People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of the Requirements for the Degree of

MASTER

Option: Computer Engineering

Title:

NAFFSI, a software application for mental health service management

Presented by:

MESSAI Amine

Supervisor:

Dr. TOUZOUT Walid

Academic Year: 2022/2023

Dedications

This thesis is dedicated to my lovely parents for their endless support and encouragement. I further extend my dedication to all members of Messai's family.

I would like to also extend my heartfelt dedication to all my friends, colleagues, and teachers who have been an invaluable support throughout my journey. Additionally, I wish to express my deepest gratitude to those who have taught, encouraged, and provided guidance to me throughout my entire academic pursuit. This master's thesis is dedicated to each and every one of them, with sincere appreciation and admiration.

Acknowledgments

First and foremost, all praise and thanks giving to Allah the most powerful and most merciful who gave me the ability and patience to accomplish the work presented.

This thesis was carried out in the electronics institute of the University of Boumerdes. I would like to express my sincere thanks to all the university staff who assisted me throughout my masters studies and offered me a very pleasant studying environment.

I would like to also express my heartiest gratitude and appreciation to my respected supervisor Dr. Touzout for his support and patience through out the project, and to all the members of jury for their valuable time and feedback.

Abstract

Mental health issues continue to be a growing concern worldwide, and access to quality online mental health services remains a challenge for many individuals. In response to this pressing need, we present Naffsi, a comprehensive mental health online counseling Flutter Firebase application, which is a service we concluded within our software development company "Mighiss" needed to exist. Naffsi aims to bridge the gap between patients and therapists by providing a platform that facilitates secure and convenient communication through messaging and video calls.

Through Naffsi, patients can connect with professional therapists specialized in various mental health areas. The application implemented a range of features, including live-streams, group therapy sessions, and well-being articles to provide users with knowledge and support, and for therapists to express themselves. By integrating these elements, Naffsi aims to deliver a holistic approach to mental health care, addressing the diverse needs of individuals seeking assistance.

Table of Contents

List of Figures

List of Tables

Abbreviations

UI	User Interface
UX	User Experience
SDK	Software Development Kit
API	Application Programming Interface
FCM	Firebase Cloud Messaging
SQL	Structured Query Language
noSQL	not only SQL
JSON	JavaScript Object Notation
CRUD	Create, Read, Update and Delete
URL	Uniform Resource Locator
UML	Unified Modeling Language

General Introduction

Naffsi is a Flutter Firebase app which core functionality revolves around enabling patients to engage in remote therapy sessions. The app provides convenient and optionally anonymous access to professional therapy services, reducing barriers such as scheduling constraints, and privacy concerns. It also empowers patients to overcome the stigma surrounding mental health. Additionally, the Naffsi app promotes community engagement through livestreams, articles and group therapy sessions. These foster a sense of belonging and create a supportive environment for individuals on their mental health journey.

This platform is a comprehensive online counseling app designed to deliver accessible mental health services. By leveraging widely-used technologies, Naffsi connects patients with therapists, facilitates secure communication, and fosters a supportive community. It has the potential to enhance the delivery of online mental health services, ensuring individuals have access to the support they need, when they need it.

The report is organized as follows: Backgrounds and project overview, this chapter introduces the project's context, including the problem statement, goals, and motivations. Tools and technologies, this chapter discusses the chosen tools, technologies, and software components used in the project's development. System design, this chapter presents the high-level architecture and interaction of system components. and then implementation, which covers the detailed process of translating the system design into a working prototype.

Chapter 1

Backgrounds and project overview

1.1 Introduction

In today's increasingly digital and interconnected age, providing access to health services across multiple platforms has become increasingly important. Cross-platform development offers a powerful solution, enabling developers to create applications that can seamlessly run on various operating systems and devices. Google's Flutter framework presents a perfect example that has gained tremendous popularity.

Relevant to this project report, Flutter will be the main building framework for the Naffsi platform, and will assist in reaching the service's desired objectives and ambitions.

1.2 Subject Presentation

In today's world, mental health is gaining increasing recognition as a vital overall well-being aspect [1], especially after the COVID-19 pandemic [2].

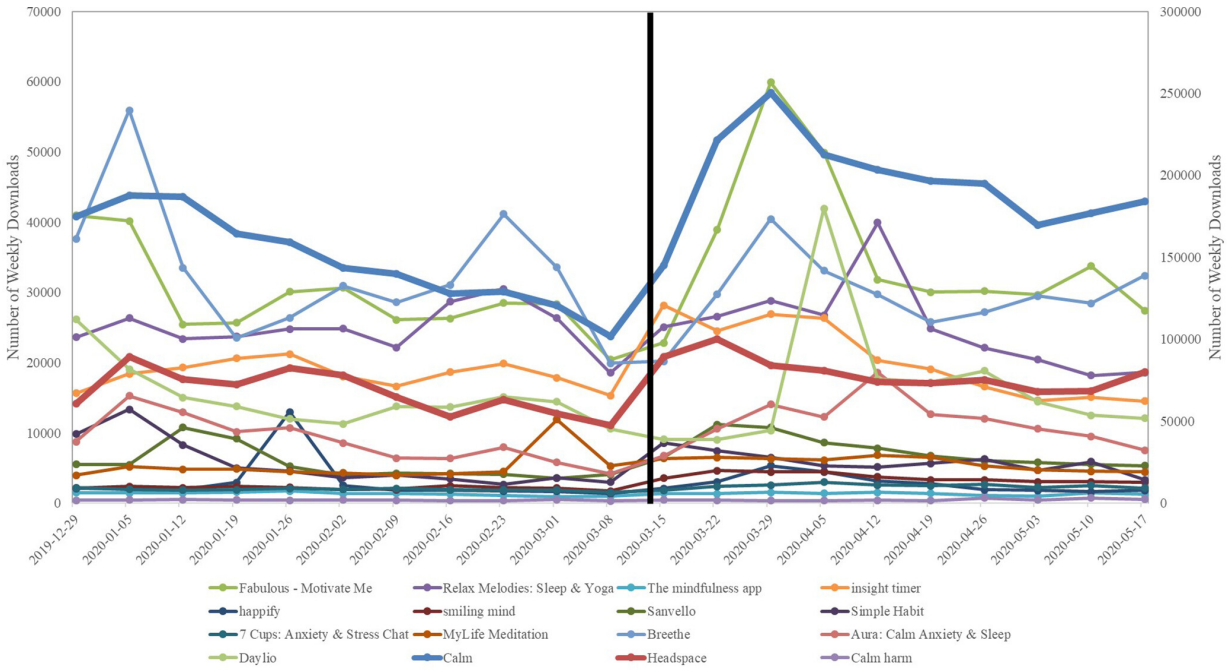

Figure 1 – *Number of weekly downloads for 16 mental health apps*

However, there are still significant barriers that prevent individuals from seeking help. Stigma[3], privacy concerns[4] [5], limited access to qualified professionals, and geographical constraints are just a few of the challenges that limit individuals from accessing mental health services. To address these barriers, we have developed the Naffsi app; an innovative online counseling platform designed to provide accessible and optionally anonymous mental health support.

1.2.1 Problem Statement

The Naffsi app aims to address the pressing need for accessible online mental health counseling, and even hopefully help individuals transition from online to traditional counseling methods, which are often not sought out due to long waiting times, limited availability of therapists, and social stigma associated with seeking help. Moreover, geographic barriers can make it challenging for individuals in remote areas to access mental health services. These factors contribute to the under-utilization of mental health resources and hinder individuals' well-being.

1.2.2 Objectives

The objectives of the Naffsi app can be summarized as follows:

- Provide a convenient and accessible mental health service.
- Remove barriers to seeking help by providing optional anonymity to its users.
- Facilitate communication between therapists and patients through messaging and video calls.
- Foster a supportive community dedicated to demonstrate the importance of mental health through live-streams, well-being articles, and group therapy sessions.

1.3 Cross-platform applications

Cross-platform applications refer to software that is designed to run seamlessly on multiple types of computers and operating systems [6] [7]. These applications provide users with the flexibility to use the software on different platforms without the need for separate versions or adaptation. The term "cross-platform" or "multi-platform" signifies the ability of software to work efficiently across different platforms, enhancing accessibility and usability for users.

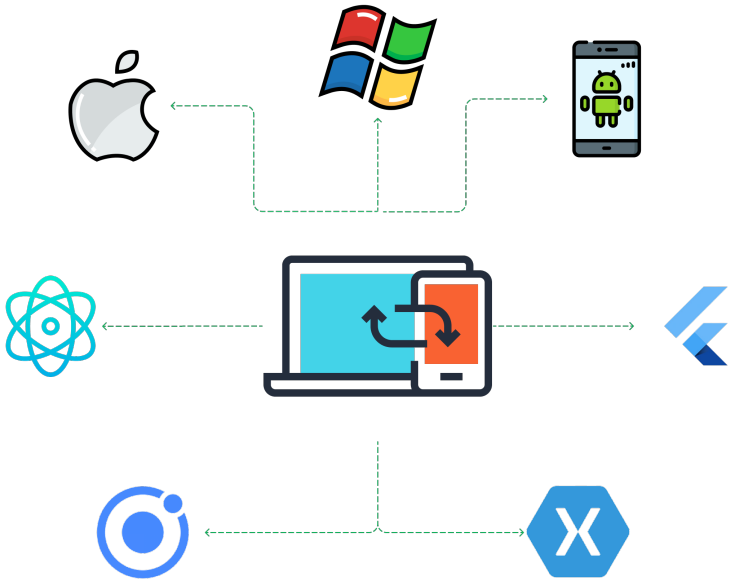

Figure 2 – *Cross platform apps diagram*

1.4 Conclusion

The backgrounds and project overview chapter introduces the concept of cross-platform development and highlights the need for a solution to conveniently accessing mental health supports. This chapter sets the stage for the subsequent chapters, where we will delve into Naffsi's development technologies used, implementation details and design considerations.

Chapter 2

Tools and technologies

2.1 Introduction

In this chapter, we delve into the tools, technologies, programming language, and frameworks employed to develop the Naffsi application. We explore their role and importance in the project's implementation. By examining these elements, readers gain valuable insights into the technical foundation of Naffsi.

2.2 Development Tools

2.2.1 Visual Studio Code

Visual Studio Code, often referred to as VS Code, is a versatile lightweight source-code editor developed by Microsoft. The editor boasts a rich set of features, including robust debugging capabilities, efficient syntax highlighting, intelligent code completion, convenient code snippets, seamless code refactoring, and integrated Git support. With its extensive range of tools, functionalities and plugins ecosystem, Visual Studio Code offers developers a highly productive and efficient coding environment[8]. It was chosen for this project specifically for its perfect adaptability with the Flutter framework development process.

2.2.2 Firebase console

Firebase Console is a powerful web-based tool provided by Google for managing various aspects of Firebase services including most importantly Firebase Auth, Firestore, Storage, and Cloud Messaging. It offers a user-friendly interface that simplifies Firebase projects' configuration and monitoring [9].

2.3 Dart programming Language

Dart is a versatile programming language designed to create high-performance applications across various platforms. It aims to provide developers with a productive and efficient language for multi-platform development, while also offering a flexible run-time platform for app frameworks. By leveraging Dart, developers can build fast and responsive applications that run seamlessly on different devices and operating systems [10] [11]. Dart was chosen for this project due its object-oriented nature, its simple UI syntax and widget structuring, and its asynchronous programming function that is helpful back-end wise in the context of fetching and handling data with databases.

2.4 Frameworks

2.4.1 Flutter

Flutter is a versatile UI toolkit developed by Google for creating applications that run seamlessly on mobile, web, and desktop platforms. With Flutter, developers can write code once and deploy it across multiple platforms, saving time and effort. It is widely adopted by developers and organizations worldwide and is an open-source framework. Flutter simplifies app development, reduces costs, and provides a fast development cycle with rich UI components, along with a wide range of community packages[12]. It enables engineering managers and businesses to unify their develop-

ment teams and build branded apps for various platforms from a single code-base, benefiting the entire customer base[13].

2.4.2 Firebase

Firebase is a comprehensive suite of cloud computing services and development platforms offered by Google. It provides a wide range of functionalities such as database hosting, authentication, real-time capabilities and integration for various applications. Firebase supports multiple platforms including Android, iOS, JavaScript, Node.js, Java, Unity, PHP, and C++, making it a versatile choice for developers[9], especially for this project, as it integrates seamlessly with Flutter.

2.4.3 VideoSDK

VideoSDk is a real-time audio-video SDKs collection which offer developers complete flexibility, scalability, and control to seamlessly integrate audio-video conferencing and live streaming capabilities into web and mobile applications. These SDKs provide the necessary tools and resources to enable real-time communication and collaboration, empowering developers to create immersive and interactive experiences for their users [14]. Additionally, this framework allows call monitoring and session recording .

2.5 Packages

2.5.1 Flutter-Facebook-Auth

The Flutter plugin for Facebook authentication simplifies seamless integration Facebook authentication into Flutter apps. This plugin is also compatible with web platforms, ensuring consistent authentication experiences across different devices [15].

2.5.2 Google-Sign-In

The Flutter plugin for Google Sign-In enables secure authentication using Google accounts in Flutter apps. It allows users to sign in with their Google credentials on both Android and iOS platforms. This plugin provides a seamless integration with Google's authentication system, ensuring a reliable and convenient sign-in experience for users [16].

2.5.3 Firebase-Auth

Firebase Authentication simplifies secure authentication system development and enhances the user sign-in and onboarding experience. It offers a comprehensive identity solution, including support for email and password accounts, phone authentication, and login options like Google, Twitter, Facebook, and GitHub [17].

2.5.4 Firebase-Core

The Flutter plugin for Firebase Core empowers developers to establish connections with multiple Firebase apps [18].

2.5.5 Cloud-Firestore

The Flutter plugin for Cloud Firestore allows seamless integration of the cloud-hosted, NoSQL database into Android and iOS applications, offering real-time synchronization and offline capabilities [19].

2.5.6 Firebase-Storage

The Flutter plugin for Firebase Cloud Storage enables easy integration of the powerful and cost-effective object storage service into Android and iOS applications. It provides a simple and efficient way to handle storage of files and data [20].

2.5.7 Firebase-Messaging

The Flutter plugin for Firebase Cloud Messaging allows seamless integration of the cross-platform messaging solution into Android and iOS applications. It provides a reliable and efficient way to deliver messages and notifications to users on both platforms [21].

2.5.8 Flutter-Calendar-Carousel

The Calendar Carousel widget for Flutter is a versatile tool that allows customizable calendar widgets creation. With this widget, navigation through dates is simple. It provides flexibility to add styling for each day, providing the ability to personalize calendars [22].

2.5.9 Image-Picker

The Flutter image picker plugin facilitates image selection from the device's library and capturing new pictures using the camera. It offers a convenient solution for integrating image picking and capture functionality into Flutter apps, enhancing the user experience when managing and manipulating images [23].

2.5.10 Fluttertoast

The FlutterToast library is a Flutter plugin that provides a simple and convenient way to create toast messages in a Flutter application with just a single line of code. In the context of mobile app development, a toast is a small, unobtrusive message that appears briefly on the screen to provide information or notifications to the user. Toast messages typically appear at the bottom of the screen and disappear after a short duration, allowing users to quickly acknowledge the message without interrupting their workflow [24]. With the FlutterToast library, developers can easily incorporate toast messages into their Flutter apps, improving the user experience and enhancing the communication of important information to the users [25].

2.5.11 Flutter-Local-Notifications

The Local Notifications plugin for Flutter offers a cross-platform solution for displaying and scheduling local notifications in mobile applications. It allows for customization based on the target platform, enabling enhanced user engagement and interaction [26].

2.5.12 VideoSDK package

The VideoSDK package simplifies the integration of Audio and Video Calling API and Live Video Streaming API into Flutter apps. It provides developers with an easy way to add audio and video communication features with minimal code [27].

2.6 Conclusion

In this chapter, we discussed the tools and technologies used in the development of the Naffsi app. This included Flutter for cross-platform development, Firebase for backend services, and various plugins like Firebase Messaging and VideoSDK. These tools greatly contributed to the functionality and effectiveness of the app.

Chapter 3

System design

3.1 Introduction

The system design phase is a crucial aspect of developing the Naffsi application as it enables a comprehensive understanding of the system and its behavior. In this chapter, we will introduce the Unified Modeling Language (UML) and its various diagram types that are utilized in the design of the Naffsi application. These diagrams serve as valuable tools for visualizing and planning the structure, interactions, and components of the application. Additionally, we will discuss the frontend and backend design, and the application's desired functionalities ultimately contributing to its effective design and implementation.

3.2 Unified modeling language (UML)

UML, the Unified Modeling Language, is a graphical language that facilitates the visualization, specification, construction, and documentation of software-intensive systems. It provides a standardized approach to depict the blueprints of a system, covering both conceptual aspects like business processes and system functions, as well as concrete elements such as classes in a specific programming language, database schemas, and reusable software components [28]. UML encompasses various diagram types, categorized into two groups: structural (or static) diagrams and behavioral (or dynamic) diagrams. The structural view emphasizes the static structure

of the system, showcasing objects, attributes, operations, and relationships through class diagrams and composite structure diagrams. On the other hand, the behavioral view highlights the dynamic behavior of the system, illustrating collaborations between objects and changes in their internal states using sequence diagrams, activity diagrams, and state machine diagrams.

3.3 Use case diagram

Use case diagrams provide an overview of the system's high-level functions and scope, as well as the interactions between the system and its actors. These diagrams capture what the system does and how actors utilize it, without delving into the internal operations of the system.

Typically, a use case diagram comprises four key components:

- Actors: Actors represent entities that fulfill specific roles within the system. They embody the actual roles of users within the system. In our application, the main actors are the therapist and the patient.
- System boundary: This defines the sequence of actions and interactions between actors and the system, also referred to as a scenario. It outlines the scope of system functionality.
- Relationships: Use case diagrams involve two types of relationships. The first type is the dependency relationship between two use cases, depicted by a dotted arrow labeled with the keyword "include." The "include" relationship signifies that the use case at the arrowhead incorporates all the steps from the included use case. The second type is the extension relationship, illustrated as a dashed line with an open arrowhead labeled with the keyword "extend." It indicates that the behavior defined in the extending use case can be optionally added to the behavior defined in the extended (base) use case. Additionally, actors and use cases are connected through associations, representing the interaction or

communication between an actor and a use case. Associations are represented by straight lines.

- Use cases: A use case represents a sequence of interactions between an actor and the system, depicting a specific function. It is visually represented by an ellipse with descriptive text. In the context of our application, we have identified the following use cases:
  - Authentication: Users can register, log in, and authenticate their accounts using Facebook, Google, or email credentials in order to access the Naffsi app.
  - Profile Information: Users, both patients and therapists, can create, update, and manage their profile information, including personal details and anonymity status.
  - Messaging: Users can communicate exchanging messages in real-time, and can view their previous conversations.
  - Appointments (Patients): Patients can view available therapists, book appointments, cancel or reschedule appointments, and provide ratings.
  - Appointments (Therapists): Therapists can manage their work hours, cancel appointments, or mark them as completed.
  - Patient Records: Therapists assigned to a patient can access and update their patient's records and notes.
  - Video Communication: Therapists can initiate video calls with patients for remote counseling sessions, and patients can join the call room from within the app.
  - Articles: Therapists have the ability to write and publish articles related to mental health and well-being for patients to read.
  - Live Events: Therapists can schedule live events such as webinars or group

therapy sessions, and concerned patients are allowed to access and participate in these events.

- Notifications: Users receive notifications indicating appointment events, new messages, and other relevant activities within the app.
- Therapists List: Users can browse the list of available therapists.
- Therapist Details: Users can view provided detailed information about therapists, including their field, description, profile picture, availability, location, and patient reviews.

The table below shows each actor and their corresponding use cases:

Table 1 – *Actors and their use cases table*

Actors	**Use cases**
Therapist	- Authentication - Profile information - Messaging - Appointments (Therapists) - Patient records - Video communication - Writing articles - Hosting live events - Notifications
Patient	- Authentication - Profile information - Therapists details - Messaging - Appointments (Patients) - Video communication - Reading articles - Joining live events - Notifications
Admin	- Addition of therapists into database

3.4 Application's use case diagrams

We utilize use case diagrams to illustrate the application's functionality for both patients and therapists. These diagrams provide a clear overview of the specific tasks and interactions involved in the Naffsi platform, offering a visual representation of how patients and therapists can effectively utilize the app's features and services.

3.4.1 Therapist's use case diagram

The figure below shows the different use cases for the therapist actor:

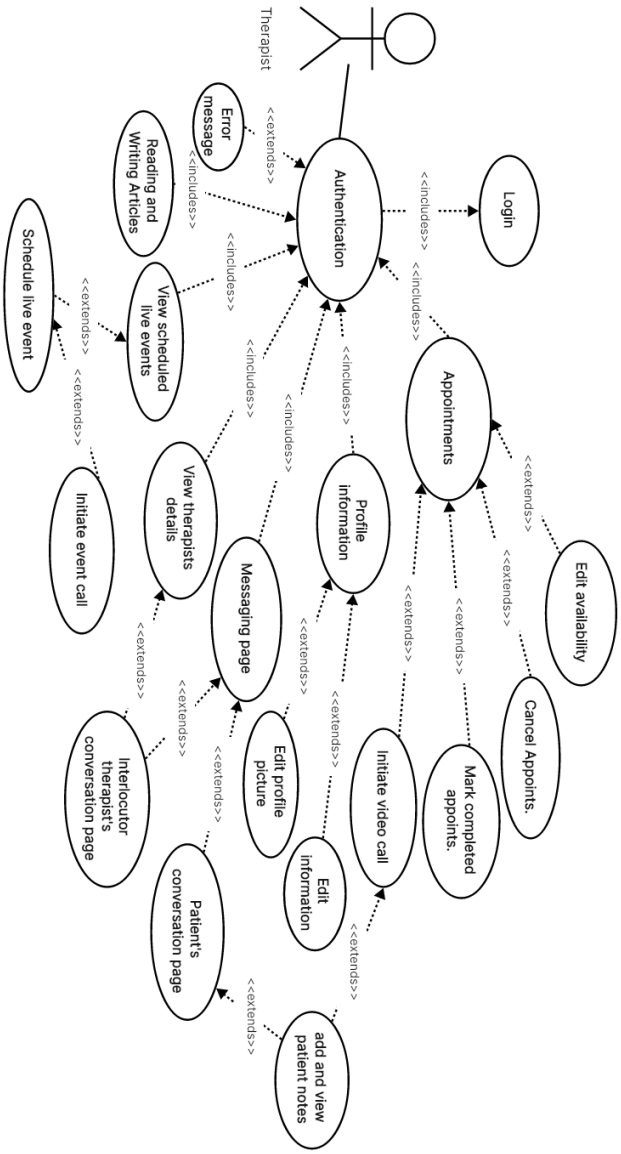

Figure 3 – *Therapist's use case diagram*

3.4.2 Patient's use case diagram

The figure below shows the different use cases for the patient actor:

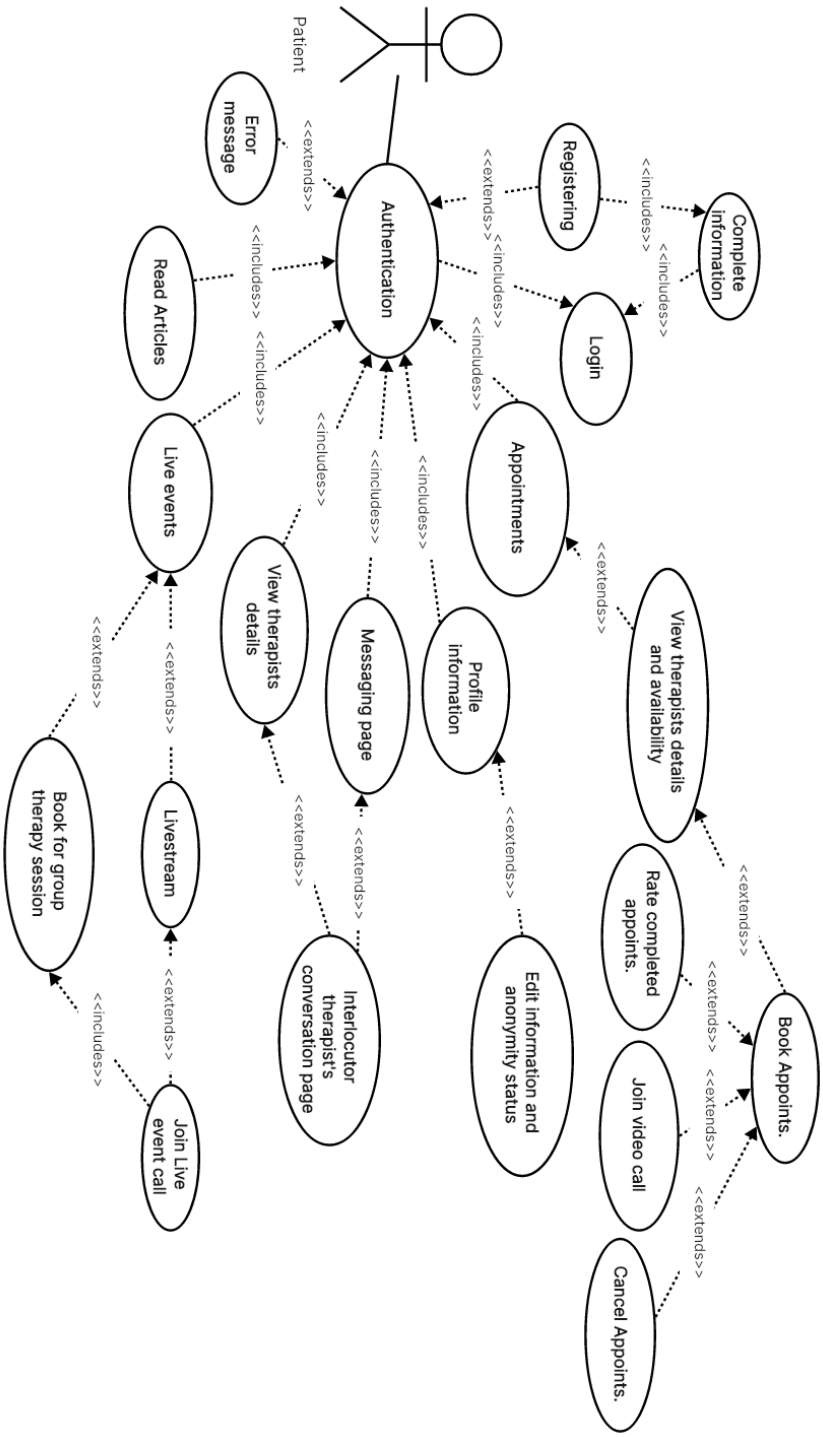

Figure 4 – *Patient's use case diagram*

3.4.3 Textual description of use cases

Therapist's use case diagram textual description

- Authentication use case

Table 2 – *Therapist's authentication use case*

Use case name	Authentication
Actor	Therapist
Object	Login and have access to the application
Scenario	1- Therapist launches the application. 2- Application requests an email and password, a Google or Facebook account login. 3- Firebase Auth verifies the query and sends favorable answer. 4- Therapist accesses the application.
Alternative	If authentication is unsuccessful, an error message is displayed

- Profile information use case

Table 3 – *Therapist's profile information use case*

Use case name	Profile information
Actor	Therapist
Object	Edit information and profile picture showcased to users
Preconditions	User should have been added manually on the 'therapists' Firestore collection, in addition to authentication
Scenario	1- Therapist navigates to profile information tab. 2- Therapists provide their information including their full name, age, gender, description, profile picture, and therapy type. 3- Therapists edit their descriptions, profile pictures and therapy types.

- Appointment use case

Table 4 – *Therapist's appointments use case*

Use case name	Appointments (therapist)
Actor	Therapist
Object	Edit availability, mark appointments as completed or cancelled, and initiate video calls
Preconditions	'therapists' Firestore collection inclusion and authentication
Scenario	1- Therapist navigates to appointments tab. 2- Therapists edit their availability in dates and times. 3- Therapist receives detailed list of previous and booked appointments, containing patients usernames, reviews, dates and times. 4- Therapist initiates video call rooms with concerned patients, or cancels appointments

- Video communication use case

Table 5 – *Therapist's video communication use case*

Use case name	Video communication
Actor	Therapist
Object	Initiate appointments video call rooms
Preconditions	Patient booked an appointment
Scenario	1- Therapist receives notification message about a patient's booking with them. 2- Therapist navigates to appointments tab and launches desired appointment's video call room joinable only by concerned patients.

- Patient record use case

Table 6 – *Therapist's patient records use case*

Use case name	Patient records
Actor	Therapist
Object	add and view own patients records
Preconditions	Patient initiated messaging, or appointment video call is ongoing
Scenario	1- Therapists open up a conversation page with a patient, or launch an appointment video call room. 2- Therapists then preview and add desired patient notes.

- Messaging use case

Table 7 – *Therapist's messaging use case*

Use case name	Messaging
Actor	Therapist
Object	Messaging with patients and therapists
Preconditions	'therapists' Firestore collection inclusion and authentication
Scenario	1- Therapists initiate conversations with other therapists through previewing the 'all therapists' list. 2- Therapists reply to and view their conversations, including patient initiated ones, within a callable user chat page. 4- Therapists navigate to chat page tab viewing their conversations list.

- Live event use case

Table 8 – *Therapist's live events use case*

Use case name	Live events
Actor	Therapist
Object	Messaging with patients and therapists
Preconditions	'therapists' Firestore collection inclusion and authentication
Scenario	1- Therapists view scheduled live events on the home page. 2- Therapists schedule their own live events through date and time selection dialog widgets. 3- Therapists initiate their live event call room as hosts for patients to join in as audience.

- Notification use case

Table 9 – *Therapist's notification use case*

Use case name	Notifications
Actor	Therapist
Object	Receiving event notifications
Preconditions	Authentication
Scenario	1- Therapist receives notification message about booking or cancelling of an appointment, or about a new incoming message. 2- Therapist navigates to the notification's appropriate page or tab

Patient's use case diagram textual description

- Profile information use case

Table 10 – *Patient's profile information use case*

Use case name	Profile information
Actor	Patient
Object	Edit information and anonymity status
Preconditions	Authentication
Scenario	1- Patient navigates to profile information tab. 2- Patients provide and edit their anonymity status (true by default) and their information including their username (if they choose not to be anonymous), age, gender, and desired therapy categories.

- Therapist details use case

Table 11 – *Patient's therapist details use case*

Use case name	Therapist details
Actor	Patient
Object	View therapists details
Preconditions	Authentication
Scenario	1- Patient clicks on a therapist profile whether from the homepage suggested list, or 'all therapists' list. 2- Patients can see the therapist's username, profile picture, therapy field, work location, description, rating scores, and availability.

- Appointment use case

Table 12 – *Patient's appointments use case*

Use case name	Appointments (Patient)
Actor	Patient
Object	Book and cancel appointments
Preconditions	Authentication
Scenario	1- Patient books an appointment with a therapist through details page. 2- Patient navigates to appointments tab. 3- Patient views their scheduled appointments and accesses an appointment's call room if its ID is provided by the therapist. 4- Patient can cancel a scheduled appointment, and can view the number of reschedulings they have left, in case of them conducting multiple payments, or in case of therapists cancellings. 5- Patient views list of previous appointments and provides these session's ratings.

- Video communication use case

Table 13 – *Patient's video communication use case*

Use case name	Video communication
Actor	Patient
Object	Join appointment video call
Preconditions	Patient booked an appointment, and therapist initiated a call room
Scenario	1- Patient receives notification message about the designated therapist's launching of a video call room. 2- Patient navigates to appointments tab, and accesses a ready appointment's meeting.

- Messaging use case

Table 14 – *Patient's messaging use case*

Use case name	Messaging
Actor	Patient
Object	Messaging therapists
Preconditions	Authentication
Scenario	1- Patient initiates conversation with a therapist by viewing the suggested therapists section, or 'all therapists' list. 2- Patient reply to and view their conversations within a callable user chat page. 3- Patients navigate to chat page tab viewing their conversations list.

- Live event use case

Table 15 – *Patient's live events use case*

Use case name	Live events
Actor	Patient
Object	Joining livestreams and group therapy sessions
Preconditions	Booking for a group therapy session
Scenario	1- Patient views scheduled live events on the home page. 2- Patient books for a group therapy session. 3- Patient joins in a livestream as an audience member, or participates in a booked for group therapy session.

- Article use case

Table 16 – *Patient's articles use case*

Use case name	Articles
Actor	Patient
Object	Reading community articles
Preconditions	Authentication
Scenario	1- Patient views articles list on the home page. 2- Patient reads an article and its author's username.

- Notification use case

Table 17 – *Patient's notification use case*

Use case name	Notifications
Actor	Patient
Object	Receiving event notifications
Preconditions	Authentication
Scenario	1- Patient receives notification message about cancelling of an appointment, about a ready video call room or about a new incoming message. 2- Patient navigates to the notification's appropriate page or tab

3.5 System diagrams

The Diagrams section of the project encompasses various visual representations, including sequence diagrams, class diagrams, and document relationships diagrams. These diagrams serve as powerful tools to depict the flow of interactions, the structural organization, and the dependencies between different components within the Naffsi application.

3.5.1 Used symbols

Here are some definitions of some symbols we used in the succeeding system diagrams:

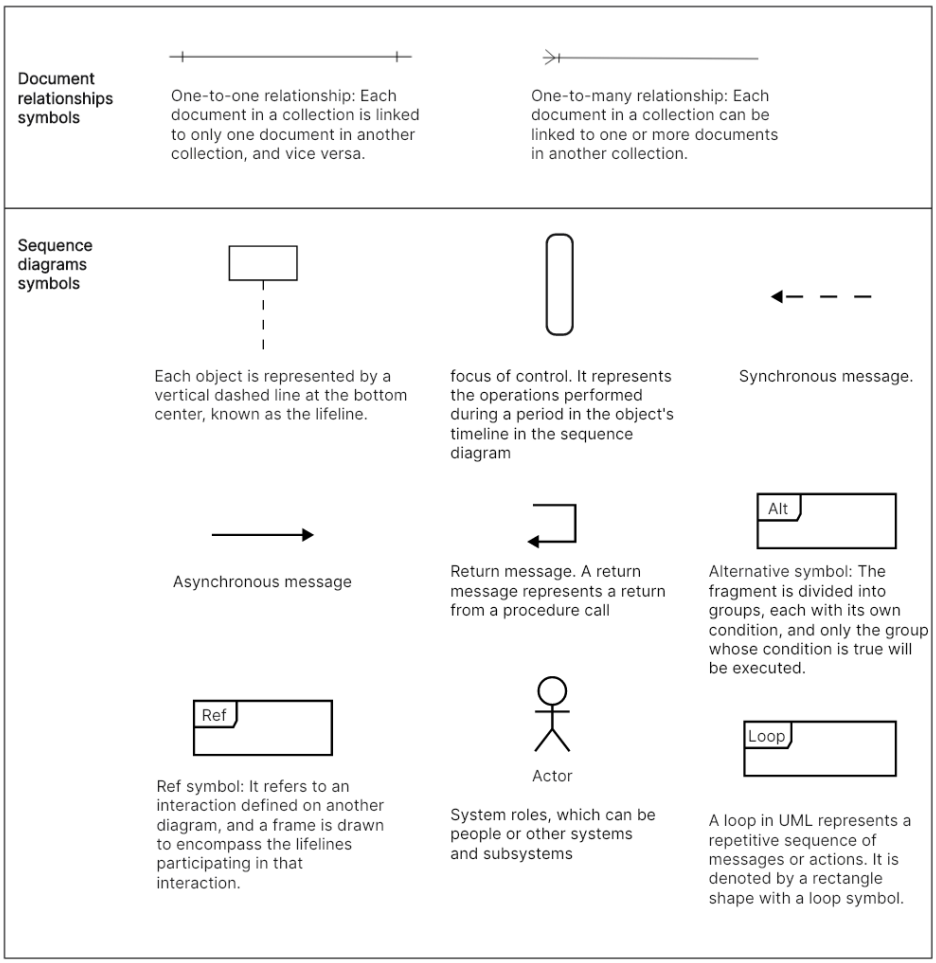

Figure 5 – *Used symbols figure*

3.5.2 Class diagram

The following diagram represents the different classes and their relationships:

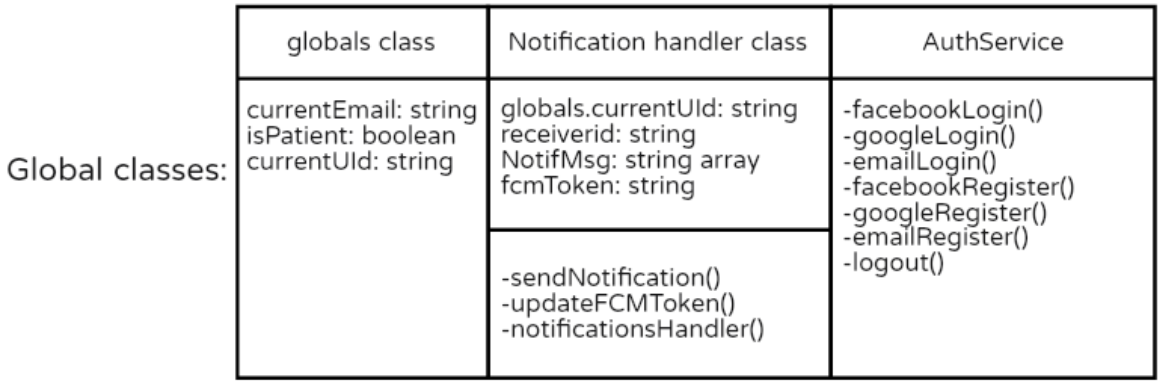

Figure 6 – *Class diagram*

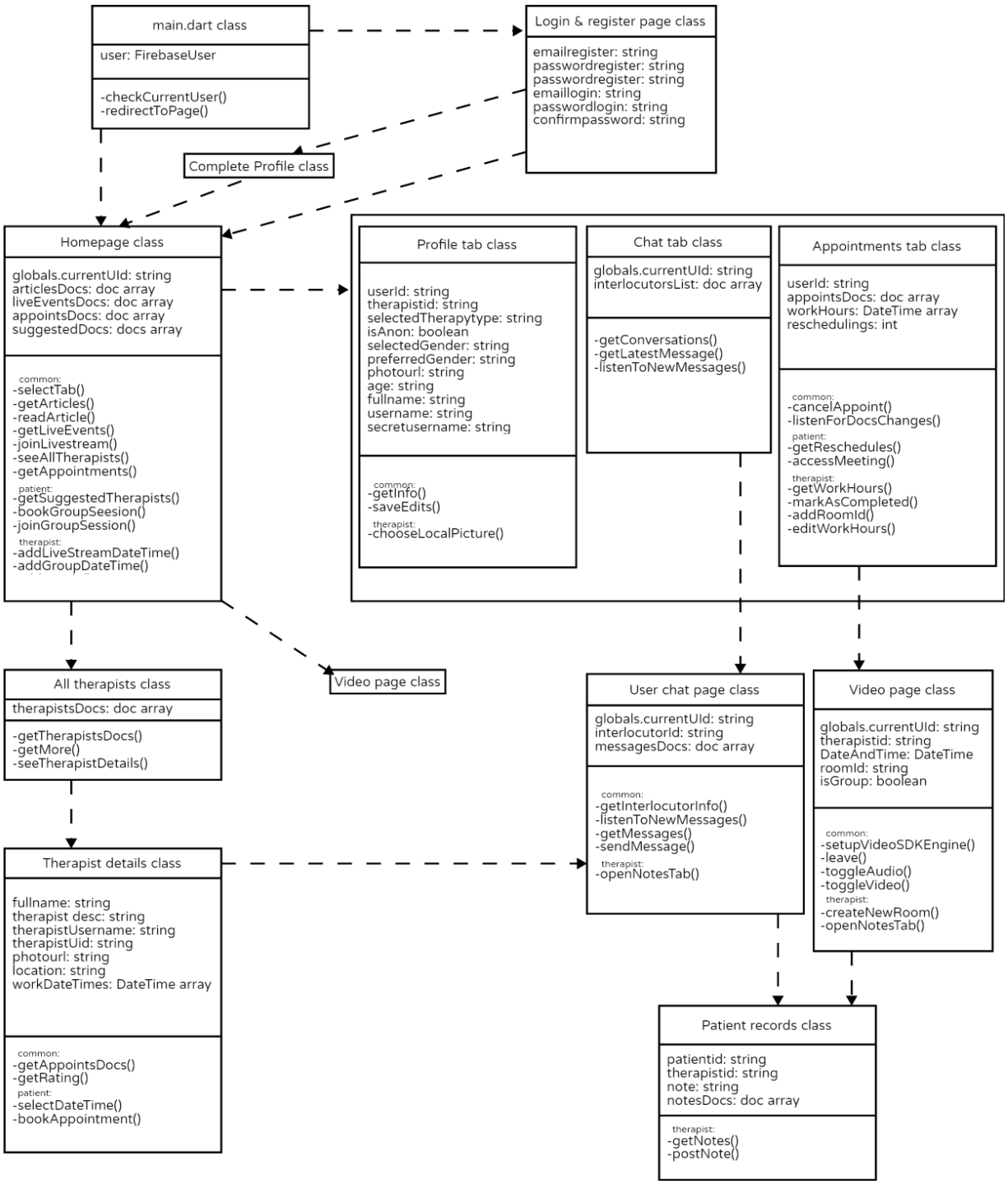

Figure 7 – *Class diagram continuation*

3.5.3 Relationships between documents

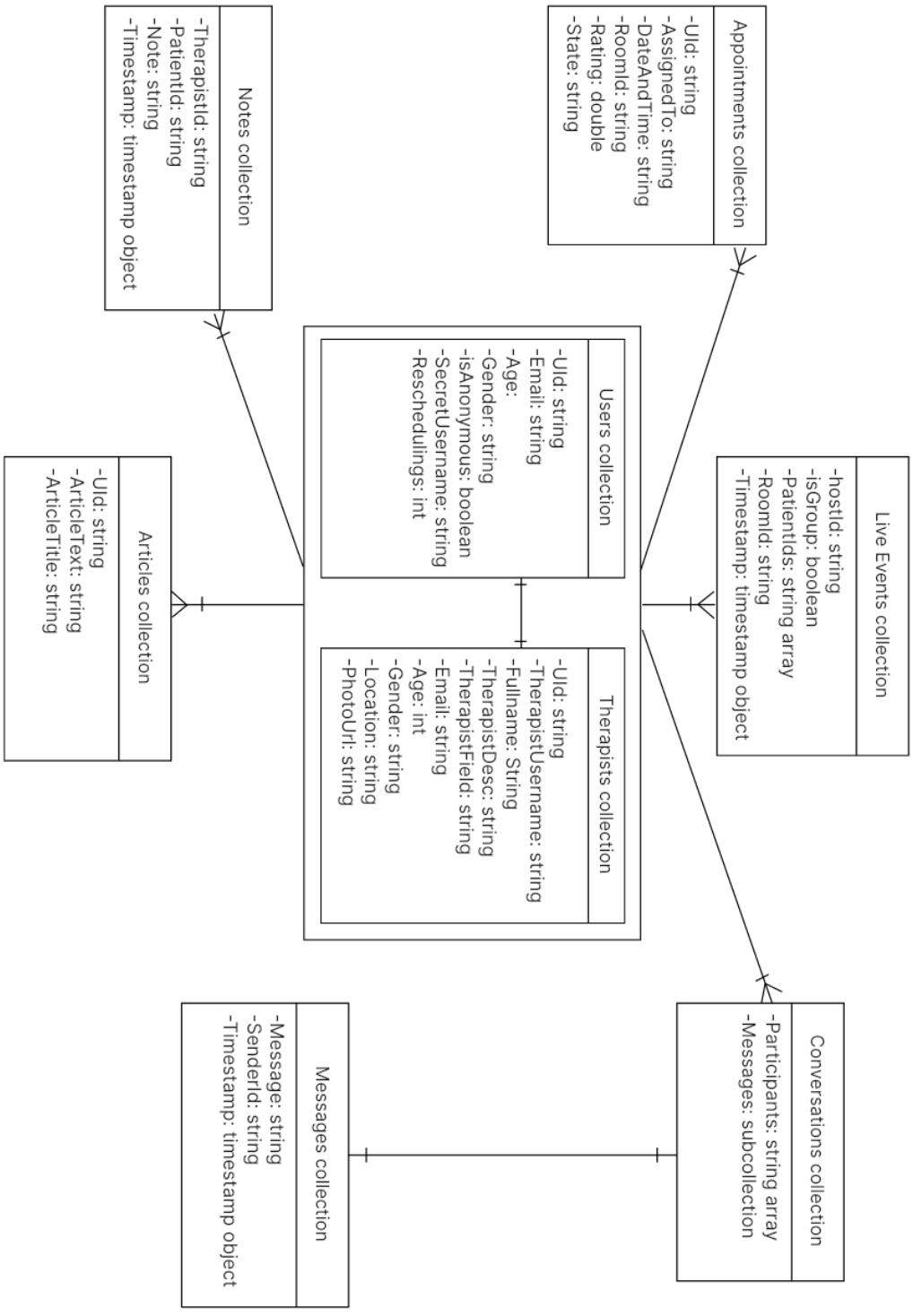

Figure 8 – *Relationships between documents diagram*

3.5.4 Sequence Diagrams

Sequence diagram, the prevalent type of interaction diagram, centers around the exchange of messages among multiple lifelines. It portrays the interaction through the sequence of messages and their occurrence specifications on the lifelines.

In order to help visualize some Naffsi's functionalities processes, we depict them in sequence diagrams.

Appointments process sequence diagram

Here we assume a one-to-one interaction between a patient and a therapist in an appointment booking process:

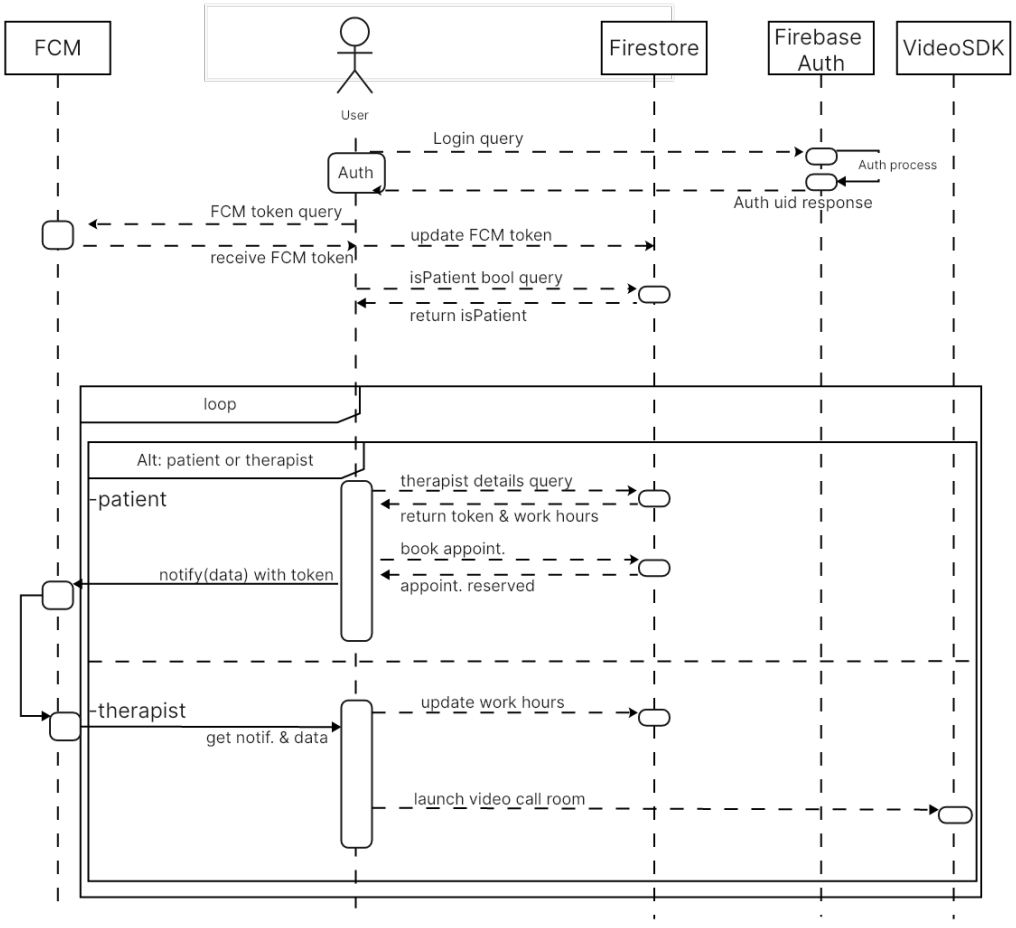

Figure 9 – *Appointment process sequence diagram*

Appointment video communication sequence diagram

Here we assume a one-to-one interaction between a patient and a therapist in a one-to-one video call:

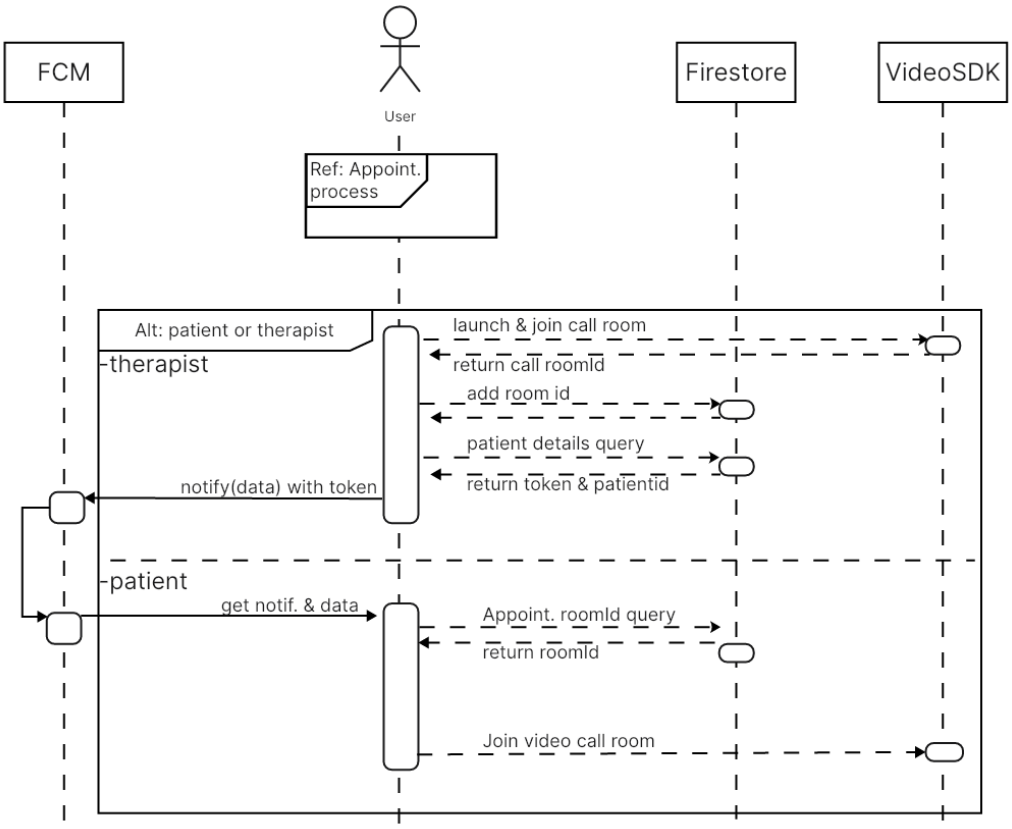

Figure 10 – *Appointment video communication sequence diagram*

Messaging sequence diagram

Here we assume a one-to-one interaction between a patient and a therapist in a messaging process:

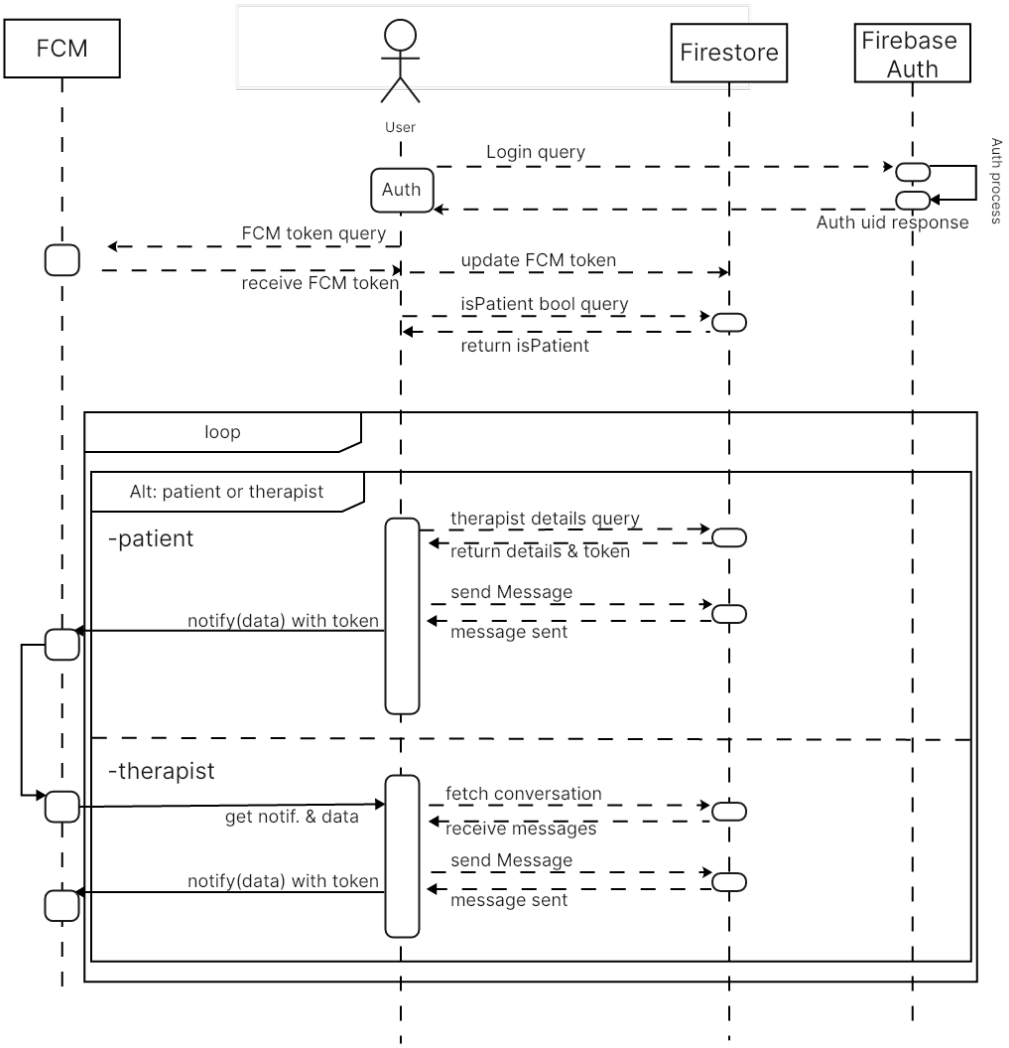

Figure 11 – *Messaging sequence diagram*

3.6 Frontend design

The UI Design section focuses on the the Naffsi app's visual and interactive aspects. Its purpose is to outline the design principles, components, and user experience considerations that were taken into account during the development process.

3.6.1 App Structure and Main Navigation

The app structure contains a main HomePage widget class containing a bottom navigation bar utilized for navigation across three tabs: the ChatPage, EditProfile and

AppointmentsList tabs. In addition to secondary widget classes accessible through the tabs such as LoginPage, VideoCallPage, and UserChatPage which will be listed and discussed further along this chapter.

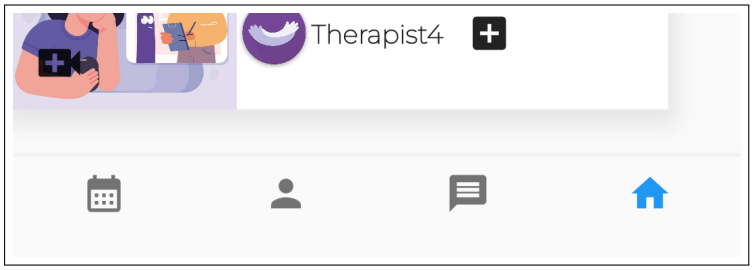

Figure 12 – *Naffsi's homepage Bottom navigation bar*

3.6.2 Login and "on-boarding" page design

The Login Page contains the typical login UI components for an application, with login and register tabs, text fields for filling in emails and passwords, and the possibility to login or register through Google or Facebook. Whereas the "on-boarding" page is a welcoming page for newly registered users, showcasing the app's objectives and values.

Figure 13 – *Naffsi's logo and welcome image*

3.6.3 Conditional rendering

Conditional rendering has been set to be utilized through out the project in a way that only relevant widgets display to the app's both user types: a therapist or a patient.

3.6.4 Home page design

The home page widget is designed to display Naffsi's essential features, through horizontally scroll-able lists representing: previous and future appointments, suggested therapists recommended for the current user, well-being articles written by the platform's therapists, and upcoming live events (live-streams and group therapy sessions). In addition to allowing the possibility for therapists to add articles and schedule live events.

3.6.5 Chat and user chat page design

The chat page is designed in a typical manner such that it displays conversations that the current user has partook in, along with the latest message sent or received. The user chat page is titled by the relevant username and displays continuously synchronized messages along with their timestamps.

3.6.6 Appointment page design

The appointment page was designed in an effort to be sufficiently user-friendly for both therapists and patients, allowing users to manage their appointments by simply viewing the dates and providing ratings. In addition to allowing therapists specifically to initiate calls, mark appointments as finished or cancelled, and edit their work hours for each day separately or collectively with an intuitive interface implementation of the calendar carousel widget, containing a calendar dialog then a thirty minute range time intervals one.

3.6.7 Edit profile page design

The design of the profile page allows the current user to view and edit their anonymity status and their profile information (username, age, gender, preferred therapy categories), along with allowing the current therapist a profile picture addition. A similar profile page is displayed on the user's first login for their information input.

3.6.8 "All therapists" and "therapist-details" page design

When the user chooses viewing all available therapists, an activity displays a list of available ones along with their descriptions, profile pictures, work locations and star reviews. Furthermore, a user may view additional data about a therapist such as their work hours, and may book an appointment or initiate messaging with said therapist through the therapist details activity.

3.6.9 Video call page design

This layout page is inspired by the VideoSDK package's one, and allows displaying the interlocutors and the user's own webcam streams, in addition to controls for toggling between enabling or disabling video and audio input.

3.6.10 Patient record design

Helpful to the therapist, this layout contains a tab allowing the addition and viewing of notes about a specific patient, whether on the user chat or the video call pages, displayed as an end drawer alongside these layouts.

3.6.11 Event message design

Event messages comprise of login or internet disconnection errors, successful updates to the database, and notification messages. While errors are displayed using flutter's Snackbar widget, executed updates are displayed as being successfully conduc-

ted using flutter Toasts, and notifications as typical alert messages depending on the device utilized, with the help of the flutter local notification package.

3.6.12 Article and live event section design

Articles are designed to be displayed in a list widget specifically on the home page screen, alongside a similar live events widget. These subsections additionally provide simple article input text dialogs for therapists, and provide a calendar and time widgets for live event scheduling inputs.

3.7 Backend design

3.7.1 Firebase authentication

In the backend development chapter, Firebase Authentication plays a key role in the Naffsi app. The Auth-service.dart class is implemented as to handle user authentication, including Google and Facebook sign-up and sign-in, error handling and sign-out functionalities. These classes' functions can be called by simply initiating the class and referencing the desired method, i.e: AuthService().logout().

3.7.2 "Complete profile" and "edit profile" backend

After choosing to register for a new account on the app, a unique userId which is used as every user's main identifier, is saved from the Firebase Authentication service to the Firestore database, along with the user information they are prompted to provide on the complete profile page. In addition, another unique username formatted as the word user concatenated to a random number, i.e: user12501 is generated and saved for each user, attempting to assist therapists in at least identifying a number associated with an anonymous user profile. This data, except for the secret username, can of course be edited and synchronized later on from the edit profile section.

3.7.3 Main.dart and global variables

Every Flutter app starts by firstly executing code on the main.dart class. Consequently, it is used to set up the app's launching in the foreground first parameters. A Firebase instance is called checking if a user is already logged in, displaying the homepage or the login page, and checking if the currentUserId corresponds to a patient or a therapist as to display relevant widgets accordingly. The therapists table is set to be configured manually on the Firebase console, as Naffsi is set not to allow new therapists directly registering. In addition, a list of global variables which use is relevant across the app are initialized, mainly: currentEmail, the isPatient boolean, FCM token and currentUserId.

3.7.4 Firebase Firestore

The Naffsi app uses a Firebase Firestore NoSQL database. NoSQL is a non-relational database approach that offers flexibility and scalability for modern applications. We chose NoSQL for the app based on the specific requirements of the project[29]. The main advantage of using NoSQL, and specifically Firebase Firestore, over a traditional MySQL database is its real-time data synchronization and horizontal scaling capabilities[30]. Firestore allows for instant updates and notifications, ensuring that data changes are immediately reflected across all connected devices, and guarantees handling increased data and user load by adding more servers or instances to the database cluster, executing tasks in a parallel manner independent of increased traffic [31], while providing a simple yet effective document-based structure to store data documents in JSON files, ordered in dynamically createable collections and sub-collections. These are crucial elements for a real-time messaging and counseling app like Naffsi, where seamless and timely updates are essential. One essential Firebase object utilization is the snapshot object, which continuously listens for a database change on a specific node, as shown in the figure below:

```
void listenToDocumentChanges() {
FirebaseFirestore.instance
    .collection('messages')
    .doc('conversation')
    .snapshots()
    .listen((DocumentSnapshot snapshot) {
  if (snapshot.exists) {
    // Handle document changes here
    // Access the document fields using snapshot.get('field_name')
  } else {
    // Document doesn't exist
  }
});
}
```

Figure 14 – *Firestore snapshot code example*

Queries and updates in a relational context

Where multiple row queries and relational fields are required, we carefully implement through out the project CRUD operations using Flutter Firestore syntax, in a manner disallowing inconsistencies such as missing fields, incorrect data types, and inaccurate data queries, accounting for the fact that Firestore is a non-relational database which adds documents into collections regardless of their fields, therefore simulating a relational database when needed. We achieve this by using appropriate code listed below for different scenarios:

- A consistent query syntax dedicated to fetching data through field equality conditions, using the CollectionReference, DocumentSnapshot, and QueryFirestore Objects, and the where(isEqualTo: “field”) method.

```
final CollectionReference _collectionReference =
    FirebaseFirestore.instance.collection('therapists');
var _querySnapshot = await _collectionReference
    .where('uid', isEqualTo: globals.currentUId)
    .get();
```

Figure 15 – *Fetching data code example*

- A query syntax dedicated to updating data.

```
FirebaseFirestore.instance.collection('users').add({
  "uid": user!.uid,
  "email": globals.currentEmailRegister,
  "username": (isAnon) ? "" : _usernamecontroller.text,
  "secretusername": "user" + t.toString(),
  "isanonymous": isAnon,
  "gender": (selectedGender) ? "Male" : "Female",
  "age": _agecontroller.text,
  "reschedulings": 0,
});
```

Figure 16 – *Updating data code example*

- Syntax for deleting a document from the database, for this example below, appointments are deleted as intended so by the therapist:

```
var snapshot = await appointmentsRef
    .where("uid", isEqualTo: globals.currentUId)
    .where("dateandtime", isEqualTo: key + " " + element)
    .where("state", isEqualTo: "available")
    .get();
for (var doc in snapshot.docs) {
  await doc.reference.delete();
}
```

Figure 17 – *Deleting data code example*

3.7.5 Data models

This subsection will discuss how data is structured and stored in the database. Firestore uses collections, which are containers grouping other subcollections, or related JSON documents together, instead of SQL tables. In the figure below are all the collections names:

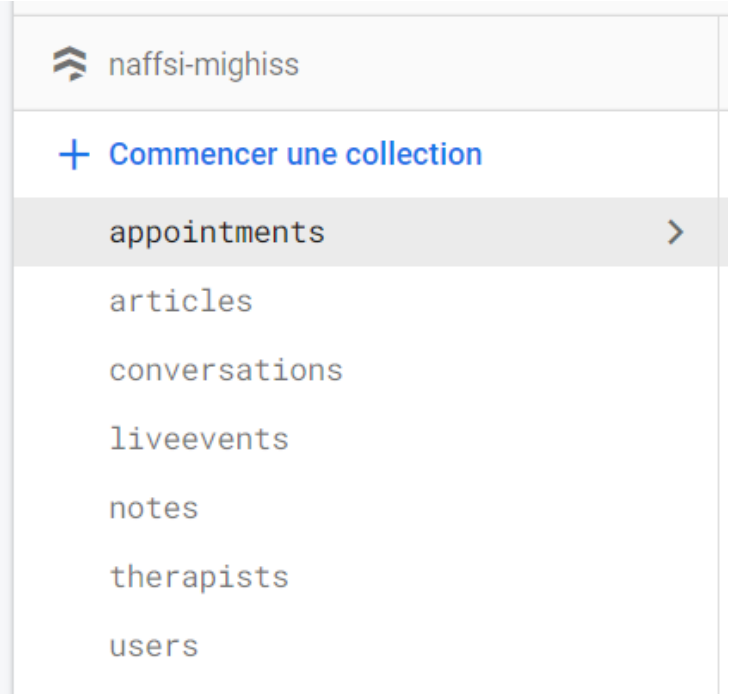

Figure 18 – *List of all collections in Firestore*

In order to assist in visualizing the data structure we propose listing and detailing these collections along with their contained documents fields as follows:

Users collection

- userId: unique user identification.
- email
- age
- gender
- isAnonymous: boolean defining user's anonymity status.
- secretUsername: randomly and uniquely generated for utilization with anonymous users i.e: "user15457".
- reschedulings: number of reschedulings available to the user in case of cancellings or multiple appointment unity price payments.

Therapists collection

- userId: unique therapist identification.
- therapistUsername and fullname: intended for patients display.
- therapistDesc and therapistField: therapist's description and field.
- email
- age
- gender
- location: work location for one-on-one traditional therapies.
- photoUrl: therapist profile picture's URL on Firebase Storage.

Appointments collection

- userId: identifies therapist concerned with specific work hour.
- assignedTo: userId array of the patients who booked the appointment.
- dateAndTime: date and time of the appointment formatted "yyyy-mm-dd hh:mm".
- roomId: joinable VideoSDk meeting room Id.
- rating: session's rating provided by the patient.
- state: current state of appointment (reserved, available, finished, cancelled).

Conversations collection

This collection contains documents with a "participants" array field, and they in turn each contain a "messages" subcollection, with message documents consisting of message text, senderId, and timestamp fields.

- participants: an array field containing the conversation's participants userIds.
- conversation: a collection containing the messages sent and received between both participants.
- message: message text.
- senderId: message sender's identification.
- timestamp: message's date and time object.

Notes collection

- userId: therapist's identification.
- patientId: patient's identification.
- note: note text.
- timestamp: note's date and time object.

Live events collection

- hostId: event host therapist identification.
- dateAndTime: date and time planned for the event.
- roomId: joinable VideoSDk meeting roomId.
- isGroup: boolean determining if live event is group therapy session
- patientIds: userIds of patients who booked for the group therapy session

Articles collection

- userId: author's identification.
- articleText: article's text body.
- articleTitle: article's title.

3.7.6 Security rules

Firebase Firestore provides setting up database's security rules allowing only authenticated users to read from and write into collections, it is implemented simply in the Firebase console as shown in the following figure:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    // Allow read access to authenticated users
    match /{document=**} {
      allow read: if request.auth != null;
    }

    // Allow write access only to authenticated users
    match /{document=**} {
      allow write: if request.auth != null;
    }
  }
}
```

Figure 19 – *Firestore security rules interface*

3.7.7 Firebase Storage

Although set to be utilized minimally throughout the project, it is worth mentioning that Firebase Storage assisted in backend development by providing nodes in which profile pictures image files are stored, and in providing better user experience. Once the therapist provides an image file and its upload is successful, Firebase Storage upload function returns the image file storage retrieval URL, which is stored in turn to Firestore under the relevant therapist's document.

3.7.8 Appointment backend

Appointments are a crucial element for an online counseling app like Naffsi. Their backend design on this project combines with query constraints, error-handling and code consistency, therefore seamlessly integrating with the frontend, and providing a comprehensive appointment management system. In consequence, it allows storing organized appointments in the database's provided collection within appointment type documents. Appointments have the "state" field initially marked as available, editable by therapists to cancelled or finished, and by patients to reserved or cancelled. Additionally, patients can retrieve their appointments as well as therapists availability from the database, permitting easy access, rating or scheduling of sessions. Regarding multiple payments and appointment cancellings, the patient reschedulings field is utilized representing how many possible bookings are available.

3.7.9 Homepage backend

The Naffsi's homepage backend handles fetching data for the app's most important features, it fetches from Firestore along with the user's appointments the suggested therapists for them, the app's therapists articles, and upcoming live events.

3.7.10 Messaging backend

The messaging backend is designed allowing sending messages, synchronously and continuously listening for received ones and their timestamps from the database, using a Firebase snapshot object on the relevant conversation node.

3.7.11 Notification backend

So as to reliably send and receive push notifications, we use Firebase Cloud Messaging. The way that FCM works is it sends notifications based on a device token, and handles received ones on the background or foreground using an event listener. Consequently, we implement a NotificationHandler.dart class containing an updateFCMtoken method called on every user login or logout ensuring the latest token is always associated with every user's device, a sendNotification method which requires notification body (title, body text, and metadata) and userId as parameters, in addition to a firebaseMessagingHandler method which handles received notifications for display using the flutter-local-notifications package.

3.7.12 VideoSDK backend

The VideoSDK plugin is utilized with an appID inside the used class, specifically set in the console platform for this project. On the backend side, users can join a room provided they possess its roomId, which is set on the "roomId" field relevant meeting's document by the therapist.

3.7.13 Patient record backend

Patient records are organized for each therapist and their designated patients in the notes collection in Firestore, and are stored along side the users' relevant userIds and the notes' timestamps.

3.7.14 Article and live event backend

It is simply designed to fetch and add organized articles and live events from and into their respective collections in Firestore.

3.8 Features and functionality

3.8.1 Appointments

Therapists have the ability to add, edit, and delete appointments individually as well as collectively, where collective work hour editing specifically is implemented by a simple algorithm, which detects multiple selected dialog days similar in work hours, then updates these potential appointments in the database accordingly. Therapists can also set the "roomId" field required for patients joining in a meeting. Patients on the other hand can pay for multiple appointments, or reschedule sessions in the case of cancelling, and provide finished state sessions' rating. This two-way interaction between therapists and patients ensures a smooth and efficient appointment process, enhancing the overall user experience of the app.

3.8.2 Video calling with VideoSDK

As mentioned on the VideoSDK backend subsection, the service requires from users a roomId in order to join a video call. We adapt this to Naffsi's desired functionalities which are: one-on-one or group video calls, livestreams where there is one host therapist, and group therapy sessions where only the audience's audio is permitted. Using the VideoCallPage.dart class optional meetingID parameter, we

define if the room is being created or joined into, depending if this variable's value is empty, otherwise we assign it. In addition, we use the optional ifAudioAllowed parameter to define if the room being created or joined into is a livestream or a group therapy session, consequently configuring the current user client role type as host or audience, and enabling or disabling webcam and audio inputs and outputs.

3.8.3 Articles and live events

Aiming to create a collaborative and supportive community, Naffsi allows therapists the ability to add articles and their titles, and the ability to schedule live events, which are displayed to all users.

3.8.4 Patient records

The patient records functionality allows therapists to input notes desired to be saved for a specific user, it is made possible from within this user's chat page or from the video call page both sides are participating in.

3.9 Conclusion

The Naffsi app's Design chapter created the foundation for the implementation stage, creating the conditions for transforming our conceptual design into a useful reality. We have developed a thorough design for the application using UML modeling, diagrams, frontend and backend design, and addressing functionalities.
Our design choices have been informed by a comprehensive study of the system requirements and a careful consideration of user needs, guaranteeing that the Naffsi app will successfully satisfy the needs of individuals looking for mental health help.

We will start the real development process of turning the design artifacts into a functioning application when we go into the following chapter: implementation.

Chapter 4

Implementation

4.1 Introduction

In this final chapter, following the design phase and the specification of user roles, our focus shifts to the implementation of the application. We bring the envisioned ideas into reality by transforming them into a functional and interactive system. This chapter encompasses a comprehensive presentation of the application, showcasing its various features and functionalities. Additionally, we provide a tangible glimpse into the user experience.

4.2 Interfaces of login and on-boarding pages

The login page provides an interface allowing email, Facebook or Google login and registering. When users first register, they are greeted with the on boarding page.

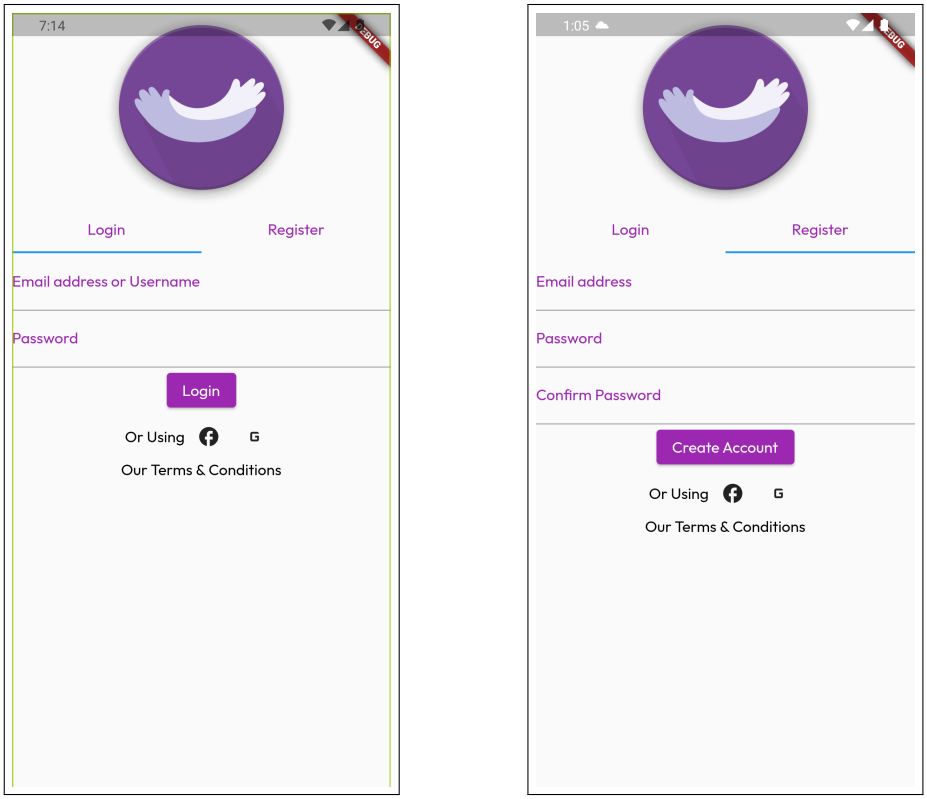

Figure 20 – *Login and register interfaces*

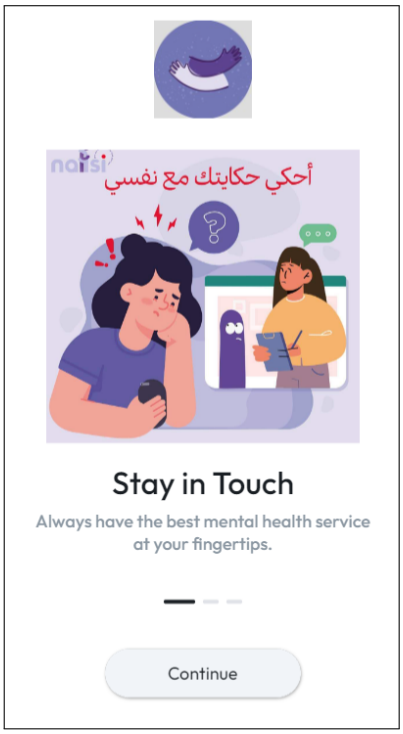

Figure 21 – *On boarding interface*

4.3 Interfaces of complete and edit profile pages

The complete profile page provides an interface similar to the edit profile tab page, allowing users to fill in their information after registering.

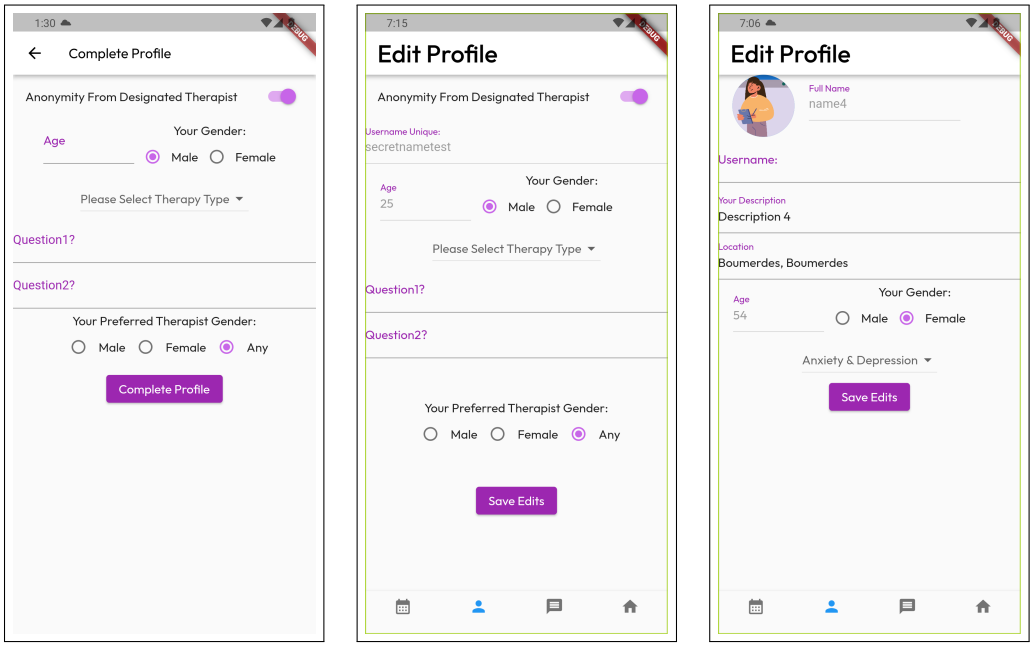

Figure 22 – *Complete profile, patient edit profile and therapist edit profile interfaces*

4.4 Interface of homepage and "all therapists" pages

On successful login, users are redirected to the homepage, where they can see a list of suggested therapists, and can choose to view all of the available ones.

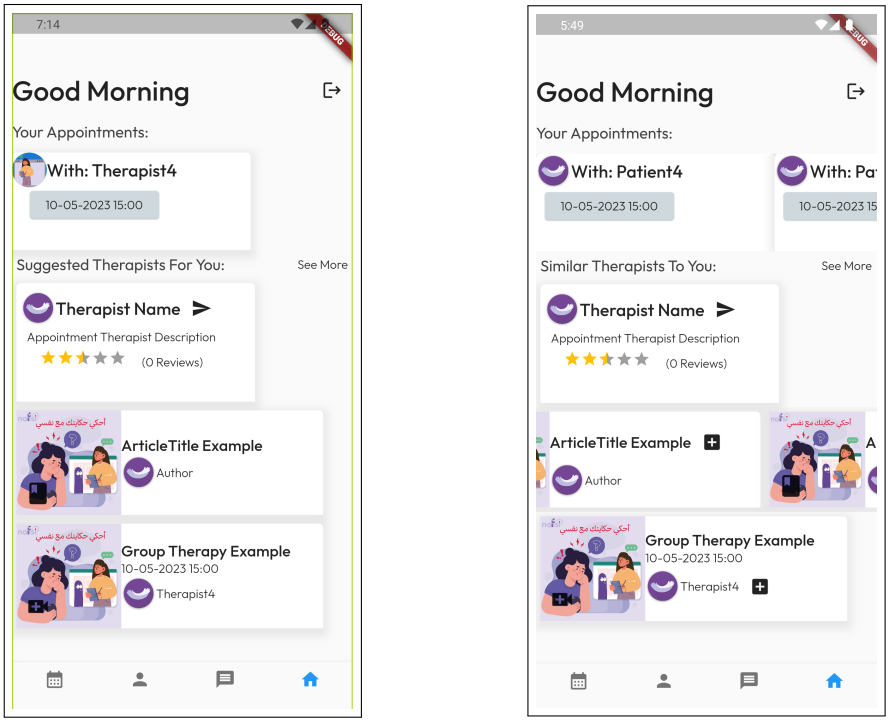

Figure 23 – *Patient homepage and therapist homepage interfaces*

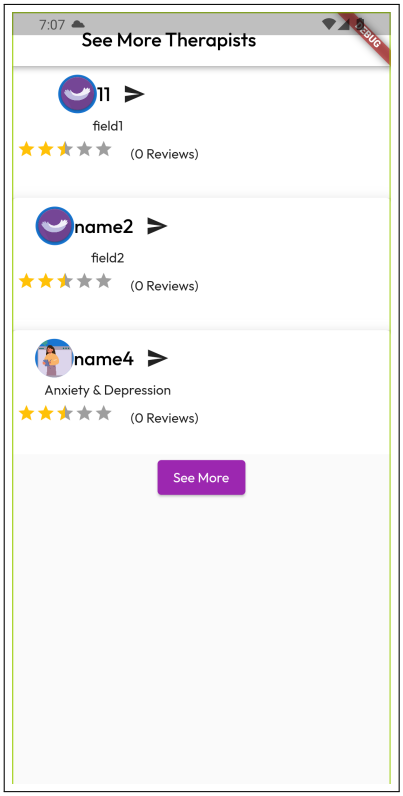

Figure 24 – *See more therapists interface*

4.5 Interface of therapist details and booking appointment pages

After selecting a therapist, users can access detailed information about the therapist and proceed to book an appointment conveniently.

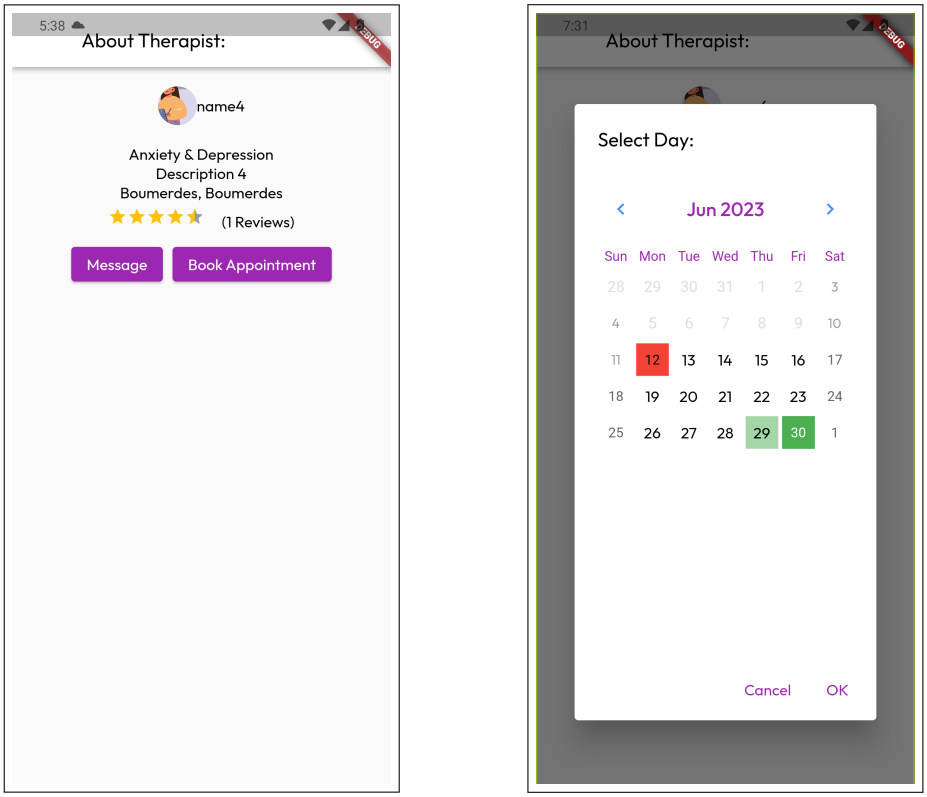

Figure 25 – *Therapist details and day availability interfaces*

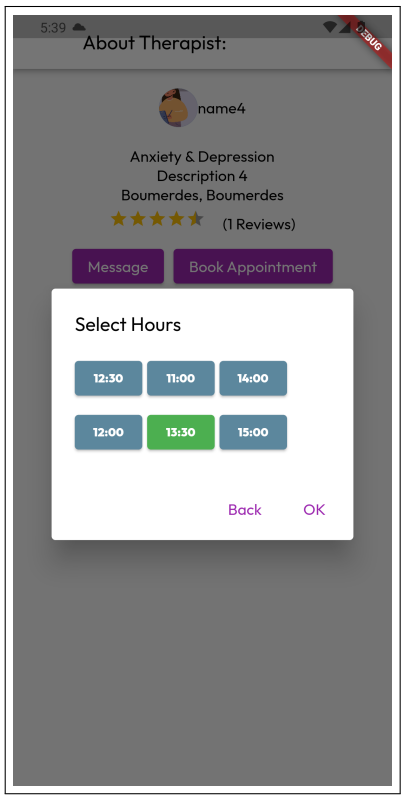

Figure 26 – *Booking hours interface*

4.6 Interface of patient and therapist appointment tab pages

The appointments tab interface displays conditionally the relevant functionalities for both types of users, where therapists can manage their work hours, and patients can manage and rate appointments.

4.6.1 Interface of patient appointment tab pages

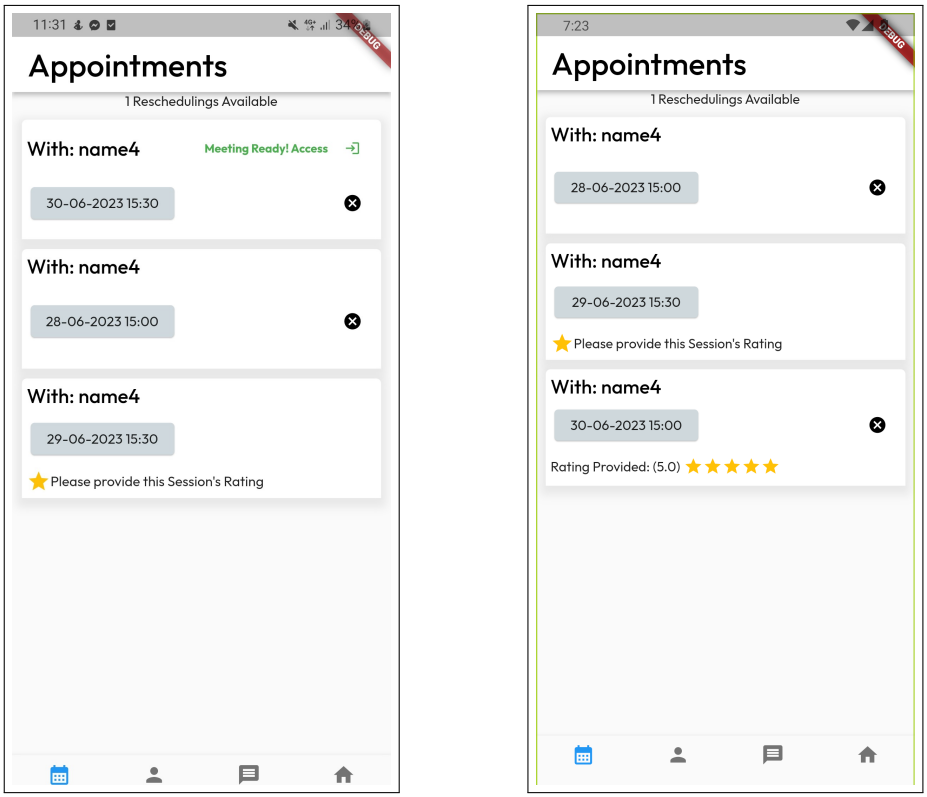

Figure 27 – *Patient appointment tab interface*

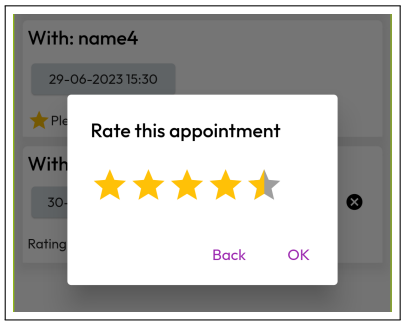

Figure 28 – *Patient appointment rating interface*

4.6.2 Interface of therapist appointment tab pages

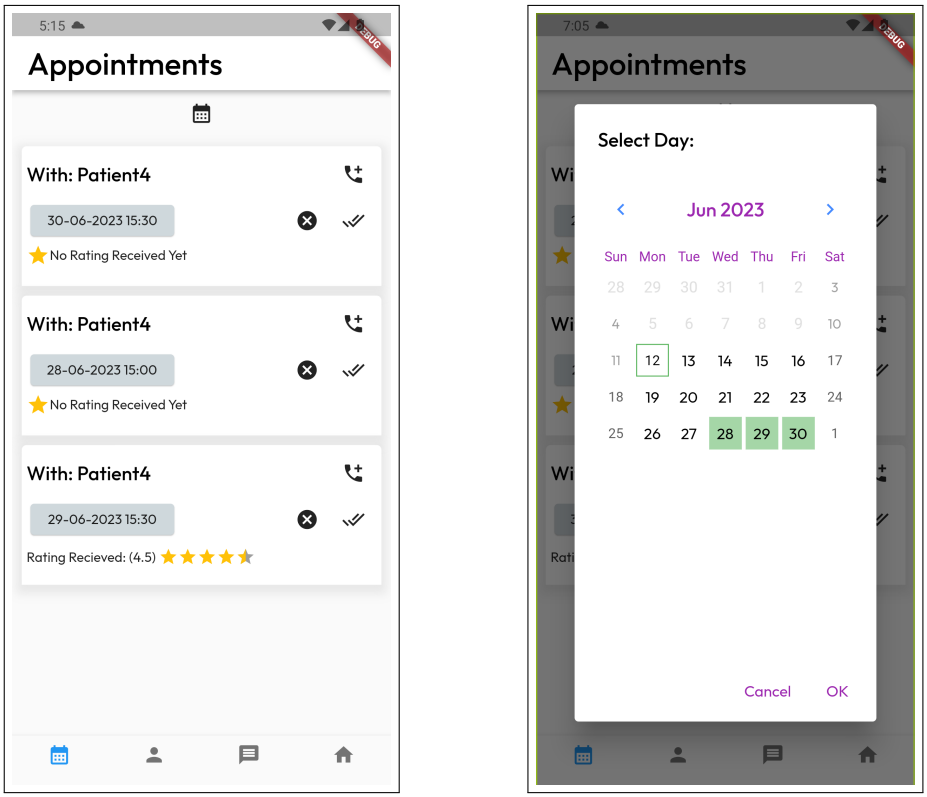

Figure 29 – *Therapist appointment and availability interfaces*

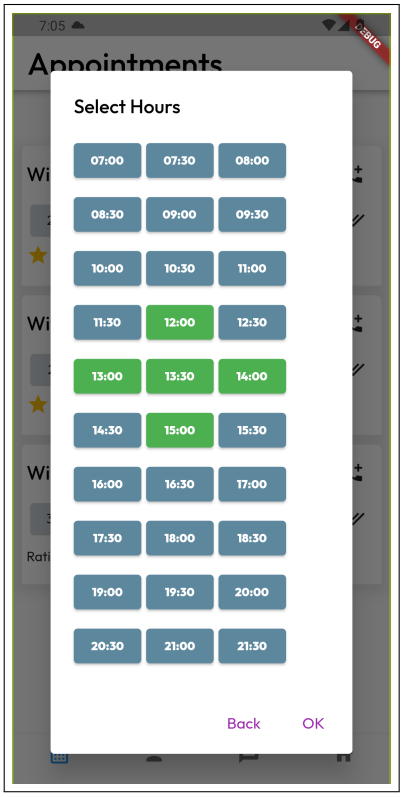

Figure 30 – *Therapist time of day availability interface*

4.7 Interfaces of therapist appointment management pages

Therapists receive notifications when appointments are booked by patients and can choose to initiate a video call room by sending the room id to the relevant patients.

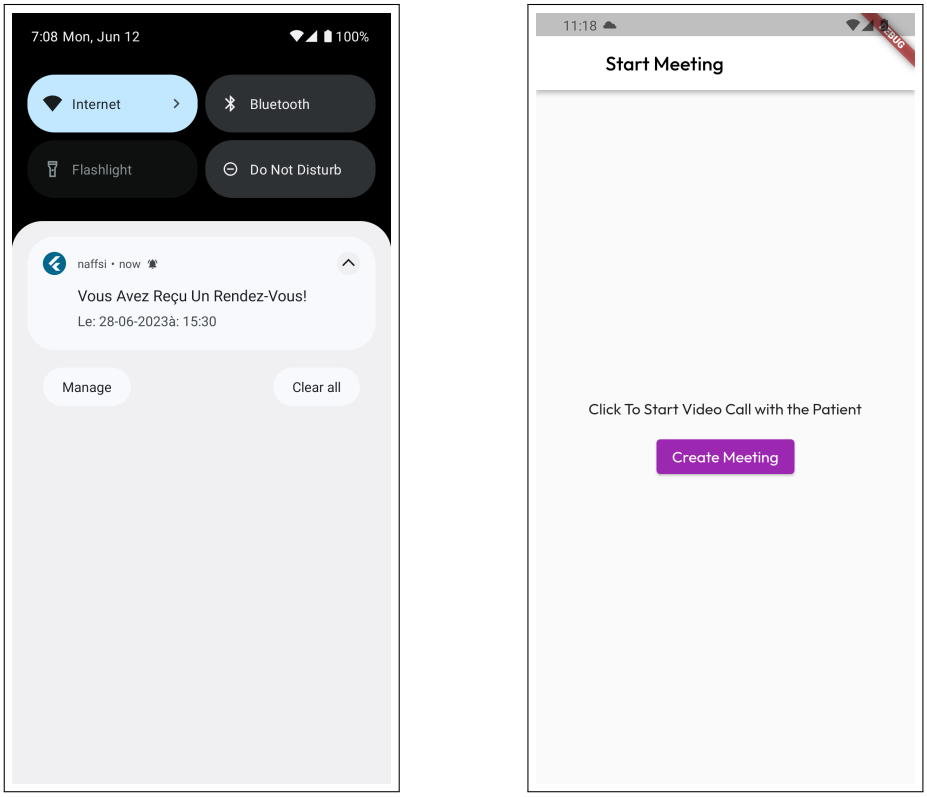

Figure 31 – *Therapist booking notification and starting call interfaces*

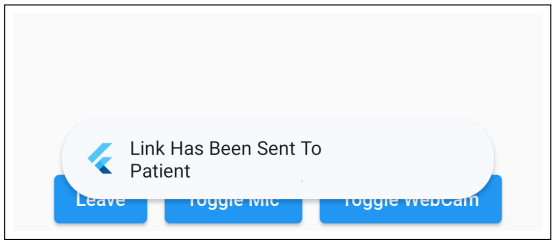

Figure 32 – *Successful video call initiation toast*

4.8 Interfaces of messaging pages

Users can view their lists of conversations, and preview messages in the user chat page, while therapists specifically can add their notes in a conversation with a patient using the drawable patient notes tab interface, accessible also from the video call page.

4.8.1 Therapist messaging and patient notes interfaces

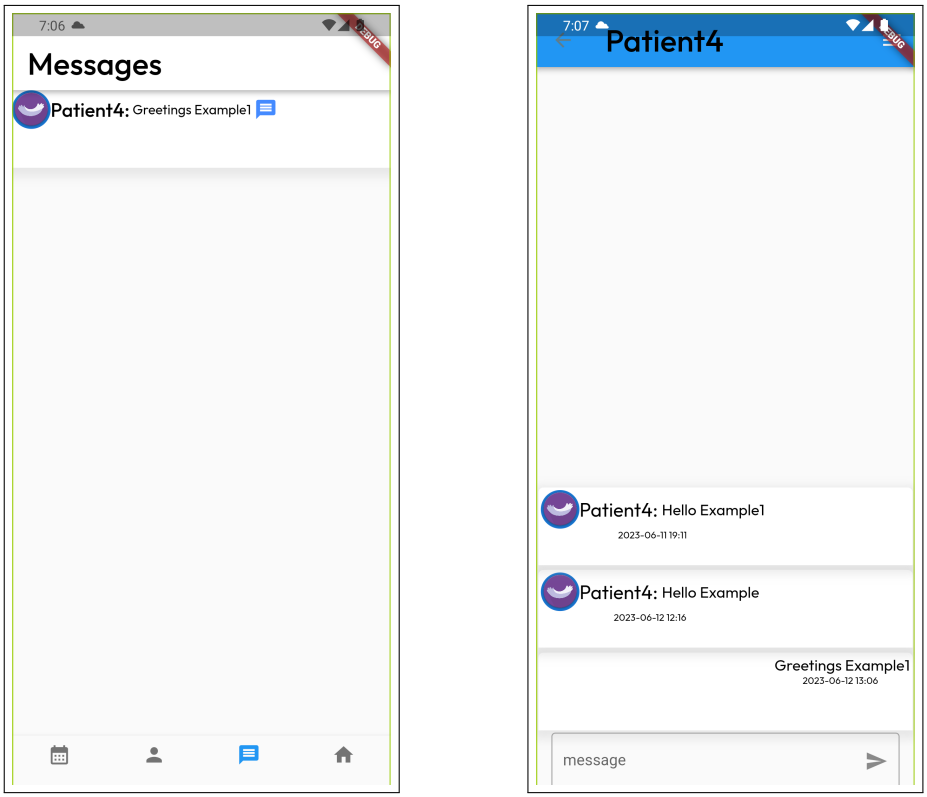

Figure 33 – *Therapist conversation list and interlocutor chat page interfaces*

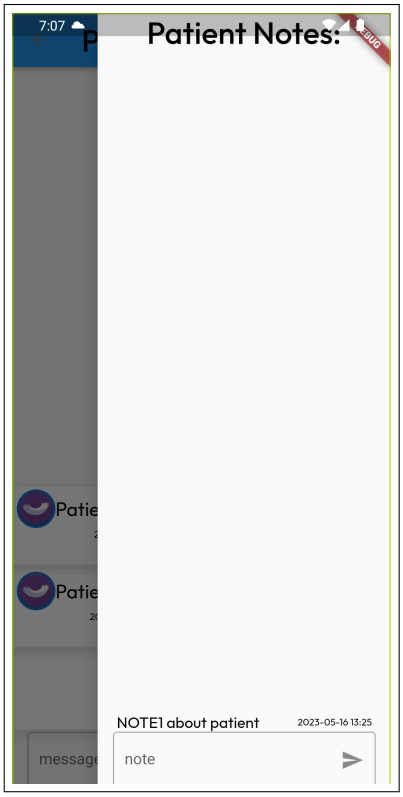

Figure 34 – *Therapist patient notes interface*

4.8.2 Patient messaging page interfaces

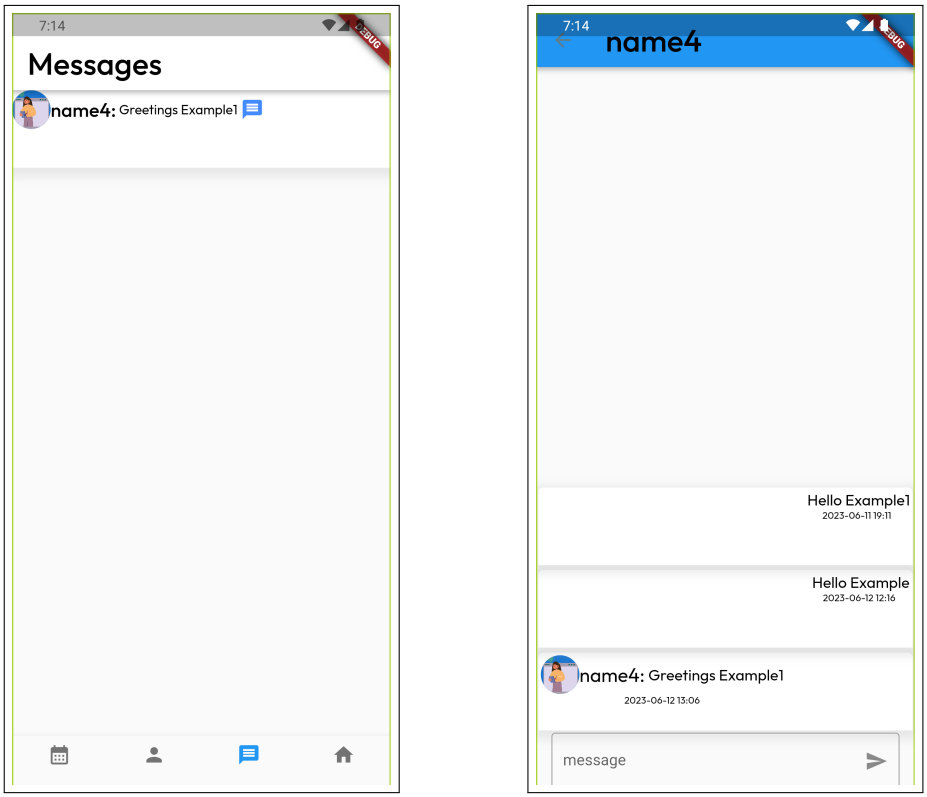

Figure 35 – *Patient conversation list and interlocutor chat page interfaces*

4.9 Web version interface examples

The app can be compiled into a web version and ran in a browser as shown in the figures below:

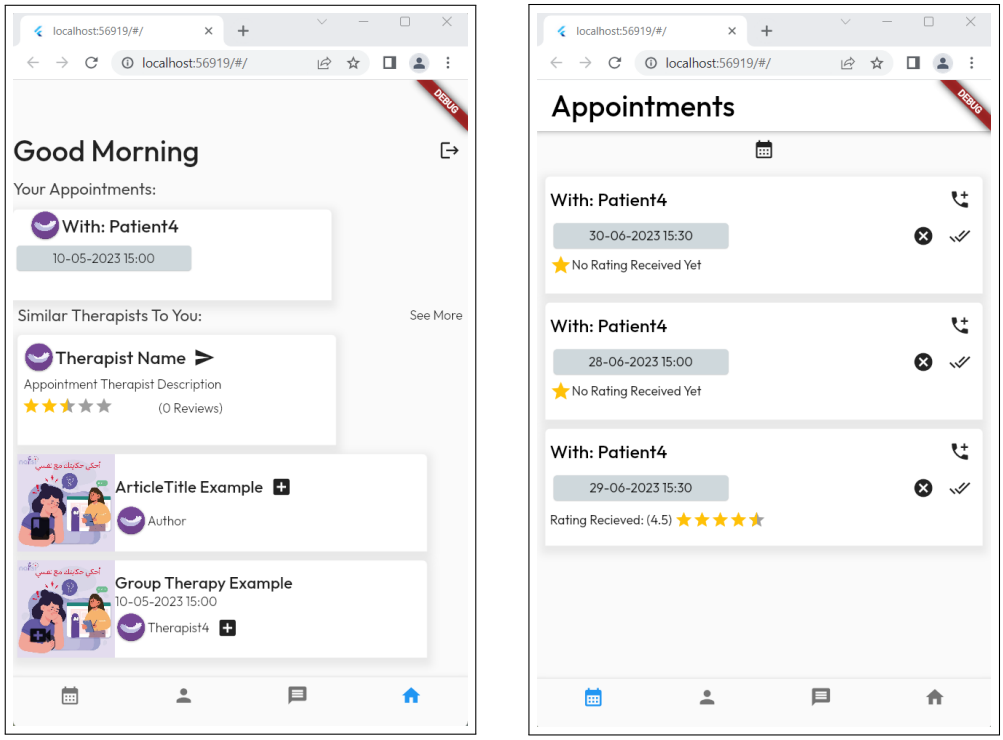

Figure 36 – *Web version homepage and appointment tab interfaces*

4.10 Interfaces of video call pages

The app includes different types of video communication interfaces, some examples use an emulator device and are shown below:

4.10.1 Interface of one-to-one video call page

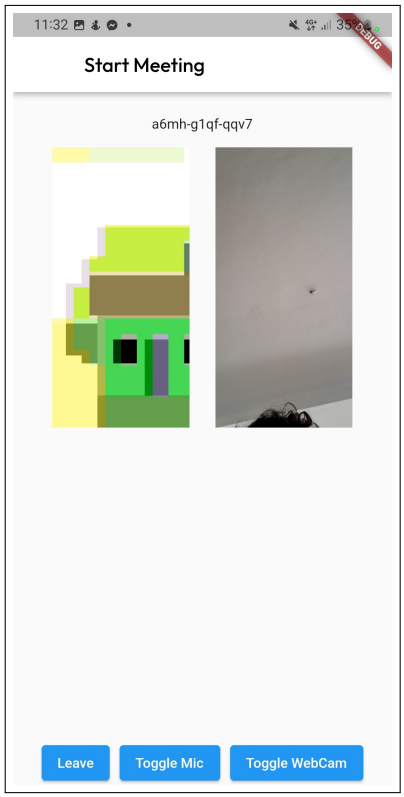

Figure 37 – *One-to-one video call interface*

4.10.2 Interfaces of group video call pages

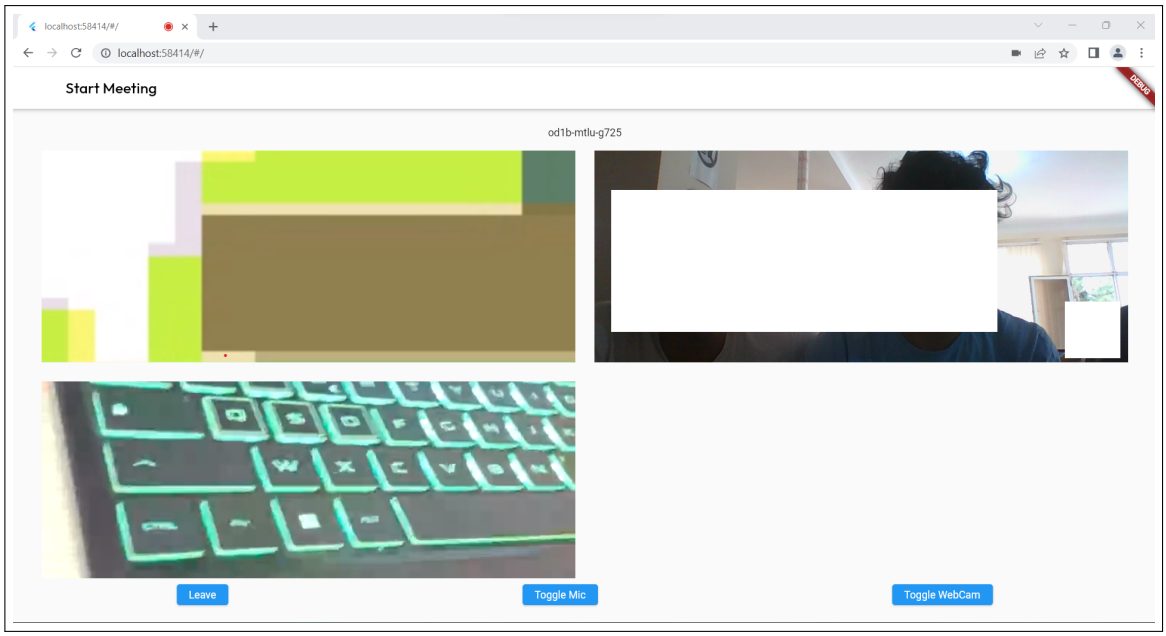

Figure 38 – *Group video call on a web version interface*

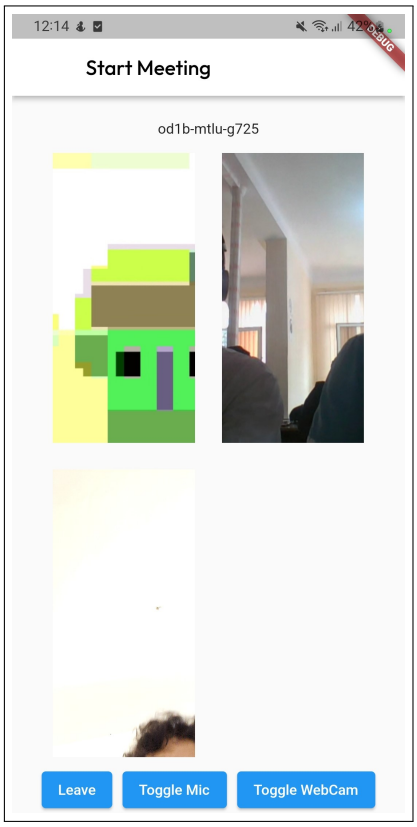

Figure 39 – *Group video call interface*

4.11 Interfaces of article pages

Patients can read articles and therapists can contribute their own with the community.

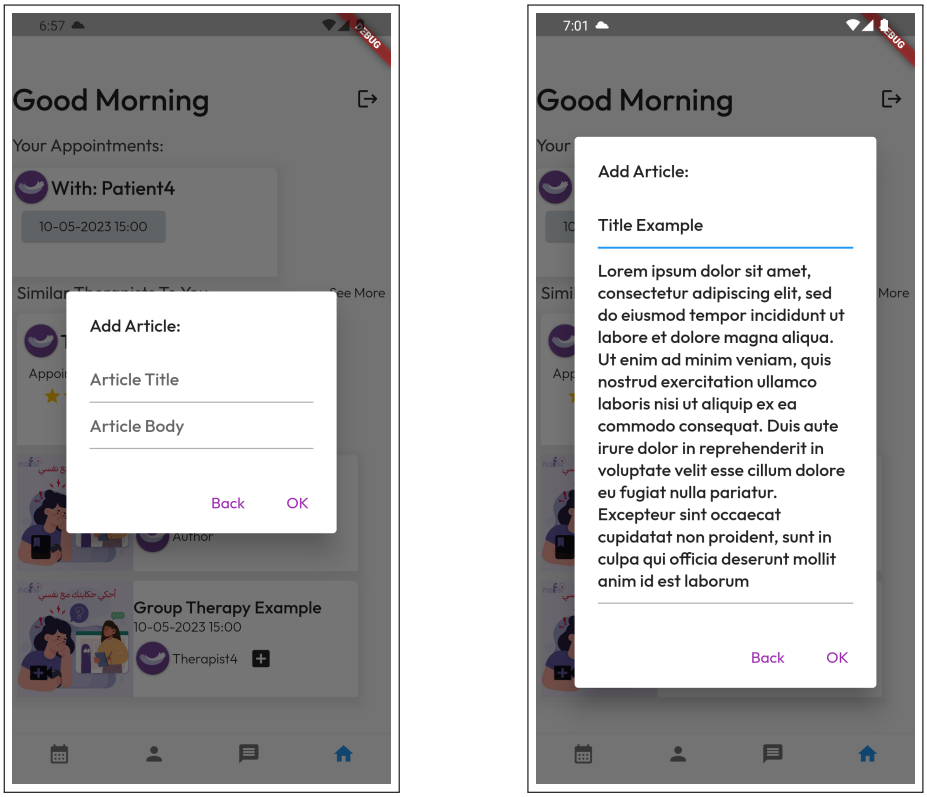

Figure 40 – *Therapist adding article interface example*

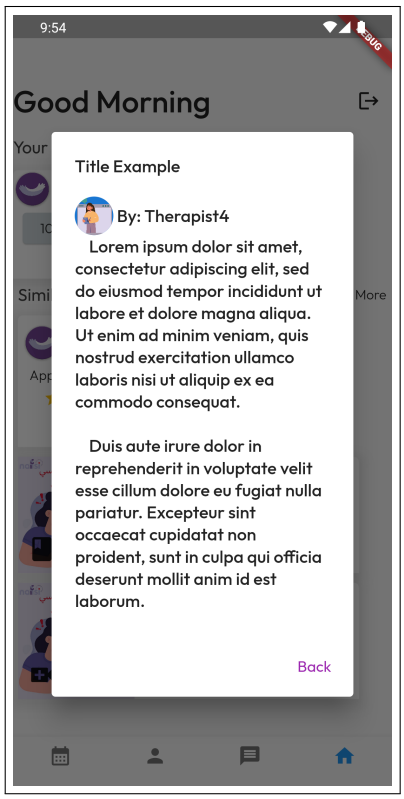

Figure 41 – *Article display interface example*

4.12 Conclusion

Within this chapter, we have presented the distinct graphical user interfaces tailored to the specific functionalities of both therapists and patients in the Naffsi application. The comprehensive overview provided demonstrates that the system actors can seamlessly utilize our application, benefiting from its intuitive and user-friendly design to effortlessly accomplish their desired tasks.

General conclusion

In this project, we have undertaken the design and implementation of the Naffsi app, a comprehensive mental health online counseling Flutter application. Our objective was to create a qualitative and accessible platform for individuals seeking mental health services.

Through careful analysis and consideration of user requirements, we have successfully developed a range of functionalities, including secure one-on-one video calls for remote therapy sessions, convenient and optionally anonymous access to professional mental health services, and the promotion of community engagement and peer support through livestreams and articles.

By leveraging widely-used technologies and applying UML diagrams in addition to frontend and backend design principles, we have ensured a robust and well-structured application. This project has not only expanded our knowledge and skills in software development but has also highlighted the importance of addressing mental health needs through innovative technological solutions.

Future works

In terms of future developments for the Naffsi app, there are several areas that can be explored to further enhance its functionality and user experience. These include:

Integration of e-payment functionality: Overcoming administrative constraints to implement an e-payment API would provide users with a convenient and secure way to make payments for the mental health services they receive through the app.

User interface (UI) enhancements: Improving the user interface by refining the layout, optimizing navigation, and incorporating intuitive design elements would create a more user-friendly and visually appealing experience.

Group livestreams: Introducing the capability for multiple therapists to host livestream sessions together for a larger audience would foster community engagement and enable users to participate in more informative conferences and conversations.

"isRead" feature for messages: Implementing an "isRead" indicator for messages would help users keep track of read and unread messages, improving communication and ensuring important messages are not overlooked.

User feedback and support: Establishing dedicated sections for user feedback and support would allow users to provide valuable feedback, report issues, and seek assistance when needed, further enhancing the app based on user input and providing timely support.

Additionally, implementing a reporting system would enable users to report instances of misbehavior or inappropriate conduct by other users, maintaining a safe and respectful community environment.

By focusing on these future works, the Naffsi app can continue to improve and hopefully provide users with enhanced mental health services, ensuring a comprehensive and user-friendly experience that supports individuals on their mental health journey.

References

[1] D. Becker, 'Acceptance of Mobile Mental Health Treatment Applications', *Procedia Computer Science*, t. 98, p. 220-227, 2016, The 7th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2016)/The 6th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2016)/Affiliated Workshops, ISSN : 1877-0509. DOI : https://doi.org/10.1016/j.procs.2016.09.036. adresse : https://www.sciencedirect.com/science/article/pii/S1877050916321652 (cf. p. 2).

[2] X. Wang, C. Markert et F. Sasangohar, 'Investigating Popular Mental Health Mobile Application Downloads and Activity During the COVID-19 Pandemic', *Human Factors*, t. 65, n° 1, p. 50-61, 2023, PMID: 33682467. DOI : 10.1177/0018720821998110. adresse : https://doi.org/10.1177/0018720821998110 (cf. p. 2).

[3] H. M. Kim, Y. Xu et Y. Wang, 'Overcoming the Mental Health Stigma Through m-Health Apps: Results from the Healthy Minds Study', *Telemedicine Journal and E-health*, mars 2022. DOI : 10.1089/tmj.2021.0418 (cf. p. 3).

[4] B. Aljedaani et M. A. Babar, 'Challenges With Developing Secure Mobile Health Applications: Systematic Review', *JMIR Mhealth Uhealth*, t. 9, n° 6, e15654, juin 2021, ISSN : 2291-5222. DOI : 10.2196/15654. adresse : http://www.ncbi.nlm.nih.gov/pubmed/34152277 (cf. p. 3).

[5] L. H. Iwaya, M. A. Babar, A. Rashid et C. Wijayarathna, 'On the Privacy of Mental Health Apps: An Empirical Investigation and its Implications for Apps Development', *CoRR*, t. abs/2201.09006, 2022. arXiv : 2201.09006. adresse : https://arxiv.org/abs/2201.09006 (cf. p. 3).

[6] T. Zohud et S. Zein, 'A Systematic Mapping Study of Cross-Platform Mobile Apps', *Journal of Computer Science*, t. 15, p. 519-536, avr. 2019. DOI : 10.3844/jcssp.2019.519.536 (cf. p. 4).

[7] techterms.com. 'Cross-platform defintion'. (2022), adresse : https://techterms.com/definition/crossplatform. (accessed: 31.05.2023) (cf. p. 4).

[8] A. Del Sole et D. Sole, *Visual Studio Code Distilled*. Springer, 2019 (cf. p. 6).

[9] L. Moroney, 'An Introduction to Firebase', in *The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform*. Berkeley, CA : Apress, 2017, p. 1-24, ISBN : 978-1-4842-2943-9. DOI : 10.1007/978-1-4842-2943-9_1. adresse : https://doi.org/10.1007/978-1-4842-2943-9_1 (cf. p. 7, 8).

[10] M. Belchin et P. Juberias, *Web Programming with Dart*. Apress, 2015 (cf. p. 7).

[11] Google. 'Dart Overview'. (2022), adresse : https://dart.dev/overview. (accessed: 31.05.2023) (cf. p. 7).

[12] A. Tashildar, N. Shah, R. Gala, T. Giri et P. Chavhan, 'Application development using flutter', *International Research Journal of Modernization in Engineering Technology and Science*, t. 2, n° 8, p. 1262-1266, 2020 (cf. p. 7).

[13] Google. 'Flutter presentation'. (2022), adresse : https://flutter.dev/. (accessed: 31.05.2023) (cf. p. 8).

[14] VideoSDK. 'VideoSDK Overview'. (2022), adresse : https://www.videosdk.live. (accessed: 31.05.2023) (cf. p. 8).

[15] Facebook. 'Facebook-Auth package overview'. (2020), adresse : https://facebook.meedu.app/docs/5.x.x/intro. (accessed: 31.05.2023) (cf. p. 8).

[16] Pub.dev. 'Google Sign-in package Overview'. (2018), adresse : https://pub.dev/packages/google_sign_in. (accessed: 31.05.2023) (cf. p. 9).

[17] Google. 'Firebase-auth Overview'. (2022), adresse : https://firebase.google.com/products/auth/. (accessed: 31.05.2023) (cf. p. 9).

[18] Pub.dev. 'Firebase-Core Overview'. (2020), adresse : https://pub.dev/packages/firebase_core. (accessed: 31.05.2023) (cf. p. 9).

[19] Google. 'Firebase-Cloud-Firestore Overview'. (2020), adresse : https://pub.dev/packages/cloud_firestore. (accessed: 31.05.2023) (cf. p. 9).

[20] Pub.dev. 'Firebase-Storage Overview'. (2020), adresse : https://pub.dev/packages/firebase_storage. (accessed: 31.05.2023) (cf. p. 9).

[21] Pub.dev. 'Firebase-Messaging Overview'. (2020), adresse : https://pub.dev/packages/firebase_messaging. (accessed: 31.05.2023) (cf. p. 10).

[22] Pub.dev. 'Calendar carousel package Overview'. (2020), adresse : https://pub.dev/packages/flutter_calendar_carousel. (accessed: 31.05.2023) (cf. p. 10).

[23] Pub.dev. 'ImagePicker package Overview'. (2020), adresse : https://pub.dev/packages/image_picker. (accessed: 31.05.2023) (cf. p. 10).

[24] FLUTTERBEADS.COM. 'Toast definition'. (2020), adresse : https://www.flutterbeads.com/show-toast-in-flutter/. (accessed: 31.05.2023) (cf. p. 10).

[25] PUB.DEV. 'FlutterToast package Overview'. (2020), adresse : https://pub.dev/packages/fluttertoast. (accessed: 31.05.2023) (cf. p. 10).

[26] PUB.DEV. 'Flutter-Local-Notifications package Overview'. (2019), adresse : https://pub.dev/packages/flutter_local_notifications. (accessed: 31.05.2023) (cf. p. 11).

[27] PUB.DEV. 'ImagePicker package Overview'. (2020), adresse : https://pub.dev/packages/videosdk. (accessed: 31.05.2023) (cf. p. 11).

[28] N. MEDVIDOVIC, D. S. ROSENBLUM, D. F. REDMILES et J. E. ROBBINS, 'Modeling Software Architectures in the Unified Modeling Language', *ACM Trans. Softw. Eng. Methodol.*, t. 11, nº 1, p. 2-57, jan. 2002, ISSN : 1049-331X. DOI : 10.1145/504087.504088. adresse : https://doi.org/10.1145/504087.504088 (cf. p. 12).

[29] M. OHYVER, J. V. MONIAGA, I. SUNGKAWA, B. E. SUBAGYO et I. A. CHANDRA, 'The Comparison Firebase Realtime Database and MySQL Database Performance using Wilcoxon Signed-Rank Test', *Procedia Computer Science*, t. 157, p. 396-405, 2019, The 4th International Conference on Computer Science and Computational Intelligence (ICCSCI 2019) : Enabling Collaboration to Escalate Impact of Research Results for Society, ISSN : 1877-0509. DOI : https://doi.org/10.1016/j.procs.2019.08.231. adresse : https://www.sciencedirect.com/science/article/pii/S1877050919311500 (cf. p. 38).

[30] C. KHAWAS et P. SHAH, 'Application of Firebase in Android App Development-A Study', *International Journal of Computer Applications*, t. 179, p. 49-53, juin 2018. DOI : 10.5120/ijca2018917200 (cf. p. 38).

[31] GOOGLE. 'Firestore read and write scale'. (2019), adresse : https://firebase.google.com/docs/firestore/understand-reads-writes-scale. (accessed: 31.05.2023) (cf. p. 38).