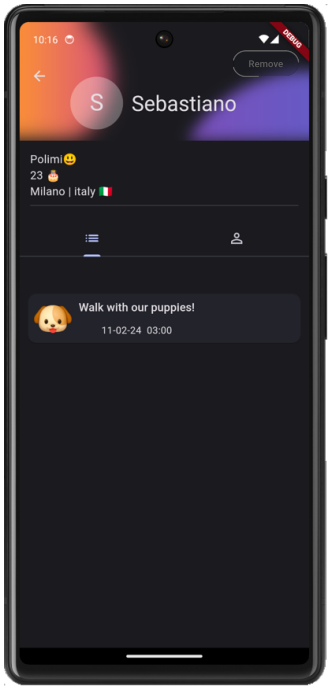


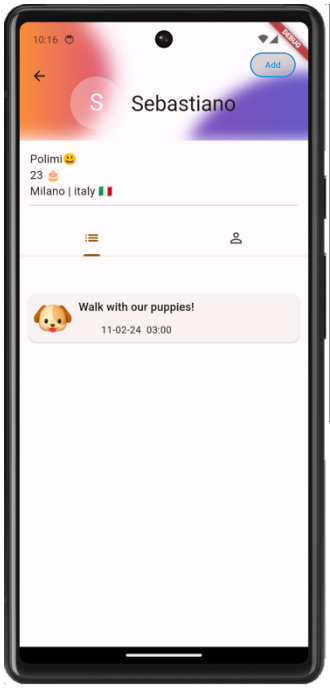
rom their “My Profile page”, a user can easily modify their username, biography, and profile picture by accessing the appropriate section in the dropdown menu



3.2.7 Profile page

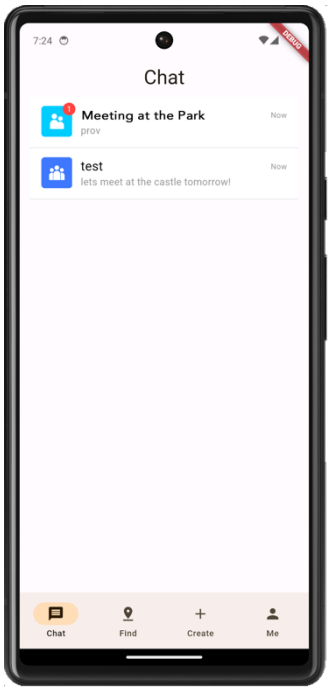
The page that users see when they visit another user's profile is a simplified version of the “My profile page”. It allows them to view only the events created by the user, not those they are attending, and the list of their friends. Additionally, users have the option to add/remove the visited user as a friend.





3.2.8 Chat page

From the chat page, users have the ability to view all event conversations and click on each one to enter the respective chat.



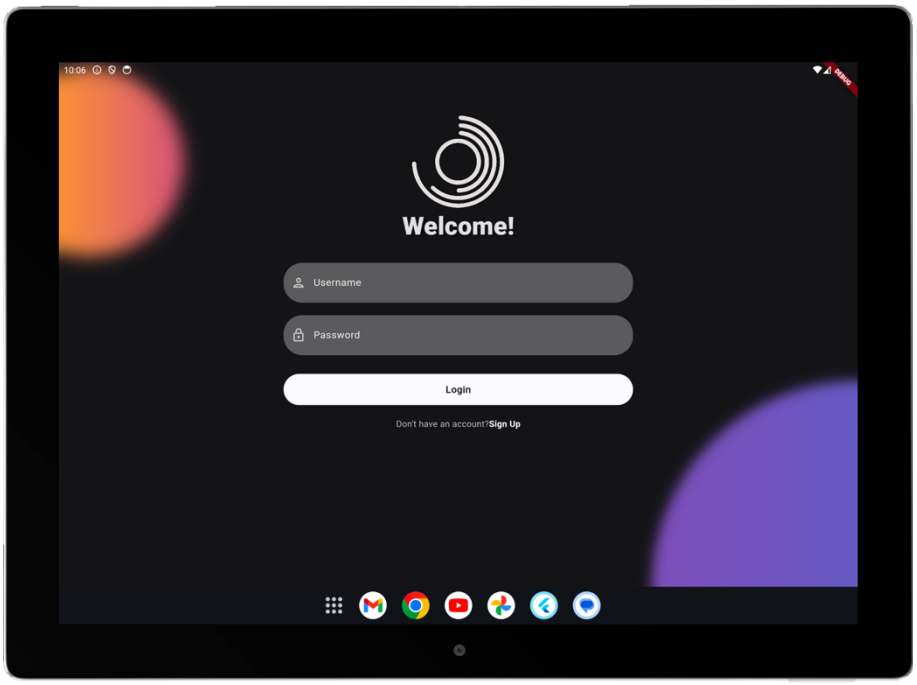
Chat view

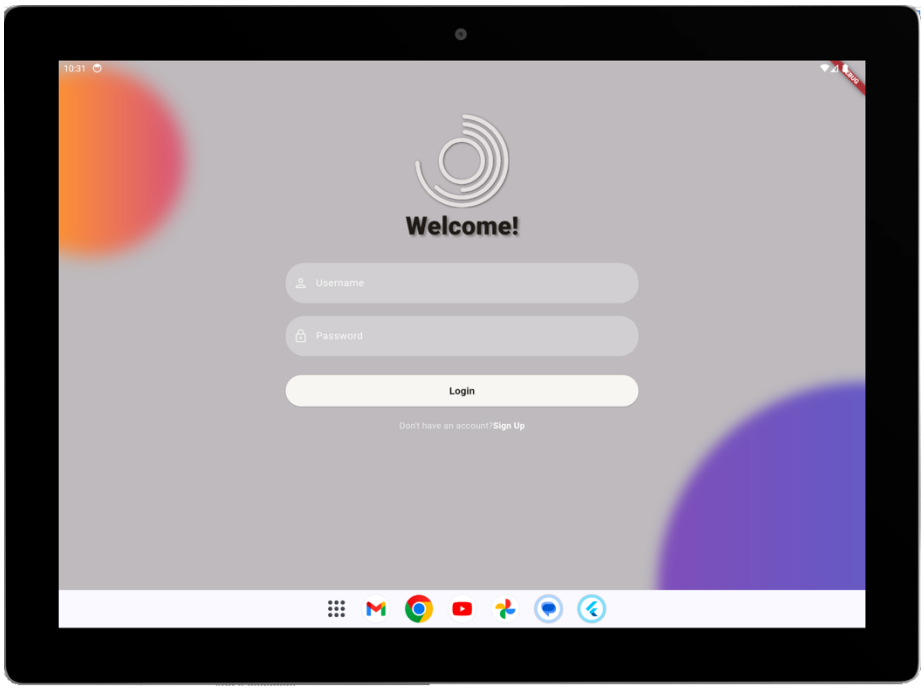
3.3 Tablet UI

The user interface dynamically adapts to the screen dimensions to enhance usability and provide a quality experience regardless of the device used. Some pages undergo significant restructuring, while others are resized to better fit the screen.

3.3.1 Sign in and registration pages

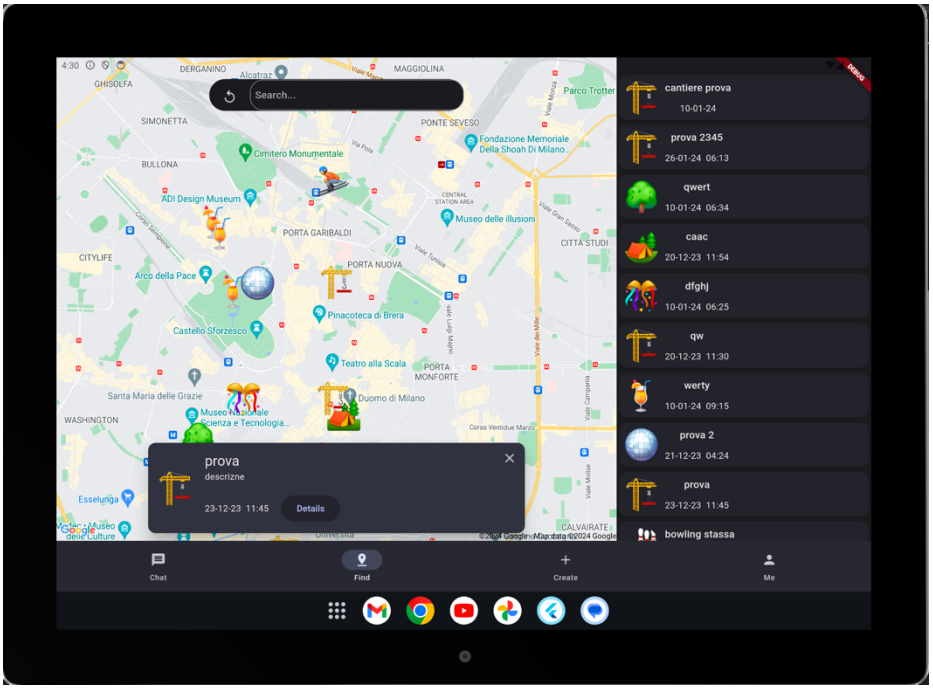
the login page hasn't undergone layout overhauls in the tablet version, we've simply applied limitations to widget sizes to maintain an elegant and functional style.





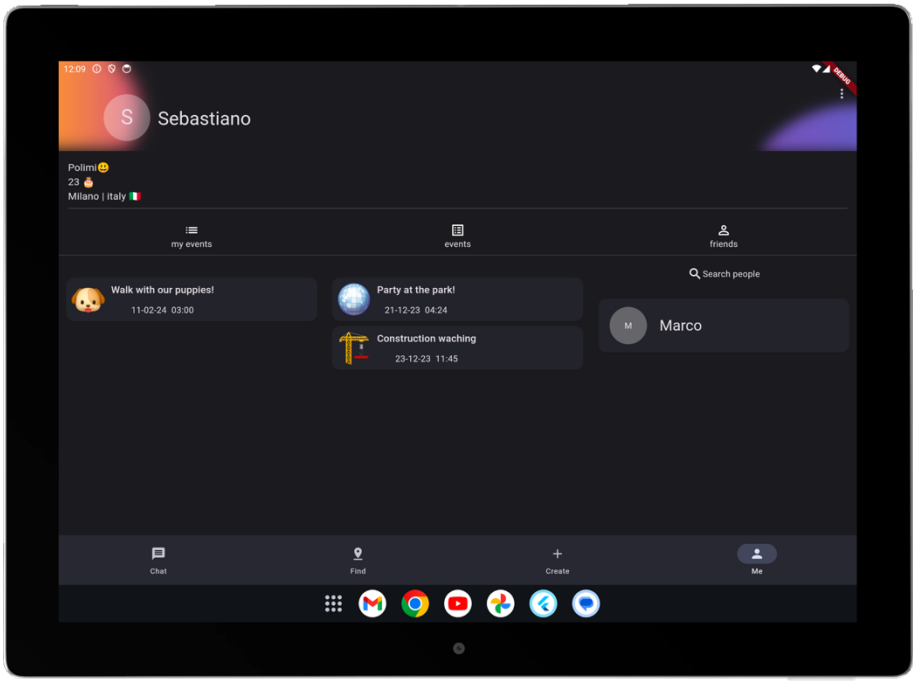
3.3.2 Find Page

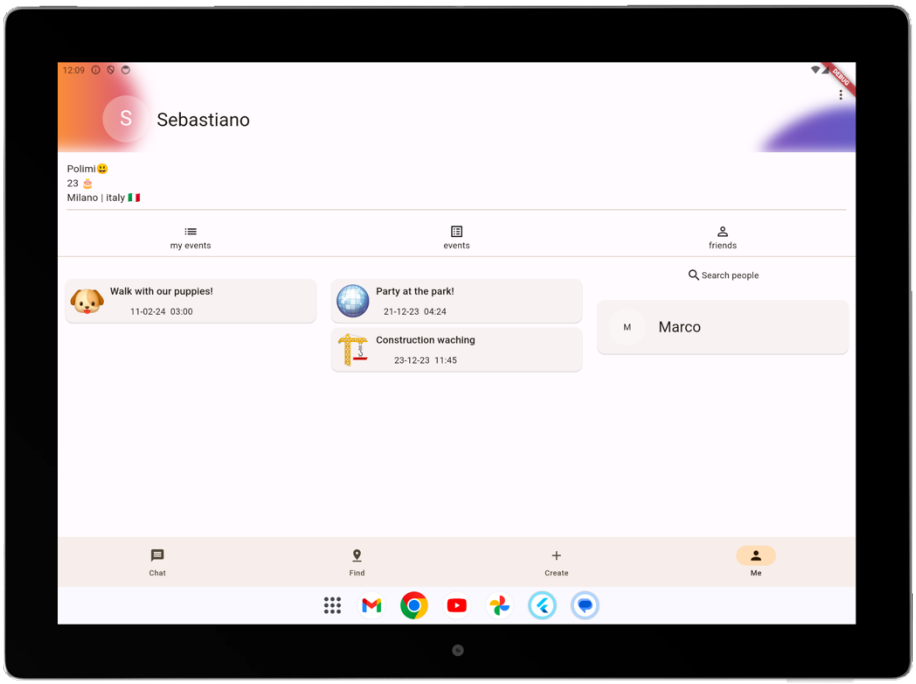
The "find page" has undergone a more substantial restyling. To take advantage of the larger screen dimensions on a tablet, we've added a list view of events alongside the map view. This ensures a clearer and easier understanding of the events available in the area.



3.3.3 My profile page

The profile page has undergone a redesign to enable users to view all tabs simultaneously, facilitating quicker navigation and optimizing screen space utilization. This redesign enhances user experience by providing seamless access to “my events”, “events” and “friends” f without the need for extensive scrolling or switching between tabs.

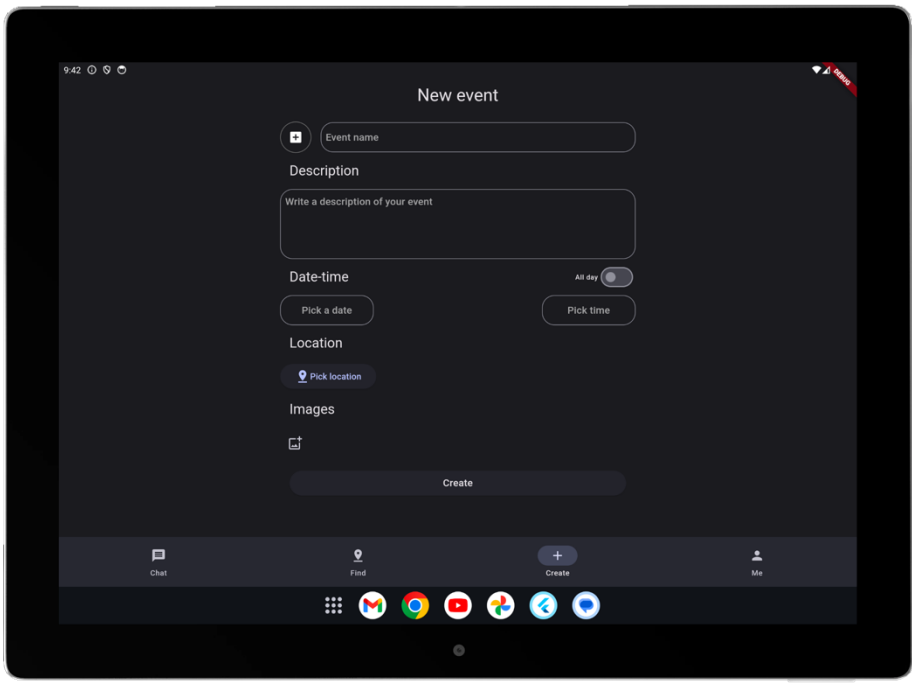




This solution was adopted based on the findings from user testing sessions.

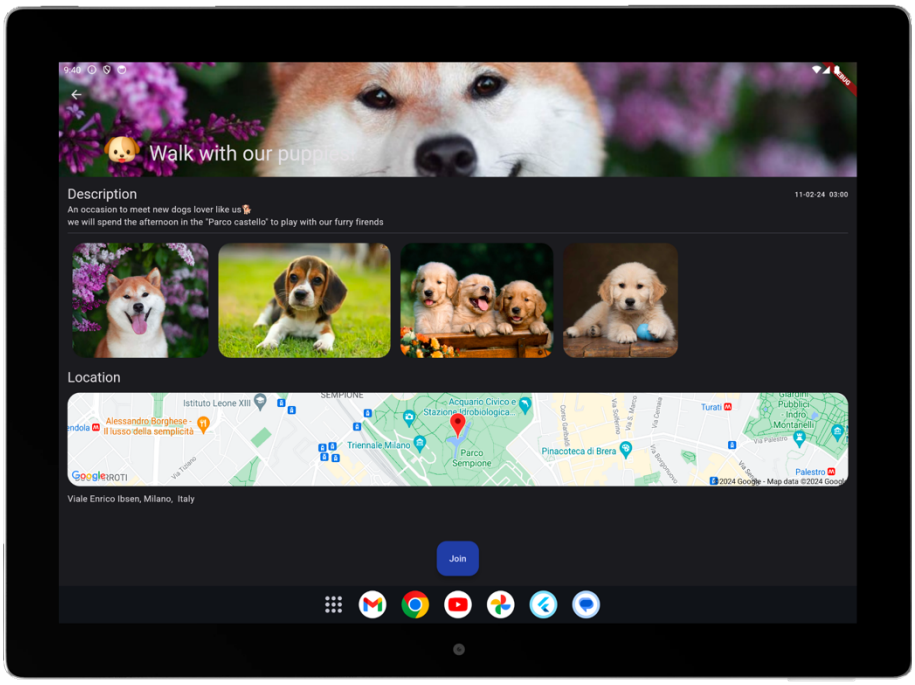
3.3.4 Create event page

This page has undergone minor adjustments to better adapt to the larger screen dimensions.



3.3.4.1 Event details page

This page has remained unchanged to fully utilize the large screen and display more information simultaneously.



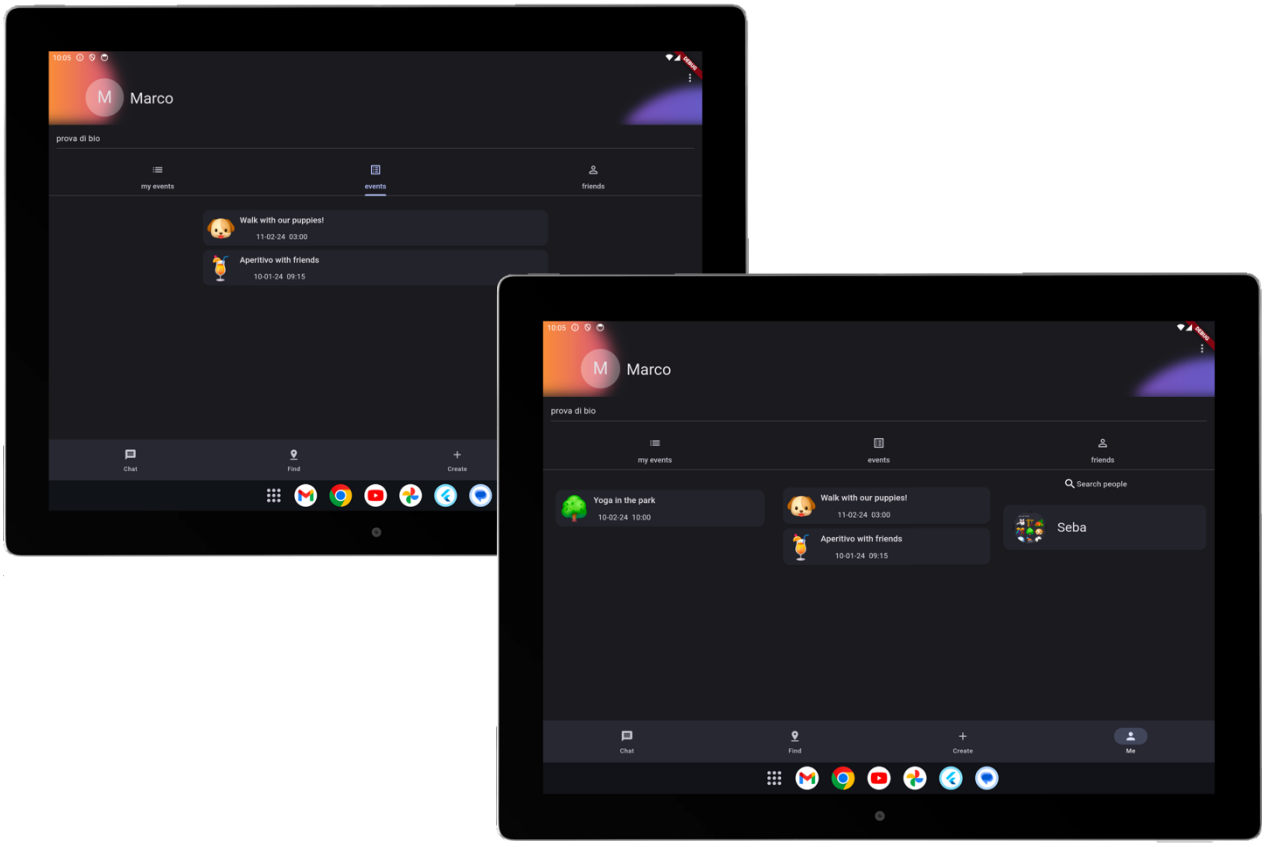
4 Testing

4.1 User testing

This phase of testing was pivotal as it primarily focused on assessing the interface design with individuals not biased by the app's implementation. The tests involved a select group of participants and were conducted in two distinct ways:

1. **Action-based Integration Test:** Participants were tasked with performing actions listed in the Integration Test. This helped evaluate the accessibility of key functions within the application.
2. **Free Navigation:** Participants were given the freedom to explore the app without specific instructions. This method assessed whether the app's design intuitively guided users towards their intended actions.

Due to the engagement of external individuals, this testing approach was the most time-intensive, considering the number of tests performed. However, it yielded high-quality feedback on the user interface. This feedback allowed us to revisit the layout of certain pages, as exemplified by the following instance:



In our initial implementation, the tablet version layout of the profile page mirrored that of the smartphone version, as we believed that displaying three lists side by side might be confusing. By presenting both versions to external individuals, we were able to determine that the first version of our implementation was more comfortable and conveyed the impression of a better-structured application.

4.2 Unit testing

Due to Flutter constraints and our heavy reliance on external services, our app's logic primarily consists of trivial calls to these services with various parameters. As these external services are extensively tested, no meaningful unit tests were deemed necessary.

4.3 Widget testing

Widget testing in Flutter is used to verify the visual and interactive behavior of widgets within the application. Through widget testing, one can simulate user interaction with the app's UI and ensure its correctness and consistency against expectations. These tests enable quick detection of rendering, layout, or widget functionality issues during the app development process.

We focused on testing that pages and the widgets within them rendered correctly on various types of devices. However, due to heavy reliance on external services, some widgets were not extensively tested, particularly those involving Firebase services, due to challenges in mocking certain Firebase services. For this reason, we prioritized user testing, aiming to analyze as many corner cases as possible.

5 Future developments

In the development of MeetOUT, we have successfully implemented all the core functionalities we set out to achieve. However, looking ahead, thanks to the modularity of the architecture, we aim to implement and enhance certain features, focusing more on the backend, which was somewhat neglected to prioritize frontend development.
Some of the functionalities we are interested in introducing include:

Improved Event Search: Implementing a recommendation system to suggest events based on users' interests, enhancing the event discovery experience.
Real-life Social Experience: Introducing the ability to access events directly via QR codes placed in cities or places of interest, adding a more immersive social experience to MeetOUT.
Regarding business integration, we are considering collaborating with event organizing companies such as concert promoters or cinema chains. This collaboration would enable users to purchase tickets directly for the events they choose to attend, facilitating expense sharing among participants. These enhancements align with our vision of making MeetOUT a comprehensive and event-centric social platform, offering a seamless and engaging event discovery and participation experience.