App Name: ChatPLus

Developer: Alaa Eddine Boubekeur

Year of Development: 2024

Tech Stack: React – NodeJS – Firebase:

Platform: Web

Runnable format: PWA (Progressive Web App)

Period of Development: 3 months

Preview Link: ChatPlus (chatplus-a3d50.firebaseapp.com)

1. Introduction

1.1 Purpose

The purpose of this document is to outline the technical blueprint for ChatPlus – Call Services, messaging services, web notifications services, installability services, media messaging services.

Outline the scope of this document which details the design specifications of the software application referenced in the Preliminary Design Document (PDD).

1.3 Definitions, Acronyms, and Abbreviations

- **ChatPlus: Name of our web application**
- **AI: Artificial Intelligence.**
- **API: Application Programming Interface.**
- **CPU: Central processing unit.**
- **RAM: Random_Access Memory.**
- **GB: GigaByte.**
- **JS: JavaScript programming language**
- **ReactJS: frontend JS framework**
- **NodeJS: server side engine for JavaScript**
- **ExpressJS: Library for server side rendering and APIs for NodeJS**
- **Firebase: a Cross platform app development SDK**
- **Firestore: a NoSQL real time database from firebase**
- **Firebase hosting: a Frontend hosting service from firebase**

- **Google Cloud:**
- **FR: Functional Requirement.**
- **HTTP: HyperText Transfer Protocol.**
- **HTTPS: HyperText Transfer Protocol Secure.**
- **IDE: Integrated Development Environment.**
- **iOS: iPhone Operating System.**
- **Json: JavaScript Object Notation.**
- **MVC: Model-View-Controller.**
- **Nav bar: navigation bar.**
- **SRS: Software Requirement Specification.**
- **UI: User Interface.**
- **UX: User Experience.**
- **VSC: Visual Studio Code.**
- **SDD: Software design document.**
- **URL: Uniform reference locator.**
- **HTML: HyperText Markup language.**
- **PER: Performance.**
- **IR: Integration.**
- **SEC: Security.**
- **CI/CD: Continuous integration/ Continuous delivery.**
- **Web push notification: a Client side API provided by browsers to receive notifications**

- **Service worker: a JS file that allows us to handle web applications on the background when they are not loaded, it’s used for caching, installing and pushing notifications**
- **Daily API: a server side api that allows us to create real time video and audio calls.**
- **PWA: progressive web application is a web app that can be installed from the browser on all platforms and can run and look like a standalone app.**
- **Material UI: Material UI is an open-source React component library that implements Google's Material Design. It's comprehensive and can be used in production out of the box.**

2. Architectural Design

2.1 Software Architecture

Our app follows the MVC software architecture, MVC (Model-View-Controller) is a pattern in software design commonly used to implement user interfaces, data, and controlling logic. It emphasizes a separation between the software's business logic and display. This "separation of concerns" provides for a better division of labor and improved maintenance.

ChatPlus gives a cross platform experience as its frontend is designed to be a PWA application which has the ability to be installed everywhere and act like a standalone app, with its feature like push notifications, ChatPlus is the definition of lightweight app that can give the full features of a mobile with web technologies.

Major Components and Module Decomposition

User interface layer (View)

It consiste of multiple react view components:

MaterialUI: Material UI is an open-source React component library that implements Google's Material Design. It's comprehensive and can be used in production out of the box, we're gonna use it for icons buttons and some elements in out interface like an audio slider.

LoginView: A simple login view that allows the user to put their username and password or login with google.

SidebarViews: multiple view components like 'sidebar__header', 'sidebar__search', 'sidebar menu' and 'SidebarChats', it consists of a header for user information, a search bar for searching users, a sidebar menu to navigate between your chats, your groups and users, and a SidebarChats component to display all your recently chats that you messaged, it shows the user name and photo and the last message.

ChatViews: it consists of many components as follow:

1.'chat__header' that contains the information of the user you're talking to, his online status, profile picture and it displays buttons for audio call and video call, it also displays whethere the user is typing.

2.'chat__body--container' it contains the informations of our messages with the other users it has component of messages which displays text messages, images with their message and also audio messages with their information like audio time and wethere the audio was played and at the end of this component we have the 'seen' element that displays whether messages were seen.

3 'AudioPlayer': a React component that can display to us an audio with a slider to navigate it, displays the full time and current time of the audio, this view component is loaded inside 'chat__body--container'.

4 ‘ChatFooter’: it contains an input to type a message a button to send a message when type on the input otherwise the button will allow you to record the audio, a button to import images and files.

5 ’MediaPreview’: a React components that allows to preview the images or files we have selected to send in our chat, they are displayed on a carousel and user can slide the images or files and type a specific message for each one

6 ‘ImagePreview’: When we have images sent on our chat this componen will display the images on full screen with a smooth animation, component mounts after clicking on an image.

scalePage: a view function that increases the size of our web app when displayed on a large screens like full HD screens and 4K screens.

CallViews: a bunch of react components that constain all calls view element, they have the capability to be draged all over our screen and it consists of:

1 ‘Buttons’: a call button with a red version of it and a green video call button.

2 ‘AudioCallView’: a view component that allows to answer incoming audio call and display the call with a timer and it allows to cancel the call.

3 ‘StartVideoCallView’: a view component that display a video of ourselves by connecting to the local MediaAPI and it waits for the other user to accept the call or, it displays a button for us to answer an incoming video call.

4 ‘VideoCallView’: a view component that displays a video of us and the other user it allows to switch cameras, disable camera and audio, it can also go fullscreen.

RouteViews: React components that contains all of our views in order to create local components navigation, we got 'VideoCallRoute', 'SideBarMenuRoute' and 'ChatsRoute'

Client Side Models (Model):

Client side models are the logic that allows our frontend to interact with databases and multiple local and serverside APIs and it consists of:

Firebase SDK: It's an SDK that is used to build the database of our web app and it consists of:

1.'Firebase Firestore': Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud, we will use it to store all of our data in our web app

2.'Firebase Authentication': Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to our app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more, in our web app we use it for google authentication and email authentication

3.'Firebase Storage': Cloud Storage for Firebase is built on fast and secure Google Cloud infrastructure for app developers who need to store and serve user-generated content, such as photos or videos.we use it in our web app to store audios and images sent in chats

AppModel: A model that generates a user after authenticating and it also makes sure that we have the latest version of our web assets.

ChatModels: it consists of model logics of sending messages to the database, establishing listeners to listen to new messages, listening to

whether the other user is online and whether he's typing, it also sends our media like images and audios to the database storage,

SidebarChatsModel: Logic that listens to the latest messages of users and gives us an array of all your new messages from users, it also gives number of unread messages and online status of users, it also organize the users based on time of last message.

UsersSearchModel: Logic that searchs for users on our database, it uses algolia search that has a list of our users by linking it to our database on the server

CallModel: Logic that uses the Daily SDK to create a call on our web app and also send the data to our server and interacts with DailyAPI.

Client Side Controllers (Controller):

It consists of React components that link our views with their specific models:

App Controller: Links the authenticated user to all components and runs the scalePage function to adjust the size of our app, it also loads firebase and attaches all the components, we can consider it a wrapper to our components.

SideBarController: Link users data and list of his latest chats, it also links our menus with their model logic, it also link the search bar with algolia search API.

ChatController: this is a very big controller that link most of the messaging and chat features.

CallController: Links the call model with its views.

Server Side Model:

Not all features are done on the frontend as the SDKs we used require some server side functionalities and they consist of:

CallServerModel: Logic that allows us to create rooms for calls by interacting with Daily API and updating our firestore databse

TranscriptModel: Logic on the server that receives an audio file and interacts with Google Cloud speech to text API and it gives a transcript for audio messages

Online Status Handler: A listener that listens to the online status of users and updates the databse accordingly

Notification Model: A service that send notification to other users

AlgoliaSaver: A listener that listens to new users on our database and updates algolia accordingly so we can use it for the search feature on the frontend.

Server Side Controllers:

CallServer: an API endpoint that contains callModel

Worker: a worker service that runs all our firebase handling services

2.2 Hardware/Software Mapping

In this section we will explain the distribution and communication of "ChatPlus" application's software components across hardware resources.

Hardware Resources:

Client Side: this application a web application designed to be a PWA (progressive web app) so it can run on all devices with their browsers and it can be installed as standalone app with the browser

Server Side: It consists of Firebase Firestore that contains all our database like users and chats as well as backend services to interact between our frontend and some APIs

Software Components:

Firebase Hosting: Our frontend app will be accessed by a URL because it will deployed to a server with firebase hosting.

Communication Protocols:

HTTPS: The primary protocols for communication between the web clients and the servers, securing the interactions between the client and server especially for API interactions to retrieve and send data.

Firestore Realtime Database Implemented for real-time live updates of users information or chats and calls interactions between users during runtime of application to ensure dynamic content delivery.

2.3 Data Storage and Access

Storage: Our web app stores all its data into Firestore database and it also stores our users data into Algolia database as well in order to be able to use its search functionality, our web app stores its assets through cache with the browser, it can also be installed.

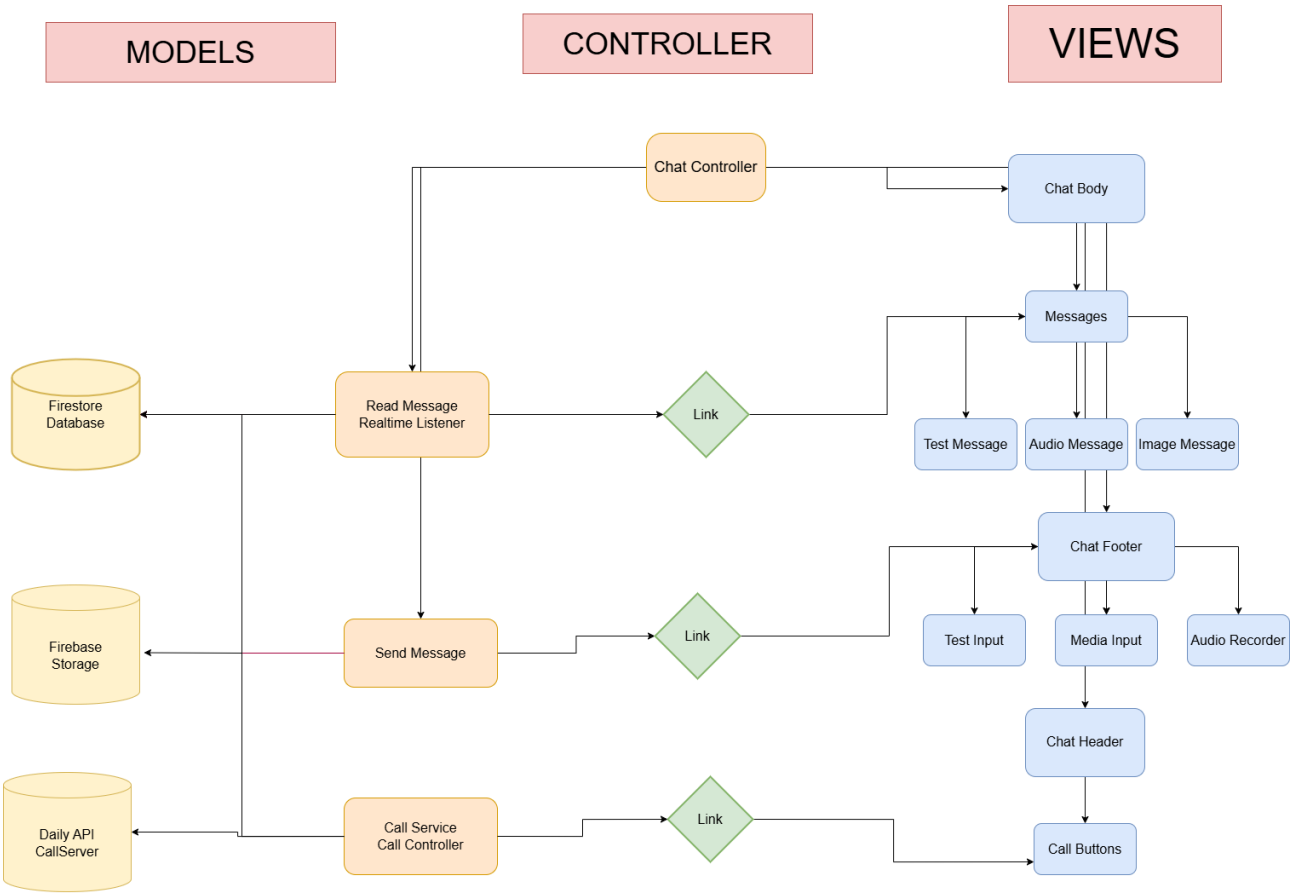
Access: Accessing our web app will be through a URL and our web app access the backend by two main ways:

1.API endpoint: our API endpoint allows the web app to set up calls and transcript audios

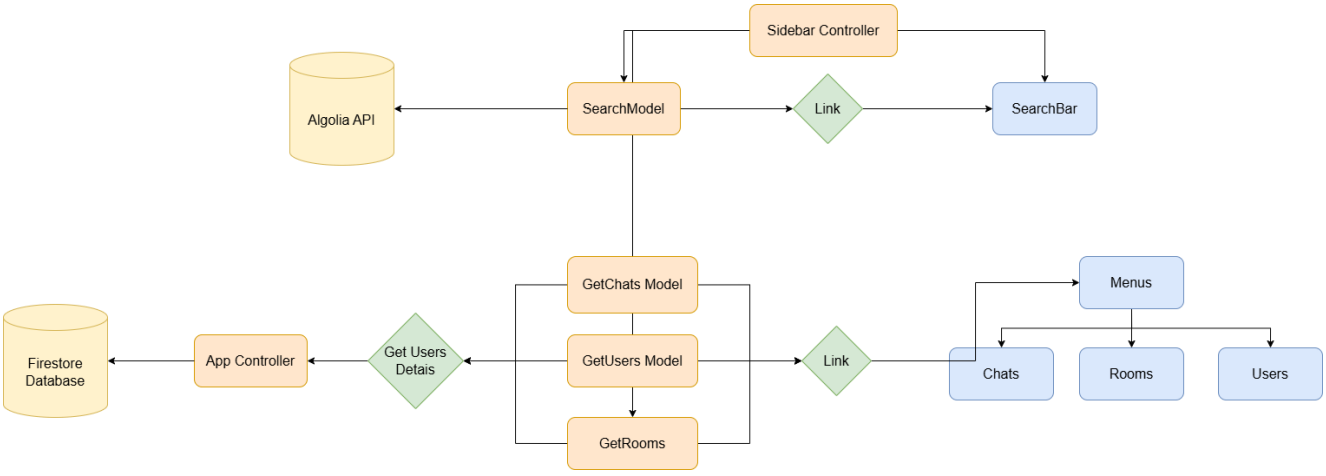
2.SDKs: firebase, Algolia and Daily provide us with an SDK that allows us to access all its services

3. Detailed Design of Flow Chart and entities:

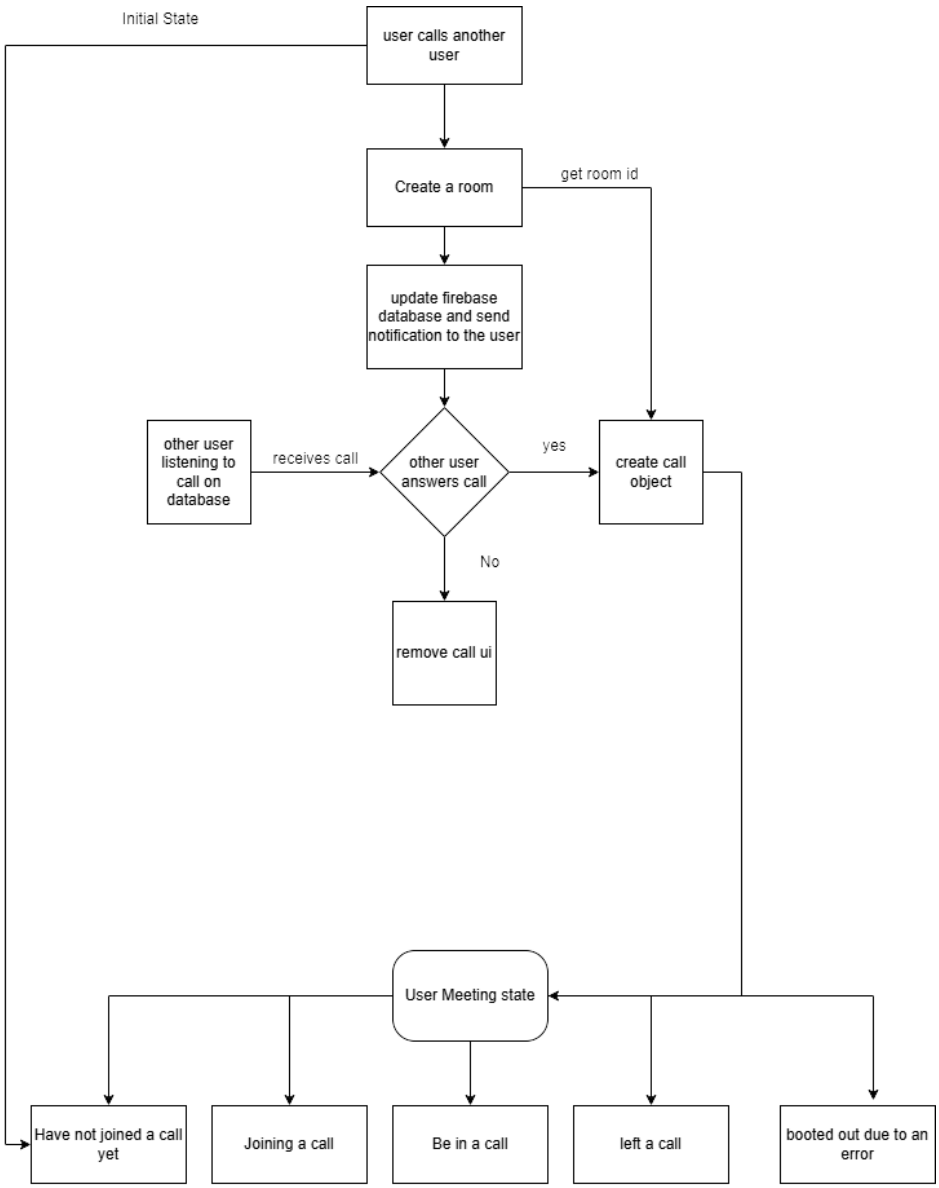
Chat Flow Chart:



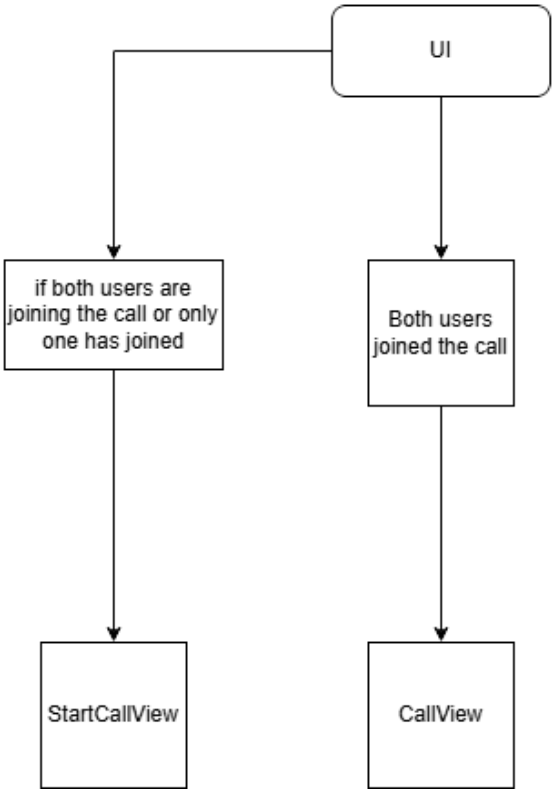
Sidebar Flow Chart



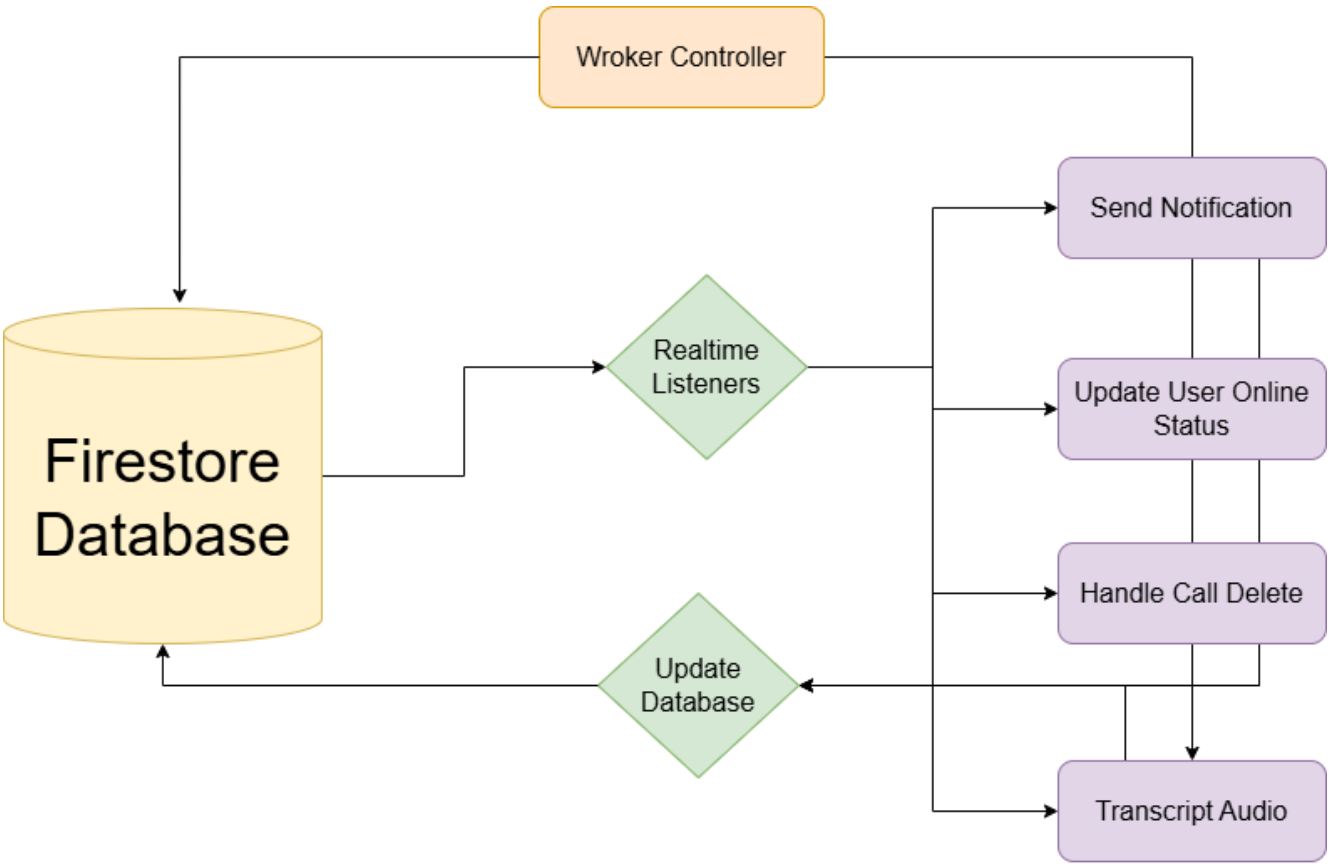
Call Flow Chart for the Backend:



Call Flow for Frontend:



Backend Worker Flow Chart



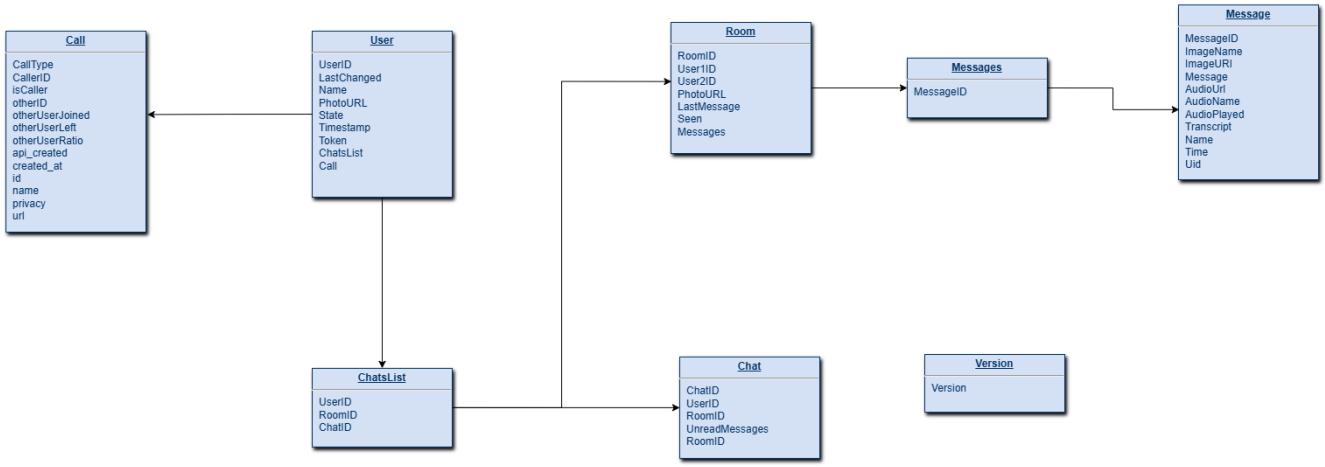
4. Data Design

Database Design

Our Web app uses Firestore for storing our database which is a Firebase NoSql database, we store users information, we store a list of all messages, we store list of chats, and we also store chats on rooms, these are the data stored on our database:

- **Users Data after Authentication**
- **Rooms that contain all the details of messages**
- **List of latest chats for each user**

- **List of notifications to be sent**
- **List of audio to be transcripted**



Data Dictionary:

“User”

- **UserID: <string> the id created for the user**
- **LastChanged: <string> the last time the user changed his information**
- **Name: <string> name of the user**
- **PhotoURL: <string> the URL of the user profile picture**
- **State: <string> “offline” or “online” the online status of the user**
- **Timestamp: <string> the time user was created**
- **Token: <string> browser notification token**
- **ChatsList: <List of Entities> list of recent chats of the user**
- **Call: <string> only “call” the data of the call the user is involved in, also there could be only one call at a time.**

“Call”

- **CallType: <string> only “video” or “audio” defines the type of call**
- **callerID: <string> ID of the user who initiated the call**
- **isCaller: <Boolean> true if you’re the caller, false if you’re receiving a call**
- **otherUserJoined: <Boolean> it’s false until the other user called joins the call**
- **otherUserLeft: <Boolean> it’s false until the other user in the call leaves the call**
- **otherUserRation: <Number> other user’s camera ratio**
- **createdAt: <string> time the call was created**
- **ID: <string>: id of the call created**
- **Name: <string> name of the call created**
- **Privacy: <string> public or private**
- **URL: <string> link to the video call**

“ChatsList”

It’s a list of entities that relate to the room of the chat and at the same time the details of the chat

- **UserID: <string> user’s id**
- **ChatID: <string> the id of the chat of the user**
- **RoomID: <string> the id of the room that contains all the details of the messages**

“Chat”

- **ChatID: <string> id of the chat**
- **UserID: <string> id of the user**
- **RoomID: <string> id of the room that contains the chat**
- **UnreadMessages: <number> number of unread messages**

“Room”

It’s the entity that contains the details of our room and relate to a list of messages

- **RoomID: <string> id of the room**
- **UserID1: <string> id of the first user**
- **UserID2: <string> id of the second user**
- **PhotoUrl: <string> URL of the profile picture of the other user**
- **LastMessage: <string> last message in the chat**
- **Seen: <Boolean>: true if the other user sees the last message**
- **Messages: <string> link to the list of messages**

“Message”

- **MessageID: <string> id of the message**
- **ImageName: <string> name of the image in the message**
- **ImageURl: <string> URL of the image**
- **Message: <string> Text of the message**
- **AudioUrl: <string> URL of the audio**
- **AudioName: <string> Name of the audio**

- **AudioPlayed: <Boolean>: true if a user plays the audio for the first time**
- **Transcript: <string> Transcript of the audio after applying speech to text on the backend**
- **Name: <string> name of the user sending the message**
- **Time: <string> time message was sent**
- **Uid: <string> id of the user sending the message**

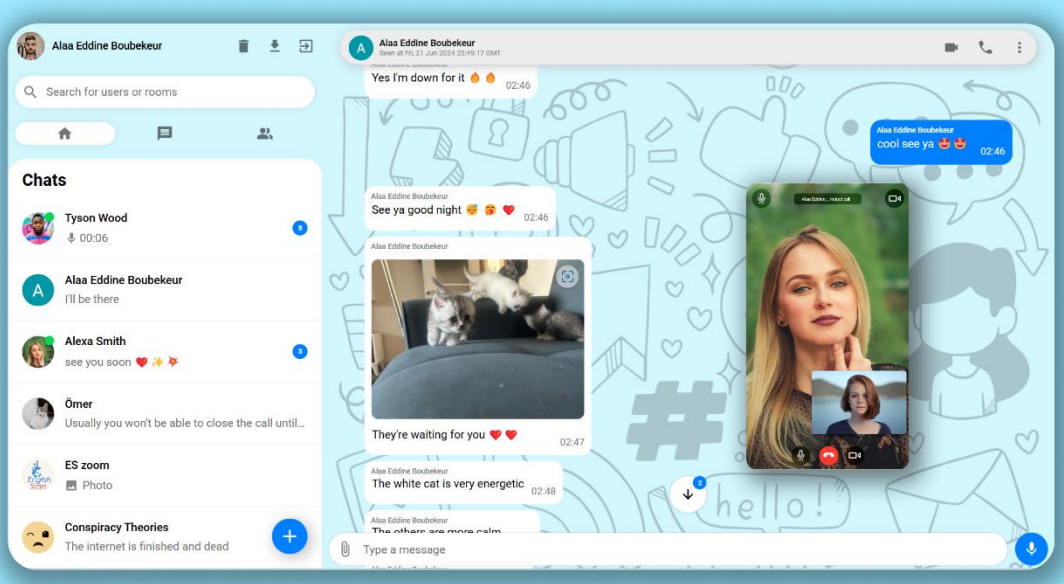
5. Human Interface Design

In order to develop our web app quickly without losing time on design since we only have two months for development and we want our web app to have many feature, we decided to simply copy the whatsapp user interface to quickly create it and focus on building the functionalities, in the next month we're going to create a new user interface for our web.

5.1 Home page:

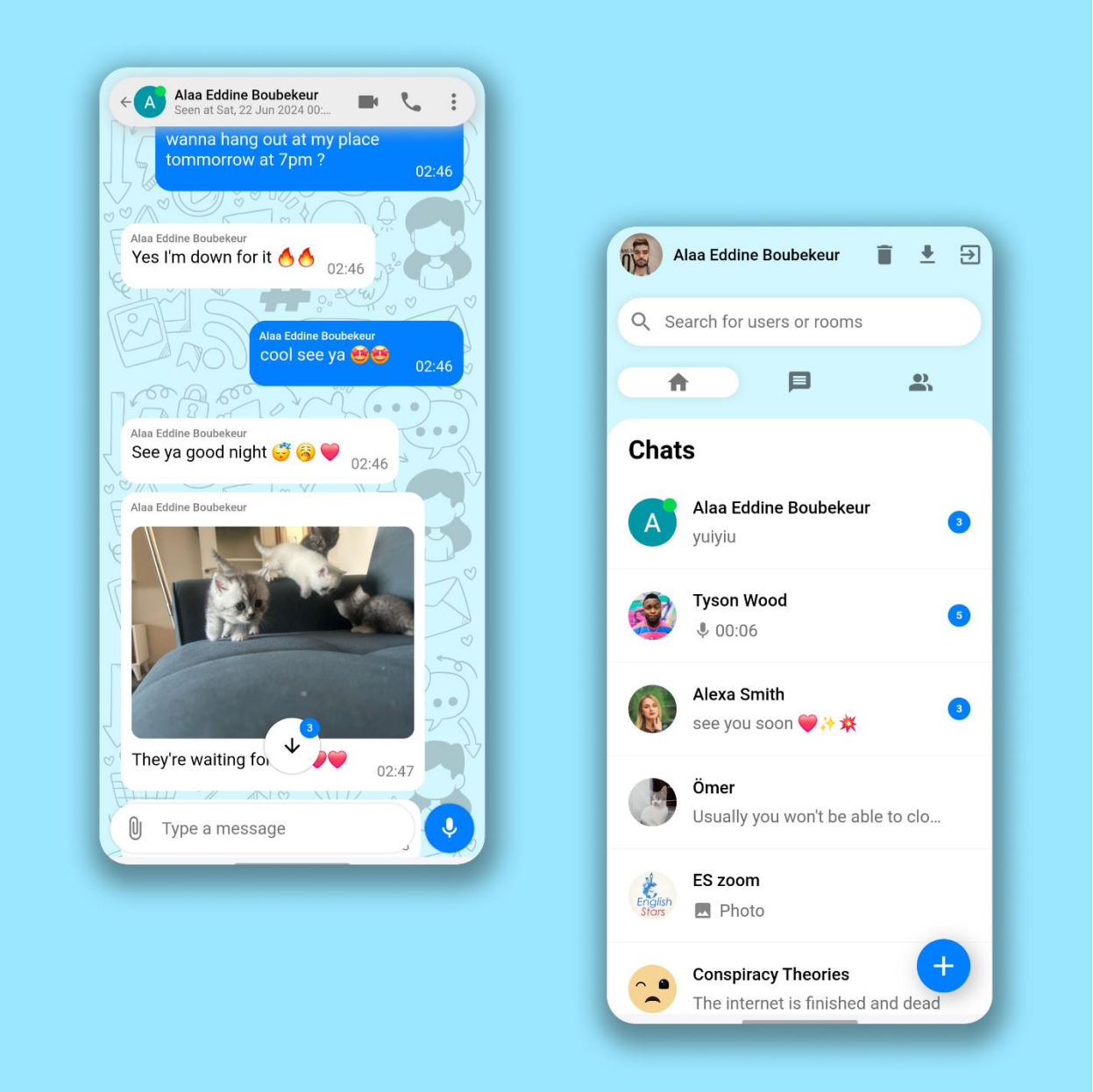
The home page on desktop display a sidbar and a chat, user can select chats on the sidebar, he can also check all users and search for them, when user clicks on a chat, the messages room will be displayed where he can interact with the other user.

On every chat user can click on the call button to initiate a call with the other user, when a user receives a call, this incoming call interface will be displayed and when user accepts the call, he will be able to see the other user and interact with him



On mobile phones, our web app will change by separating the sidebar and chats, user can still interact easily, user can see the last message of each chat and number of unread messages, and can also see their online status.

On the chat user can click on audio messages to listen to them, he can also click on images to expand them and see them on fullScreen.



5.3 Navigation Flow

Menus

- **Located in the Sidebar**
- **Contains a list of chats from each menu**
- **Menus are chats, rooms and users**
- **User can navigate through the 3 menus**

- **User can click on any chat to go to its messages room**

Sidebar Search

- **Located at the top of the sidebar**
- **User can type the name of a user to search him**
- **User can click on the users from the search result and it will take to the room of chat**

Chat

- **Displays after a user clicks on a chat**
- **User can scroll to see all messages**
- **If a user clicks on an image than it will navigate to a page with the image on full screen**
- **User can click on the back button to go back to the sidebar on mobile, or to the previous chat on desktop**

5.4 Functionalities and User Interactions

- **Messaging users in real time.**
- **User can text messages.**
- **User can send Images.**
- **User can send an audio message.**
- **User click on an image sent in a chat and it will smoothly animate to the center so user can see it.**
- **User can see if the other user is typing or recording.**
- **User can record an audio and send it.**
- **User can delete the conversation.**

- **User can search for users.**
- **User can see the online status of users.**
- **User can see the unread messages.**
- **User will have a seen at the bottom of a chat room if the other user saw his message.**
- **User have an arrow button that allows user to scroll down the chat, user also sees the number of unread messages in it.**
- **User can fetch 30 messages in a chat and if user scrolls up user will fetch more messages.**
- **The audio slider is grey when user sends an audio message and is green when user receives one and becomes blue if the receiver plays the audio.**
- **The audio player allows the user to see the full time of the audio and if user plays it, he will see the current time.**
- **User can receive notifications if a user sent him a message.**
- **A sound is played when user sends a message or receives one in a conversation.**
- **Another sound is played if user receives a message from another user.**
- **User can click on the arrow down button at the home page and the web app is installed in his device.**
- **User can click on the call button and send a call to another user**
- **User can receive calls from other users**
- **When on an incoming call user can accept or reject a call**
- **When on a call user can turn off and on his camera, and audio.**

- **When on a call user can go full screen to only see the call**
- **User can drag the call element all around the page**
- **User can exit a call.**

6. Resource Estimate

6.1 Hardware Requirements

Our web application is designed to run on all web browsers on any operating system though there are some requirements as follow:

- **Make sure you have the latest version of your browser**
- **Safari on IOS must be at least 16.5 to run the app perfectly**
- **User must accept notifications on browser to be able to receive them**
- **User must also accept installing the app on browser to use it as a standalone app**

6.2 Software Requirements

Operating system

Web app can run on all operating systems as long as they have the latest version of their browsers.

It is advisable to use a chromium browser to get the best performance on the web app

Supporting Software

- **JavaScript: It is the main programming language of our web app**
- **React: The frontend framework used to build our interface**

- **Visual Studio Code: the IDE use to develop our web application**
- **Firebase: A cloud service that provided us with services and SDKs to create a database for our web app and host it**
- **Railway: The platform used to deploy our backend**
- **Algolia: Cloud service that provided us with an API to search our users**
- **DailyAPI: WebRTC cloud service that provided us with an API and an SDK to easily create calls**
- **Github: Cloud platform used to upload our code and monitor its versions**
- **Google Cloud: Cloud service that we use to do the speech to text functionality to transcript our audio messages**
- **ExpressJS: Backend framework used to create our web server and worker**

Libraries:

- **Material UI: Used to quickly create an interface for our web app, it provided us with react elements, icons and widgets**
- **React-Router: React library that we used to create navigation locally on our web app**
- **AnimeJS: JS animation library that we used to run animations on our web app**
- **PDFJS: JS library that allows us to read a pdf and display the first page as an image**
- **NanoID: JS library to generate IDs**

- **CompressorJS JS library used to compress the size of images before sending them**

7. Security Design

7.1 Security Requirements

- **API security: Our APIs are all protected by a key**
- **SDK security: Our SDKs like Firebase and Algolia are protected by a configuration file and keys**
- **Web protocol protection: Our web app is hosted with an HTTPS server**
- **Database access: Firebase allows us to set security rules for accessing our data, and we specified the rules to allow access to authenticated users only**

8. Maintenance and Support

8.1 Maintenance Requirements

Our web app is maintained by firebase and google cloud, we can access all our data in the console, our code is also hosted on a Github repository, our frontend assets are maintained by firebase hosting, and our backend is maintained by Railway server

9. References

- **Firebase** Firebase | Google's Mobile and Web App Development Platform
- **React** React
- **React-Router** Home v6.23.1 | React Router

- **MaterialUI** MUI: The React component library you always wanted
- **DailyAPI** Daily | API documentation
- **Algolia** AI search that understands | Algolia
- **NanoID** ai/nanoid: A tiny (124 bytes), secure, URL-friendly, unique string ID generator for JavaScript (github.com)
- **CompressorJS** Compressor.js (fengyuanchen.github.io)
- **AnimeJS** anime.js • JavaScript animation engine (animejs.com)
- **PDFjs** PDF.js (mozilla.github.io)
- **Google Cloud Speech to Text** Speech-to-Text AI: speech recognition and transcription | Google Cloud
- **NodeJS** Node.js — Run JavaScript Everywhere (nodejs.org)
- **ExpressJS** Express - Node.js web application framework (expressjs.com)
- **Railway** Railway
- **Visual Studio Code** Visual Studio Code - Code Editing. Redefined

Test Plan Document

1. Introduction

This document outlines the testing strategy for "ChatPlus". The testing scope includes functional, performance, security, and integration aspects of the application. The main goal is to verify that the features of the application are working according to the requirements specified in the documentation. This verification process will guarantee that the application is reliable, secure, and operates effectively in various scenarios.

2. Features to be Tested

Functional Requirements

- **<u>Sidebar Functionalities</u>**

- **Test whether user can search for all users, and can get no results if there are no users**
- **Test whether navigation works without any problems**
- **Test if list of recent chats of user can displayed filtered by the time of last message, new chats should appear at the top**
- **Test if chats display the correct number of unread messages**
- **Test if chats display the correct format of last message**

- **<u>Chats Functionalities</u>**

- **Test if user can send messages**
- **Test if user can see the typing and recording status of the other user**
- **Test if user can record audio and send it**
- **Test if audio slider's color changes when the receiver of the audio plays it for the first time**
- **Test if audio transcript is working**
- **Test if user can send images**
- **Test if user can display images on full screen**
- **Test if user can send multiple images with a message for each one**
- **Test if user can click on the scroll button and the page will go down**

- **Test if user can see number of unread messages on the scroll button**
- **Test if user can see "seen" at the bottom of the chat room when the other user sees last message**

➢ **<u>Call Functionalities</u>**

- **Test if users can call other users**
- **Test if users can receive incoming calls**
- **Test if users can reject incoming calls**
- **Test if users can accept calls**
- **Test if user can drag the call element without lag around the page**
- **Test if the user can switch on and off his camera and audio**
- **Test if the user can go full screen on the call**

➢ **<u>Backend Functionalities</u>**

- **Test if user receives notifications when another user sends him a message**
- **Test if the notification come with right format**
- **Test if audio messages are converted to a text by our backend service**
- **Test if online status of users is updated**
- **Test if Calls are handled by deleting them when one of the 2 users in a call loses connection**
- **Test if algolia database is updated for search ability**

Performance Benchmark

- **<u>UI Performance</u>**

- **Assess the intuitiveness and clarity of the application's interfaces through user testing and user experience evaluation.**
- **Test the responsiveness of our web app on multiple screens**
- **Test the istallability of our web app on local devices.**
- **Test the smoothness of animations in our web app.**

- **<u>App Performance</u>**

- **Test if data of users chats and messages are being fetched at small time**
- **Test if writing to our database like sending messages, and updating user info happens at a small time**
- **Test if our assets are downloaded at a short time when accessing our web app.**
- **Test if our backend responds quickly to our changes**
- **Test the performance of our call functionalities, quality of video and audio should be high**

Security Features

- **It is imperative to guarantee that API communication is secure by utilizing HTTPS and appropriate authentication techniques.**

- **Additionally, it is crucial to validate that the application seeks and obtains explicit user consent prior to retrieving any data from the device.**
- **Lastly, ensure the secure storage of API credentials by confirming that they are stored in a secure manner and are not hardcoded into the application's source code.**

Integration Requirement

- **Verify the seamless and uninterrupted integration of the call service within the application's user interface, guaranteeing its proper functionality within the app's context.**
- **Validate that all external APIs deliver precise and timely responses as per the application's requirements.**
- **Validate the dependability of automated build and deployment procedures.**
- **Conduct integration testing between the call, users, chats, and messages to ensure consistent data across different modules.**

3. Approach

Unit Testing

- **Sidebar: Verify if the system allows users to search for all users and returns no results if there are no users found, Validate the smooth functioning of the navigation feature without encountering any issues, Confirm if the list of recent chats for a user can be filtered based on the time of the last message, ensuring that new chats appear at the top, Ensure that the**

number of unread messages is accurately displayed in the chat interface, Validate if the last message in the chats is presented in the correct format as expected.

- **Chats: Verify whether the user is able to send messages, Check if the user can view the typing and recording status of the other user, Test if the user can record audio and successfully send it, Validate if the color of the audio slider changes when the receiver plays the audio for the first time, Ensure that the audio transcript is functioning properly, Test if the user can send images, Verify if the user can view images in full screen mode, Test if the user can send multiple images, with a separate message for each image, Check if the user can click on the scroll button to navigate down the page, Verify if the scroll button displays the number of unread messages, Test if the user can see the "seen" indicator at the bottom of the chat room when the other user views the last message.**
- **Call: Check if users are able to make calls to other users, verify if users are able to answer incoming calls, Confirm if users can decline incoming calls, Ensure users can pick up calls, Test if the user can move the call element smoothly across the page, Validate if the user can toggle their camera and audio on and off, See if the user can go into full screen mode during the call.**
- **Backend Units: Check if the user gets notifications when another user sends a message, Verify that the notifications are in the correct format, Test if our backend service converts audio messages to text Ensure that the online status of users is being updated, Confirm that calls are deleted when one user loses connection, Validate if the Algolia database is updated for search functionality.**

Integration Testing:

- **Call Sever and Chats: Verify that the call service is very well integrated with our application, make sure it is running smoothly and it doesn't interrupt other functionalities in our web app**
- **APIs and Application Interface: Ensure that the external APIs are seamlessly integrated with the application, ensuring the provision of precise data to the user interface.**

System Testing

- **Full Application Workflow: The complete application workflow involves conducting thorough tests that encompass all features in order to guarantee the flawless operation of the entire system**
- **Performance: Evaluate the overall functionality of the application, with a specific emphasis on the responsiveness of the user interface and the interaction of the chatbot when subjected to heavy usage, in order to verify that the system meets the predetermined performance standards.**

Acceptance Testing:

- **User Experience Evaluation: Involve actual users in evaluating the usability of the complete application, specifically focusing on the ease of use when navigating through the shopping list, meal planner, and recipe sections.**
- **Chat Messaging Usability: Have users interact with each other on the web app by sending each other messages and evaluate the satisfaction of the users**
- **Call service Usability: Have users call each other and talk and see each other on the app, and evaluate whether they are satisfied by the experience**

- **Integration Validation: Please conduct a thorough evaluation of all essential integrations (including Call and Chat functionalities with UI, and external APIs) to verify their seamless operation and guarantee consistent data across the application.**

4. Pass/Fail Criteria

Functional Requirements

➢ **<u>Sidebar Functionalities</u>**

- **Test whether user can search for all users, and can get no results if there are no users => Passes when user can search for users and get a response of an array, Fails if app cannot access Algolia database and search users or results of search are false**
- **Test whether navigation works without any problems => Passes if user can jump between pages, Fails if user cannot access a page**
- **Test if list of recent chats of user can displayed filtered by the time of last message, new chats should appear at the top => Passes if user can get his recent chats, Fails if user doesn't get his recent chats or get chats with false date**
- **Test if chats display the correct number of unread messages => Passes if user can see the correct number of unread messages, Fails if user sees no number or a false number**
- **Test if chats display the correct format of last message => Passes if last message has the correct format, Fails if last message has a false format**

➢ **<u>Chats Functionalities</u>**

- **Test if user can send messages => Passes if user can send messages, Fails if user cannot send messages**
- **Test if user can see the typing and recording status of the other user => Passes if user can see the typing and recording status of the other user, Fails if user cannot see the status or the status is stuck on typing or recording**
- **Test if user can record audio and send it => Passes if user can record audio and send it, Fails if user cannot send audio**
- **Test if audio slider's color changes when the receiver of the audio plays it for the first time => Passes if color changes, Fails if color doesn't change**
- **Test if audio transcript is working => Passes if we get text of the audio, Fails if we don't get any text for the audio**
- **Test if user can send images => Passes if user can send images, Fails if user cannot send images**
- **Test if user can display images on full screen => Passes if user can display messages on full screen, Fails if user cannot display image on full screen**
- **Test if user can send multiple images with a message for each one => Passes if images are sent with their text, Fails if images are not sent or sent without their text or sent with the wrong text**
- **Test if user can click on the scroll button and the page will go down => Passes if user can scroll down, Fails if user cannot scroll down**
- **Test if user can see number of unread messages on the scroll button => Passes if user can see number of unread messages, Fails if there is no number**

- **Test if user can see “seen” at the bottom of the chat room when the other user sees last message => Passes if there is a seen, Fails if there isn’t**

➢ **<u>Call Functionalities</u>**

- **Test if users can call other users => Passes if user can establish a call, Fails if user cannot start a call.**
- **Test if users can receive incoming calls => Passes if user can receive calls, Fails if user doesn’t receive calls.**
- **Test if users can reject incoming calls => Passes if user can reject call, Fails if user cannot reject a call.**
- **Test if users can accept calls => Passes if user can accept a call, Fails if user cannot accept a call.**
- **Test if user can drag the call element without lag around the page => Passes if element is smoothly dragged over the page, Fails if element cannot be dragged or lags while being dragged or doesn’t follow the mouse or touch of screen correctly.**
- **Test if the user can switch on and off his camera and audio => Passes is user can switch off and on camera and audio, Fails if user cannot switch off and on camera and audio.**
- **Test if the user can go full screen on the call => Passes if user can go full screen, Fails if user cannot go full screen**

➢ **<u>Backend Functionalities</u>**

- **Test if user receives notifications when another user sends him a message => Passes if user receives notification, Fails if user doesn't receive notification**
- **Test if the notification come with right format => Passes if notification come with the right format, Fails if notification doesn't come with the right format**
- **Test if audio messages are converted to a text by our backend service => Passes if audio is converted to text, Fails if audio isn't converted to text**
- **Test if online status of users is updated => Passes if online status is updated, Fails if online status isn't updated**
- **Test if Calls are handled by deleting them when one of the 2 users in a call loses connection => Passes when call is deleted, Fails when call doesn't get deleted**
- **Test if algolia database is updated for search ability => Passes if we're able to search users, Fails if we cannot search users**

Performance Benchmark

Our performance benchmark passes when all these conditions are met and it fails when one of the conditions is not met

- **<u>UI Performance</u>**

- **Assess the intuitiveness and clarity of the application's interfaces through user testing and user experience evaluation.**
- **Test the responsiveness of our web app on multiple screens**
- **Test the istallability of our web app on local devices.**

- **Test the smoothness of animations in our web app.**

➢ **<u>App Performance</u>**

- **Test if data of users chats and messages are being fetched at small time**
- **Test if writing to our database like sending messages, and updating user info happens at a small time**
- **Test if our assets are downloaded at a short time when accessing our web app.**
- **Test if our backend responds quickly to our changes**
- **Test the performance of our call functionalities, quality of video and audio should be high**

Security Features

Our Security feature passes when all these conditions are met and it fails when one of the conditions is not met

- **It is imperative to guarantee that API communication is secure by utilizing HTTPS and appropriate authentication techniques.**
- **Additionally, it is crucial to validate that the application seeks and obtains explicit user consent prior to retrieving any data from the device.**
- **Lastly, ensure the secure storage of API credentials by confirming that they are stored in a secure manner and are not hardcoded into the application's source code.**

Integration Requirement

Our integration passes when all these conditions are met and it fails when one of the conditions is not met

- **Verify the seamless and uninterrupted integration of the call service within the application's user interface, guaranteeing its proper functionality within the app's context.**
- **Validate that all external APIs deliver precise and timely responses as per the application's requirements.**
- **Validate the dependability of automated build and deployment procedures.**
- **Conduct integration testing between the call, users, chats, and messages to ensure consistent data across different modules.**

5. Test Environment & Data

- **Hardware: Any device with a chromium browser**
- **Software: Chrome version minimum 90 is required**
- **Data: Firebase and Algolia and Daily are activated so we can reading and writing our data**
- **Tools: a NodeJS environment will be used to test everything, it will come with many testing tools**

6. Schedule

Month	Project Phase

March	**Basic functionality building with basic UI**
April	**Building the messaging features with notification**
May	**Building the call functionality**
June	**Rebuild the UI and AI chatbots to the app**

7. Risks and Mitigation

- **Risk-1: Inadequate test coverage of the application can result in critical parts being left untested, leading to undetected bugs until the market release.**
- **Mitigation-1: Create and maintain a thorough test plan that covers all functionalities, along with utilizing automated testing tools to enhance coverage.**

- **Risk-2: Integration failures may occur when components that work correctly on their own fail to function as intended when integrated with other parts of the application.**
- **Mitigation-2: Adopt continuous integration practices to identify and resolve integration issues at an early stage.**

- **Risk-3: Security vulnerabilities and flaws, particularly in APIs integration and data storage, can allow unauthorized access.**

- **Mitigation-3: Ensure that all external APIs have robust authentication measures and are accessed through secure connections and protocols.**

- **Risk-4: User acceptance issues may arise if the final product does not meet user expectations or preferences, impacting user satisfaction.**
- **Mitigation-4: Engage stakeholders and beta testers in the acceptance testing phase to validate the product against user needs and requirements.**

8. Demo of the web application:

Our web app is currently hosted at this URL [ChatPlus (chatplus-a3d50.firebaseapp.com)](chatplus-a3d50.firebaseapp.com).

When visiting the app do not forget to allow notifications, you can search for user "alaa eddine" and talk to me, you'll find two accounts, it's better to chat with me on the one with my profile picture.