LAZY TECH

March 23, 2019

Dr. Craig Scratchley
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia, V5A 1S6

Re: ENSC 405W/440W Design Specification for a Minute Taker Device

Dear Dr. Scratchley,

In the attached document, you will find the *Design Specifications for an Autonomous Minute Taking Device* **MeetAssist** as our Capstone Engineering Science Project. The Lazy Tech team is constructing an AI-powered minute taking a device that will capture meeting minutes, resolve the problem of identifying the speaker and provide a web application interface that creates a wholesome user experience with analytics and reports.

The purpose of this document is to elaborate and explain the design requirements of the general system, hardware and software that merge to create a functional product. All design requirements will be explored in different phases of our product development namely: the proof of concept, prototype and final product.

Lazy Tech consists of five final-year engineering students each with industry experience and specializing in various concentration from Electronic Engineering, Computer Engineering, to System Engineering: Vanshaj Kochar, Sarthak Sood, Arghavan Nassiri, Yagnik Vadher and Rafiul Islam. This document will be heavily relied upon during the design considerations and various development phases of our product. Should you have any questions regarding our requirement specifications, please contact our Chief Communication Officer at anassiri@sfu.ca.

Sincerely,

[illegible]

Yagnik Vadher
Chief Executive Officer
Lazy Tech

Enclosure: *Design Specifications for an Autonomous Minute Taking Device*

Design Specification

MeetAssist

Meetings Evolved

Project Members
Yagnik Vadher
Vanshaj Kochar
Sarthak Sood
Asal Nassiri
Rafiul Islam

Contact person
Asal Nassiri
anassiri@sfu.ca

Submitted To
Craig Scratchley (ENSC 405)
Dr. Andrew Rawicz (ENSC 440)
School of Engineering Science
Simon Fraser University

Issue Date
March 23th, 2019

Revision
1.0

Abstract

With modern corporations putting more emphasis on office tools that boost productivity, there’s been a new emphasis on holding meaningful and fruitful meetings. Enterprise, small to large, are now looking to eliminate reasonings behind their unproductive meetings. Common reasons such as having multiple speakers jumping from discussion to discussion without a coherent a sense of direction are killing the effectiveness of meetings.

Various solutions exist in the market however no single product provides the attendees at the meetings a centralized web application interface that provides a real-time transcription of the meetings along with a post-meeting summary and analytics on team engagement. This lack of a solution is a significant gap in the market.

Lazy Tech in aims of resolving the market gap, is prototyping an integrated hardware-software solution. Microphone clips are attached to meeting attendees to transcribe the meeting in real-time and record the audio. The clips interact with a central hub that connects to a cloud server to store audio recordings. A final web application provides users with an interface to review meeting notes, summary and analytics.

The design specification document outlines the design needs for hardware and software components that interact with one another to provide a comprehensive user experience. The document will elaborate on each component’s design details, namely: the clip and the hub, and the web application. A high-level test plan in Appendix A will summarize the testing on the design of hardware and software components to ensure quality assurance of features. Furthermore, another appendix is dedicated to the user interface (UI) and the appearance of software and hardware components with the intention of visualizing the design specifications.

Table of Contents

List of Figures

List of Table

1 Introduction

The central theme of today's technological advancement is undoubtedly artificial intelligence. Lots of application of Artificial intelligence and Internet of things have been developed to solve complex problems. As a result, Lazy tech introduced "*MeetAssist*", a product that aims to revolutionize the way meetings are conducted by making meetings more productive, efficient and collaborative.

Meetings are vital for an organization and taking meeting minutes is an essential component to it. It is considered extremely valuable in terms of communication providing a mechanism to craft strategic plans and discuss potential challenges and opportunities impacting their business. Taking meeting minutes is a key skill that requires the person to be an active listener and being attentive throughout to understand the goals and actions being discussed and be able to summarize the important findings. Currently, meetings do not have automation for taking meeting minutes or notes which requires a dedicated person to record meeting summaries and action items which is tiresome and not an efficient way since the person cannot entirely concentrate on the meeting with a tendency to lose focus. By using Lazy Tech's product *MeetAssist*, the user will be able to record a meeting, get an automatic summary of the meeting, and meeting analytics. This revolutionary product will create automated meeting minutes and analytics which will transform how the meetings are organized by making them more efficient and productive.In fig 1, a high-level behavioural overview is given.

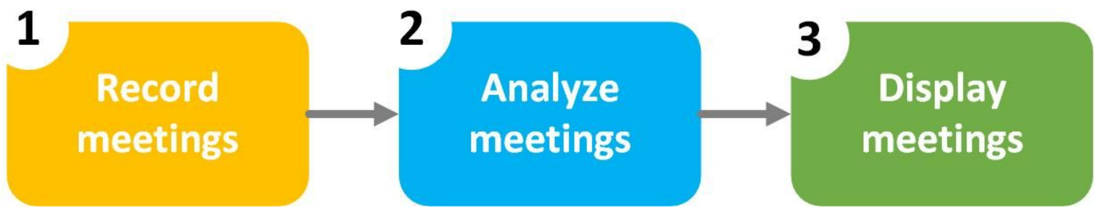


Fig 1: High-level behavioural Overview

MeetAssist follows a three-step process as shown in the figure above. First step, recording the audio of meeting from each individual with user recognition. Second step, to analyze the meetings audio, convert it to text and run summarizer to create meeting minutes. Third step, to display the results in the highly configurable, editable and most user comfortable way possible by the means of the web application (web app).

This document will discuss the in depth design specifications required to build this product and how each stage of the product is built separately and then integrated to achieve our final product. Prototype and final product designs are presented for both the hardware and software designs.

There is also an appendix section which includes the supporting test plans, CAD designs and summary of the appearance and requirements for the user interface.

1.1 Scope

This document supplements the requirement document providing detailed design solutions to the requirements mentioned. Apart from the design specification it also provides details on the products consistency with Engineering Standards and endeavours towards Sustainability/Safety. The document will list the design specifications for two Stages of the product i.e. the prototype stage and final product stage. It will include Appendix A, B and C that will list the test plans, CAD designs and user interface guidelines for the product.

1.2 Intended Audience

This document is intended to be a guideline during the development stages of Product *MeetAssist* for Lazy Tech members, Dr. Craig Scratchley, Dr. Andrew Rawicz and teaching assistants. The hardware and software engineers can use this document as a reference at any time during the development and/or testing stages of the product. The hardware designers can refer to the firmware specifications for understanding the modules and services required to build a communication network. Quality assurance engineers can use the test plan appendix to test system functions and follow safety/sustainability concerns so that the product meets industry standards and the product is user friendly.

2 System Overview

MeetAssist will consist of three primary components: Hub, Clips and web portal (connected via cloud). All three components will be connected together and communicate over wireless technology standard such as Bluetooth and Wifi. The figure 2.1 shows the high level design of the product,

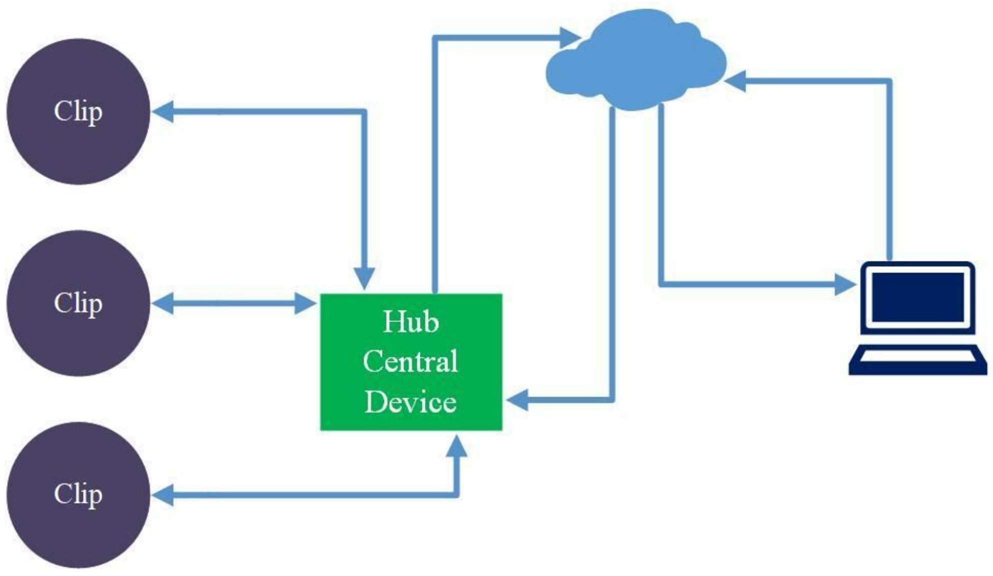


Fig 1.2: MeetAssist Product overview

The Clip is a device that is responsible to record the high-quality audio and transmit the recorded data to the Hub. In Prototype stage the clip will contain the Nordic Thingy 52 which is a prototype board containing the required microprocessor and microphone to capture and transmit voice data to the Hub. In the Product stage we will design custom made PCBs with a built in BLE and microphone to communicate with the Hub.

The Hub is a platform and device which is responsible to collect all the recordings from multiple clips and send it to cloud in real time. This will contain a Raspberry Pi 3+ with Broadcom chip BCM2837 to process the incoming voice data and transmit it to the cloud.

The third component of the *MeetAssist* is a web portal. It serves as a web application where the consumer will be able to see the past meetings or a project manager could edit, make changes and access features of *MeetAssist*. All the communication is bidirectional and therefore it will utilize the minimum throughput and provide real-time data transfer. It will consist of Azure cognitive services, RESTful APIs and backend server will comprise of Node.js. The web app will be made with AngularJS along with HTML and CSS.

3 Software Design

MeetAssist product relies heavily on software therefore it acts as crucial part of the system. In this document we have categorized three software system for better developmental plan and ease of classification. It is as follows:

1. Cloud services software
2. *MeetAssist* web portal
3. Artificial Intelligence

Cloud services, which is communication bridge between client end solution and the hardware. It also acts as computation platform. Web Portal is web app delivering rich and user-friendly dashboard for managing meeting minutes and other productivity feature on tip of a finger as it is cross platform and accessible anywhere. Artificial intelligence is machine learning algorithms to find a politeness or emotion from the conversation and to do speech to text translation. We have described complete detail of the design specification and requirements for PoC, Prototype and Final product in the sub section which is as following.

3.1 Cloud Services Software

Computation is needed in almost every product these days whether it is giant robots or tiny devices. As for *MeetAssist,* we need computation as well. For that purpose, we have chosen to use cloud Infrastructure provided by Azure. It is virtual machine provided by azure services.

In this virtual machines, 3 types of services will be deployed:

1. Web server
2. Database
3. Artificial Intelligence model

Main goal of cloud services software is to deploy above software's, establish continuous integration, development operation and add public port forwarding. We do this by writing scripts that runs continuous checks.

Web server will run 24 hrs, and 7 days. In case of crash, web server will restart and send alert to product engineer. When traffic gets beyond the server capacity it will notify the product engineer. It will have integration with gitlab project for continuous integration. In result, it will scale on demand in terms of computation speed. **Req 3.3.2.1-PT, Req 3.3.2.3-PT** is accomplished by our defined cloud structure. Authentication will be done with 2 factor authentication technology. This leads to secure communication. This fulfills our final product requirement. As required by **Req 3.3.2.4-FP** and **Req 3.3.1.5-FP,** Intercommunication between databases and the web server is done through RESTful APIs. It is based on representational state transfer technology, an architectural style and approach to communications often used in web services development. It takes explicit advantage of HTTP methodologies defined by the RFC 2616 protocol. They use GET to retrieve a resource; PUT to change the state of or update a resource, which can be an object, file or block; POST to create that resources; and DELETE to remove it. **Req 3.1.1.7-FP** is accomplished by described RESTful apis design.

For prototype version of our product we would be using limited services on cloud. Web server will run and accept users request. Database will be configured with minimum of 5gb of data space. Server will open up public port for access. Azure cognitive service will return real time speech to text result. Requirements are fulfilled by the azure virtual machine configuration. Also, azure provides easy to use user dashboard for setting up file system size and resource allocation. Bash scripts will be run on VMs for setting up environment variables and running provision scripts. VMs will be using Linux Ubuntu LTS 16.04 OS and on top, it will run *MeetAssist* services.

3.2 *MeetAssist* Web Portal

All of our results will be displayed on web app on user end. *MeetAssist* web portal acts as crucial part as it determines the most of the user experience.*MeetAssist* web portal is designed in a way to give highest priority to user experience as stated in **Req 3.3.3**. As it is web application and we are following server-client architecture, it consists of 3 major software part namely, backend, frontend and database. We have discussed each of the part in detail below. Also, fig 3.2.1 shows the overall architecture of the web portal.

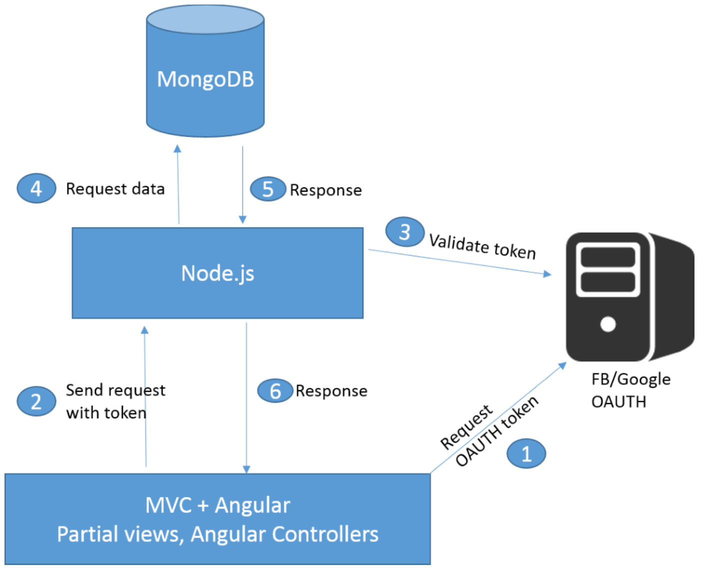


Fig 3.2: Web Portal overall architecture

3.2.1 Backend

For backend system, fast, lightweight and robust framework is required. Out of many Technologies, we have decided to use Node.js for our backend server.

Node.js is a good technology option on which to develop the interoperability layer core component for the following reasons:

- ➔ It is very lightweight
- ➔ It has a robust HTTP library
- ➔ It has robust support from 3rd party libraries for reading and modifying HTTP requests
- ➔ It is highly performant

Node server architecture that we will be implementing is described in the figure below.

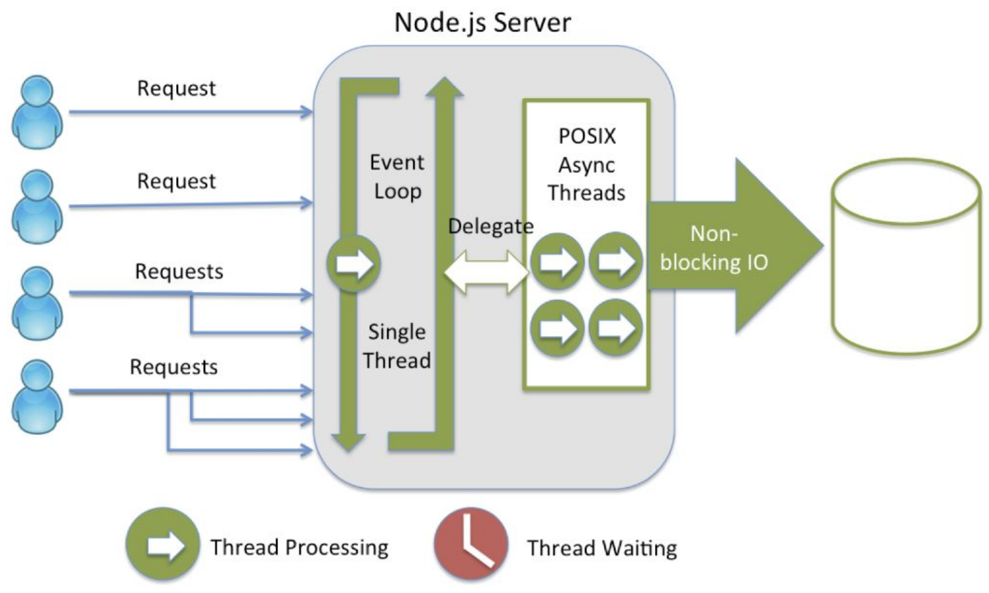


Fig 3.2.1: Node.js server architecture [6]

Backend is mainly responsible for providing various RESTful APIs to the frontend required by **Req 3.1.1.7-FP**. It also collects the data from database and runs regular checks to make sure database is updated.

It acts as secure communication medium for communicating data to user. Key design steps to take for backend is described below:

- ➔ It must open up REST api for fetching voice data.
- ➔ It must have secure authentication implemented using google OAuth.
- ➔ Backend shall have secure interface for the data retrieval from database.
- ➔ Backend must support multiple request at a time and should remain accurate which is configured on cloud services provided by azure.
- ➔ Backend must be able to serve web portal to user upon request. Azure VMs allows us to open up public port with Instant SSL certificate.
- ➔ In the event of a system failure, the server may be restarted, which must have a boot up time of less than 5 seconds. Azure VMs meet 5 seconds boot up time requirement.

Above steps are taken with the help of latest libraries and tools given by Node.js community. We will be using express js for web serving and other client-side request management. We will be using mongoose node module to communicate with database.

3.2.2 Frontend

Frontend is application that is in direct touch with client. Client will be using web browser to access the web portal (Frontend) vie HTTP request. This determines much of the user experience. Therefore, having a proper framework that can provide speed, performance and reliability is must as stated in Requirements document.

To add great user experience, the app is expected to load in less than 5 seconds. Also, the average response time shall not exceed 0.1 seconds. If response is taking longer than 1 second, the app will display a “waiting” symbol. As per **Req 3.1.2.4-FP,** App will be available on Google chrome and Mozilla Firefox web browser for best user experience.

In order to incorporate stated requirements **“Req 3.3.3.1-PT”** to “**Req 3.3.3.8-FP** ” We need a system design that would have modularity, cross platform capability and high performing. AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. AngularJS's data binding and dependency injection eliminate much of the code we would otherwise have to write.

We have listed design consideration for the frontend below:

- ➔ When the app first launches, the app will prompt the user to create a new account. If the user has an existing account, then they may directly login. The signup process shall be very simple and requires only the user’s name, email and a password. The user’s name field will be optional.

➔ When launching the app for the very first time, internet access shall be required for account creation and login. All consequent launches may not require internet as data will be stored in browser cache. The cloud shall store the login information and shall sync the user’s settings, configurations and users' meetings.

➔ As a precaution against hackers, there will be a maximum number of logins attempts for each email. Users can also recover their password by clicking on “Forgot password?” which will send an email with their old password.

➔ On the main screen user will be able to navigate to old meetings section where they must see the past meeting records including meeting minutes, audio and attendee's information.

➔ Layout of main screen will follow the material design concepts in order to keep user interface consistent and intuitive.

➔ User will have an option to create a new meeting. This will be accomplished by create new meeting module.

➔ User will be able to see real time transcription of the meeting on the web portal. This will be accomplished by real-time module in angularJS.

➔ User will have option to edit the meetings, save it and send it out to the attendees. This will be accomplished by edit meeting module with office 365 integration.

➔ There shall be an option for statistics which takes to detailed meeting analysis. This will be accomplished by new meeting module.

➔ AngularJS follows the MVC pattern and that makes a development organized. All of above design specifications will be achieved with the help of angularJS module following the MVC pattern. Figure below describes the MVC pattern.

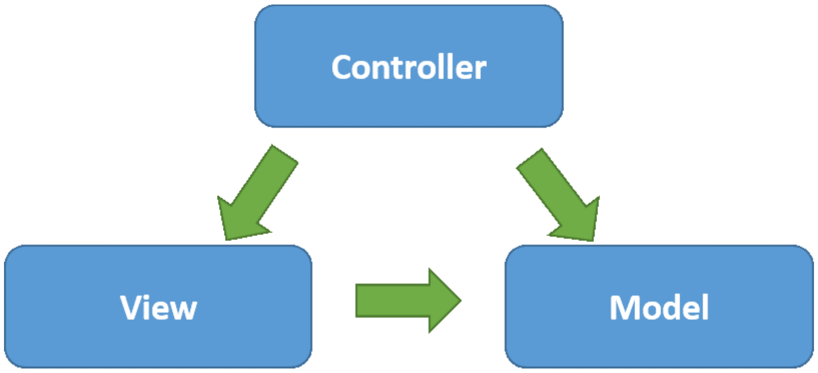
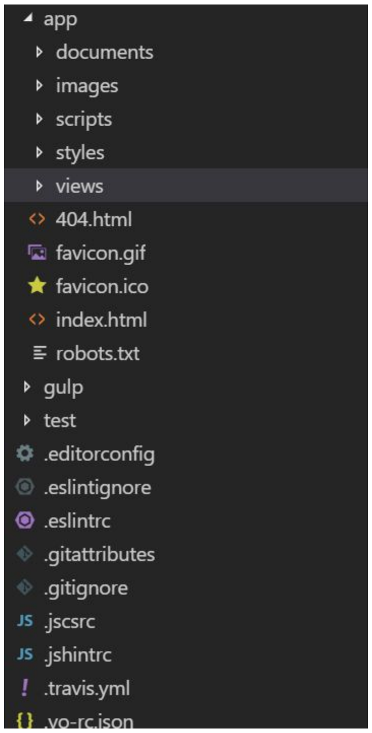


Fig 3.2.2.1: MVC framework [5]

MVC is made of three part namely model, view and controller. Model is responsible for managing application data. It responds to the request from view and to the instructions from controller to update itself. View takes care of presentation of data where UI parts get involved. The controller responds to user input and performs interactions on the data model objects. For our frontend design, we are planning to have followed high level design.



```
app
  documents
  images
  scripts
  styles
  views
  404.html
  favicon.gif
  favicon.ico
  index.html
  robots.txt
gulp
test
.editorconfig
.eslintignore
.eslintrc
.gitattributes
.gitignore
.jscsrc
.jshintrc
.travis.yml
.yo-rc.json
```

Fig 3.2.2.2 Code base architecture

As you can see from the code base architecture in fig 3.2.2.2, it is been modularized and it uses latest Technologies like ES6(ECMAScript standards 6) for better code management. Our main module will be application or root. It will connect with another modules. Config is redirected to routes and routes talks to view and controller. The architecture flow chart is given below in figure Fig 3.2.2.3. It depicts that each modules are connected well with well known software methodology [2].

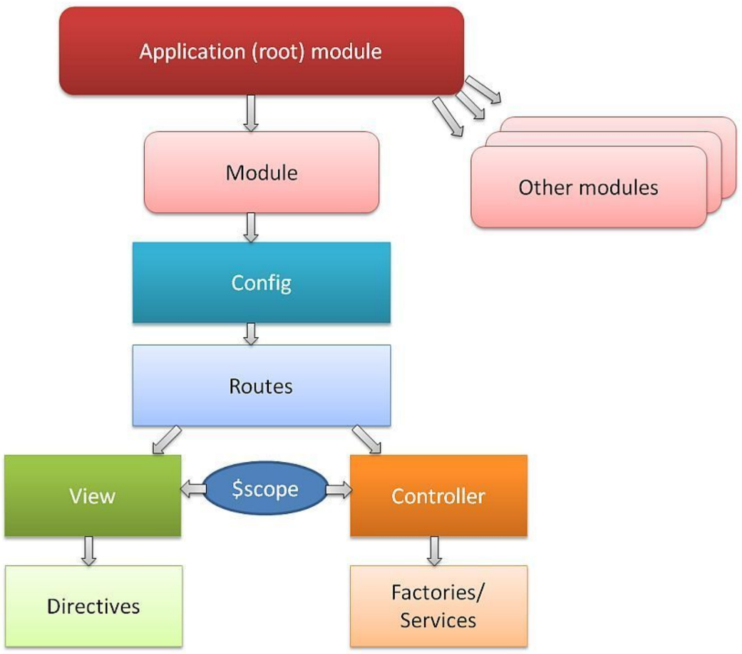


Fig 3.2.2.3: MeetAssit frontend modules architecture [7]

The system sequence diagrams presented below summarize the use cases for the webapp. Instructions for each diagram are also described to show how the major modules, classes, and functions relate to one another in the system.

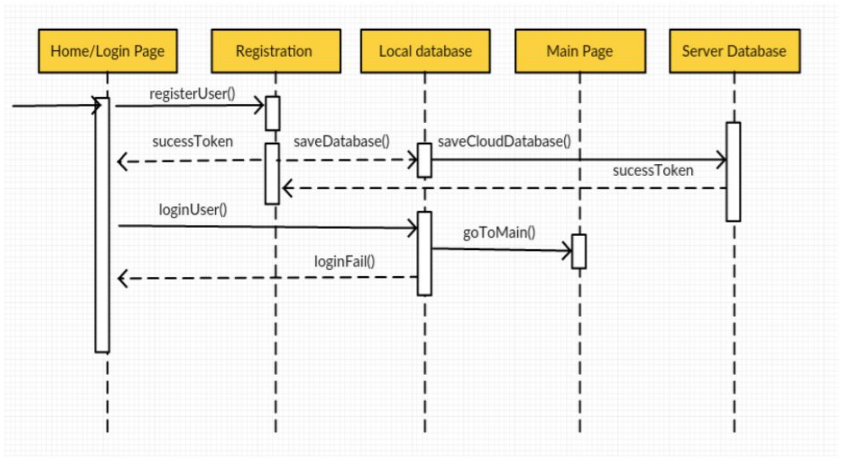


Figure: 3.2.2.4: Sequential UML Diagram for App Launch

The user launches the app for the first time (done by a project manager/attendees or admins). The user launches the application. The Home/Login Page is shown with the text input boxes for email and password. "Create a New Account" and "Forgot Password" buttons will be displayed below.

User taps on the “Create a New Account” button. User is taken to the Registration screen and can input their information. Once the Submit button is clicked, the user’s information is saved to the local database. A call to update the cloud database with the entered information will be made. User is redirected to the Login Page. User enters their new username and password, which is checked by the local database. A successful login will direct user to the Main Page of the app. An unsuccessful login attempt will prompt an error message and will display the Login Page again.

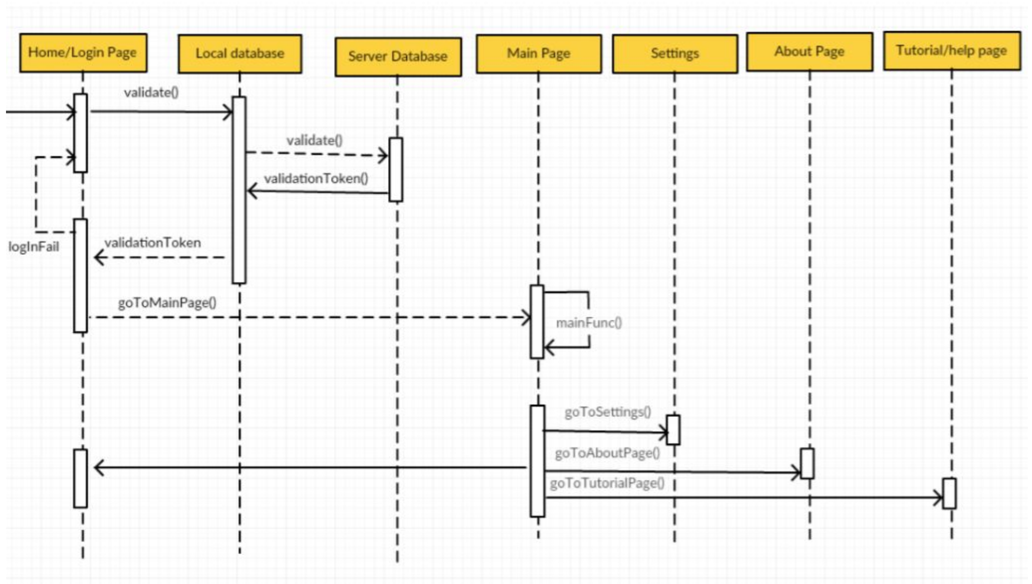


Figure: 3.2.2.5: Sequential UML diagram for Main page activity

Above diagram represent the main activity routing activity. Main function here involves user checking meeting minutes, updating it, sending out and etc. Diagram explains exactly how router will be navigated.

3.2.3 Database

We have planned to use NoSQL database for our application mainly because of ease of development and fast scalability. Our main data storage is text and voice based. For our voice recordings, we will be converting all voice data to base64 encoding format.

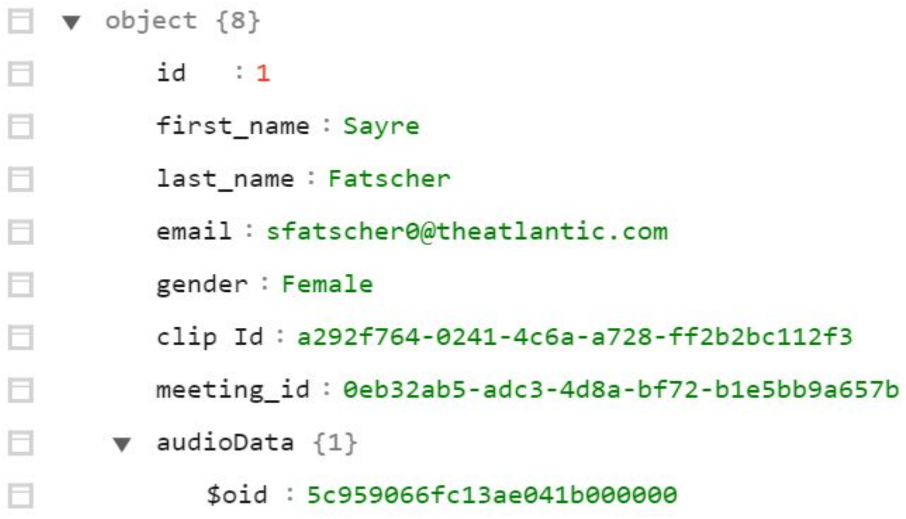
Further detailed design specifications are listed below:

- ➔ Database shall be able to store audio data in base64 encoding.
- ➔ Database should fetch any data less than 1 second.
- ➔ In case of database crash, it will preserve the data and disk.

- ➔ It will handle concurrent data request from server.
- ➔ Database will be capable of real time data manipulation.
- ➔ Database will authenticate before accepting any request for data retrieval or update.

For our prototype version, we will have database to store text-based data. It will update data in real time with maximum delay of 2 seconds. Database will use REST apis for CRUD (create, read, update and delete) operations.

As for prototype, we would not have any security measures for database but it will have basic functionality stated as per in requirements document.



```
▼ object {8}
      id   : 1
      first_name : Sayre
      last_name : Fatscher
      email : sfatscher0@theatlantic.com
      gender : Female
      clip Id : a292f764-0241-4c6a-a728-ff2b2bc112f3
      meeting_id : 0eb32ab5-adc3-4d8a-bf72-b1e5bb9a657b
   ▼ audioData {1}
         $oid : 5c959066fc13ae041b000000
```

Fig 3.2.3.1: NoSQL Database initial schema

Above data object in fig 3.3.1 shows the basic format of schema we will be using. This is up to the future development as data needs keep changing based on development cycle. But as of now, we are aiming to have similar data schema as shown above.

3.3 Artificial Intelligence

MeetAssist relies heavily on the artificial intelligence component. It is primary goal to convert audio recordings to text, therefore we need AI that support accurate transcription. There are various ways to tackle this problem. For our prototype model we will be using azure cognitive services for the speech to text[3]. For final product, we aim to develop or modify existing models of machine learning and create our own model with user tailored experience. As well as, we aim to have sentiment analysis on speech which could detect the health status of the meetings and flow of emotion throughout the meeting. We plan to user open source model by stanford research. We aim to accomplish this in final product. \

4 Hardware Design

4.1 Hub

4.1.1 Prototype

The Hub for our system will act as a bridge between the microphone interface (clip) and the cloud server. For this purpose we choose Raspberry Pi model 3+ as the central processing unit for the Hub in the Prototype stage. The Hub will communicate with the Cloud server using wifi sending data to a Demo Laptop using Azure Iot Services. The Physicals parameters of the Raspberry Pi have been taken into consideration. Raspberry Pi will be communicating with Nordic Thingy 52 using Noble.js a Bluetooth central low energy module running on the Raspbian OS Stretch.

4.1.2 Final Product

The Raspberry Pi used in the Proof of concept stage will be used in the Prototype stage along with additional refinements to the firmware. An outer hardware casing will be added to cover the Raspberry Pi. In this stage Wifi will be used to communicate with the cloud server along with supporting modules such as noble-device, BLE and services like Azure Cognitive Services.

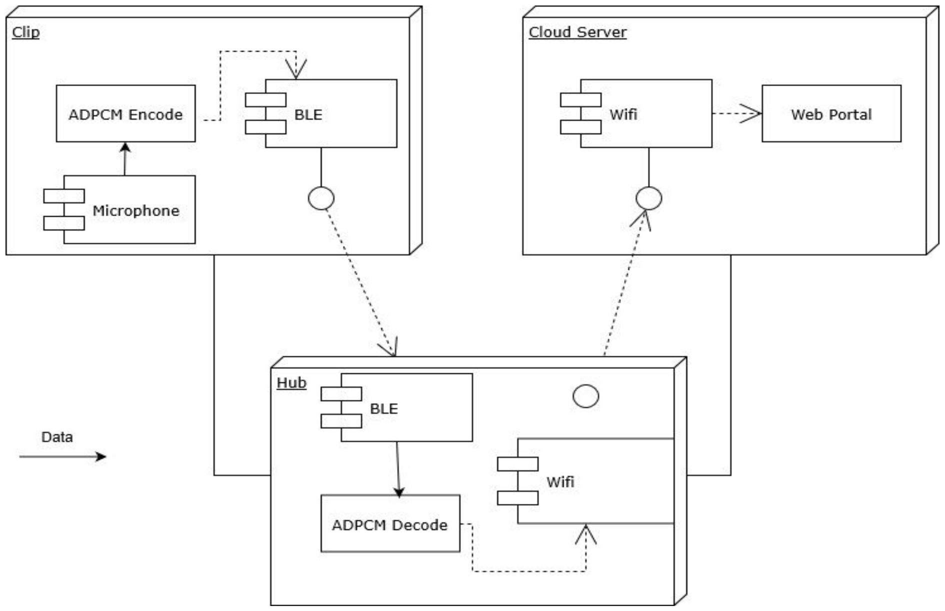


Fig 4.1.2: Block Diagram - Hub as a bridge between clip and Cloud server

4.1.3 Electrical Requirements

Following are the Electrical Requirements to provide the required power to the components of the hub, it includes Voltage and Current parameters, along with the resistance to external noise sources which could affect the electrical functioning of the microprocessor

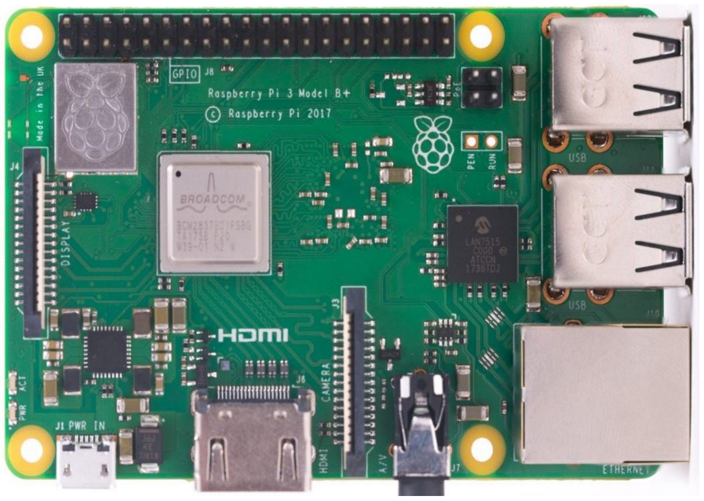


Fig 4.1.3: Raspberry Pi - Core Component of the Hub

The power supply to the Raspberry Pi 3 model consist of a 5.1V micro USB supply. The exact current consumption of the Raspberry Pi varies (700 - 1000mA) depending on the no. of GPIO peripherals connected but the official document confirms that a 2.5A power supply will be sufficient to power the Raspberry Pi. The above parameters will fulfill requirements **Req 3.4.2.1.1- PC** [1]

The Raspberry Pi has a thermal management system to ensure that excess heat can be handled and it consist of internal temperature sensor that communicates with the firmware to ensure that board will manage temperature uptil 85 ℃, thus meeting the requirement **Req 3.4.2.1.2- PC** . The newer version of Raspberry Pi which will be used in the Hub has improved heat dissipation and increased thermal mass. [2]

The Hub will have casing which will minimise external noise such as Electromagnetic interface, Radio Frequency interface, Cross Talk, Thereby fulfilling requirement **Req 3.4.2.1.3 - PC** [3]

The Final product designed will be storing data in real time with the cloud server, thereby if there is loss in electrical power the voice data transmitted to the Hub till the point of power loss will be stored securely on the cloud, thus meeting the requirement **Req 3.4.2.1.4 - FP**.

4.1.4 Microprocessor

The Microprocessor used in the Raspberry Pi is Broadcom chip BCM2837 which contains the ARM Cortex A53 cluster. This chip runs at 1.2Ghz resulting in 50% faster performance than the previous generation Pi chip. This will ensure that the Hub is functioning at a performance level that meets the requirement of data processing of the exchanged voice data and can handle the communication of processed data to the cloud server.

The primary requirement for the Hub was to consist of Bluetooth module which would be compatible with the bluetooth module on the clip. The Raspberry Pi 3+ has Bluetooth 4.2 and BLE which will be compatible with Clip Bluetooth module(**Req 3.4.2.2.1- PC**), also consisting of Gigabit Ethernet over USB 2.0 (maximum throughput 300Mbps) and 4 × USB 2.0 ports to satisfy the requirement for the board to communicate data at a acceptable rate(**Req 3.4.2.2.2-PC**). The Board consist of memory unit consisting of 1GB LPPDDR2 SDRAM, thus meeting the requirement **Req 3.4.2.2.5 - PC**.

The physical parameters of the Raspberry Pi has been shown in the image below, the physical parameters meet the HUB's outer casing which will allow our product to look compact and sleek (**Req 3.4.2.2.6 - PC**).

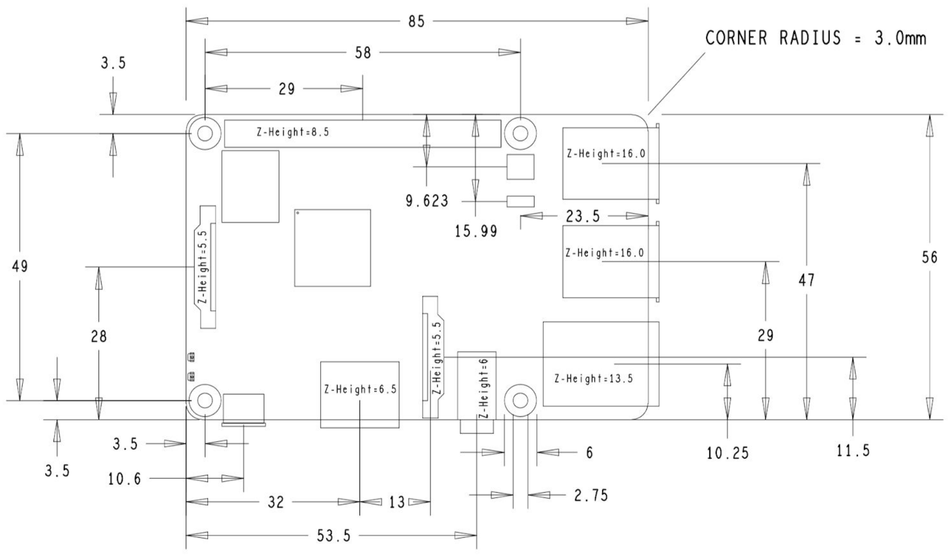


Figure 4.1.4: Physical Dimensions of Raspberry Pi [4]

The Raspberry Pi has Extended 40-pin GPIO header which will meet our requirement of indicating the state of the Hub(**Req 3.4.2.2.7- FP**). The board consist of 2.4GHz and 5Ghz IEEE

802.11.b/g/n/ac wireless LAN which will provide a secure medium to transfer data from the RPi to the cloud(**Req 3.4.2.2.8- FP**).

4.1.5 Firmware for the HUB

The hub will consist of the Raspberry Pi which will run on Raspbian OS Stretch (Ver 9) which will have all the capabilities of handling the incoming voice data and providing the required Voice Data encoding format for the Hub. The Pi is well supported by the Linux community and the developers follow a 2 year update cycle, thus meeting **Req 3.4.2.3.3 - PC** . This will ensure that the Hub component gets regular firmware updates.

The firmware powering the hub will be developed by the Team at Lazy Tech. Unethical practices or software will not be used in the design of the firmware. Industry software standards will be followed to ensure code quality and appropriate referencing will be used for freeware API used to in the firmware(**Req 3.4.2.4.2** - PC and **Req 3.4.2.4.3 - PC**). The Final Product will use Encryption and Decryption of text-to speech data to ensure the data transmitted over the internet is not susceptible to third party hack(**Req 3.4.2.4.5 - FP**).

4.1.5.1 Raspberry Pi Integration

The noble module will be installed on the Raspbian OS running on the Raspberry Pi, which will enable the Bluetooth module on the Hub to communicate with the Bluetooth module on the Clip by decoding the ADPCM encoded data. The Raspberry Pi will then convert this ADPCM decoded data to PCM format which is required by the Microsoft Azure IoT Hub to perform the speech to text conversion.[5]

For the integration of Hub and Clip three main components are required :

- ➔ Node.js JavaScript runtime.
- ➔ noble-device (A Node.js lib to abstract BLE (Bluetooth Low Energy) peripherals, using noble).
- ➔ Bluetooth 4.0 USB dongle supported by noble and node-bluetooth-hci-socket.

To connect the Raspberry Pi and the Cloud Server we will require:

- ➔ Azure IoT Hub/ Azure Cognitive Services
- ➔ Raspberry Pi 3 with power supply

➔ SSH and I2C connections on the Pi

The SSH connection to the cloud server over wifi will enable the Raspberry Pi to communicate with the cloud server. It will send the stored Decoded ADPCM Voice Data converted to PCM on the Raspberry Pi to Azure Cognitive service to convert the data into text. Following Diagram shows the overview of the firmware design on the Hub

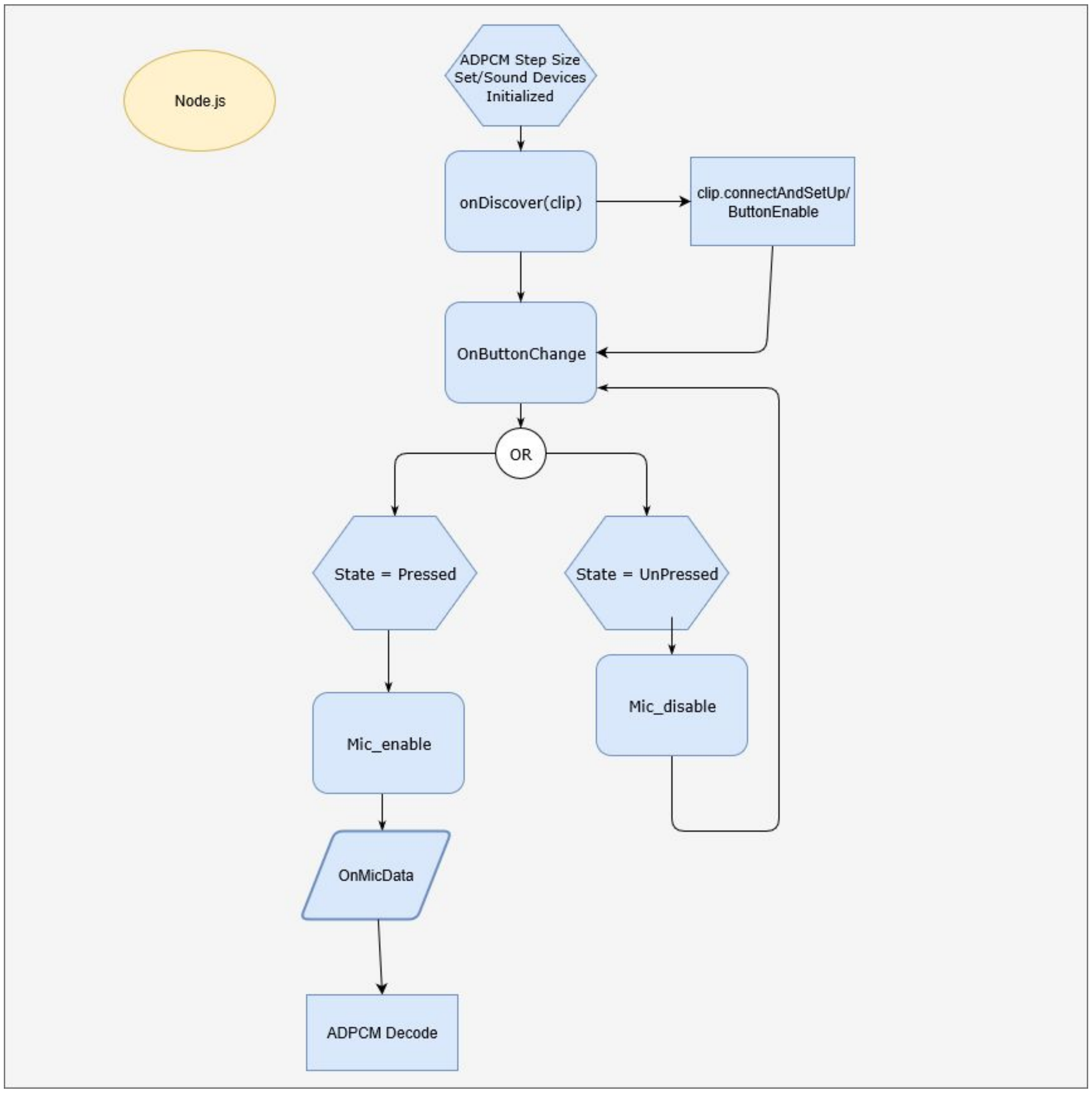


Fig 4.1.5.1: Firmware Flowchart for Prototype Hub and Clip Communication

The Firmware running on the Hub will initialize the variables and arrays including the ADPCM step size, Thingy id, Sound device, and button. On discovering a Bluetooth enabled Thingy, it will initialize and complete the connection between Raspberry Pi and Nordic Thingy and enable the button peripheral on the Thingy. Once the initial set up is complete, it will check the state of the button, if the button is not pressed it will remain in the stage where it is connected to the Thingy, if the button is pressed the Raspberry Pi will enable the mic on the thingy, once Mic Data is detected from the Thingy the Raspberry Pi will Decode the incoming ADPCM data and store it in a buffer to send to the cloud server.

For the Final Product we will have our custom made PCB communicating with the Hub Raspberry Pi, the button mechanism on the Product Clip will work differently from the one running on the Prototype, and will work more like a switch - The first time a the button is pressed the user will indicate that he wants to speak, the second time the user presses the button the user will indicate that he has finished speaking. Therefore the state of the button pressed will be stored on the HUB's firmware.

4.1.6 Hub Functional Requirements

Through the use of BLE the HUB will be able to connect to 8 concurrent Clip connections to enable voice data being received from multiple users in the same meeting(**Req 3.4.2.4.1 - PC**). The Raspberry Pi running on Raspbian OS with the installed modules will be connected to a Power Supply. The next step will be to connect to the Organisations Wifi once this is completed no other setup will be required to connect the system to the Clip and the cloud server.

For the Final Product The Hub Wifi setup will be completed by using the credentials provided by the organisation which will enable our to team to authenticate the connection between the Raspberry Pi and the Organisation's Wifi. The State of the Hub will be provided by LED lights connected to the Raspberry Pi. The LED lights on the RPi will light up colors indicating the state of the Hub.Colour coding for the Product is shown in the table below

State of the HUB	Color Code
Hub is on	Green
Hub has connected to Wifi	Flashing Yellow
Hub has encountered an error/Failed to connect to Wifi	Red

Table 4.1.6 : Color coding for the State of the Hub Product Stage

4.2 Clip

Clip is the integral component in our design phase for the product that comprises of a microphone and microcontroller that transfers data to the hub through the medium of BLE. To make our clip design efficient and compliant with the requirements specifications our team decided to choose Nordic Thingy: 52 which is a compact, power optimized, multi-sensor development kit that incorporates a built in microphone reducing work effort for our team to connect and configure an external microphone. It connects to Bluetooth low energy-enabled devices which would be Raspberry Pi 3+ (the hub) in our case and is capable of sending voice data through the built-in microphone to the hub via the medium of BLE [13]. The basic design for Nordic thingy: 52 labelling various components on both top and bottom PCB is shown below [14]

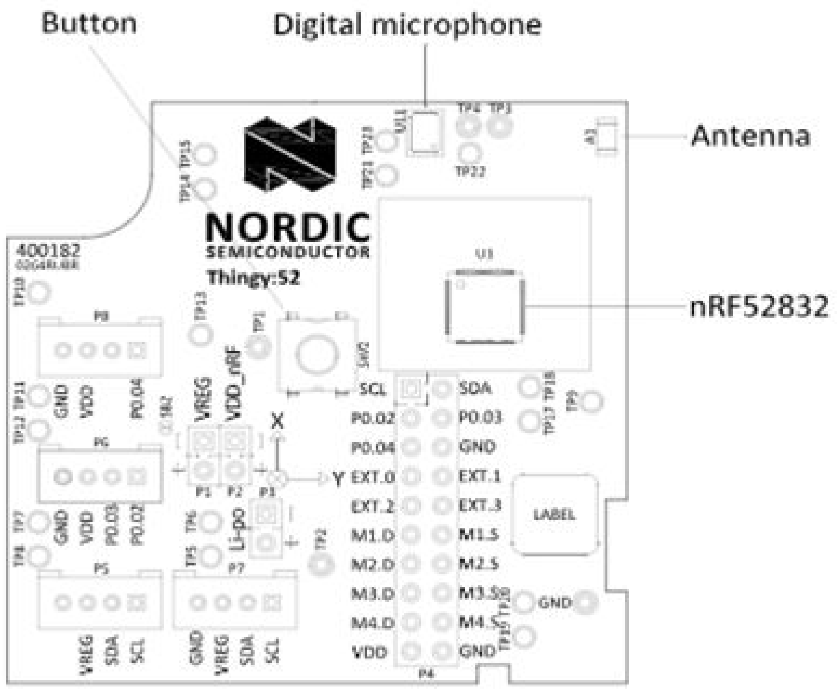


Figure 4.2.1 (top): Schematic diagram for top side of thingy PCB [14]

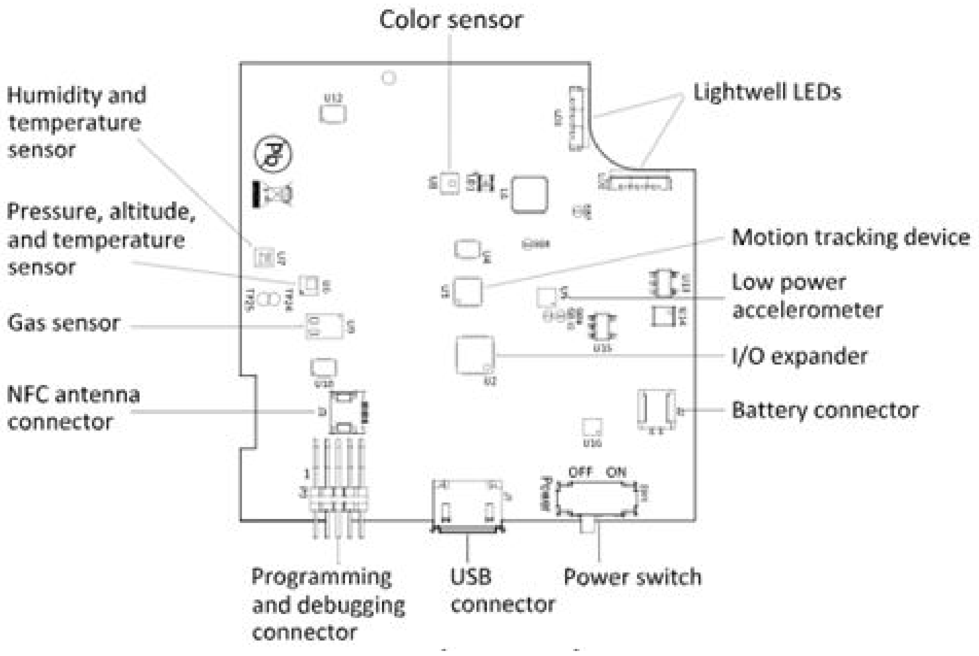


Figure 4.2.2 (Bottom): Schematic diagram for bottom side of thingy PCB [14]

4.2.1 Proof of concept

The primary task required by the clip is to record audio by at least one device and transmit data to the hub. We connected the Nordic device to the iPhone app and then recorded audio through the built-in microphone which was accessible to listen on the app which showed us that the Thingy 52 can be configured with the hub via BLE and will be able stream sound to the host (hub).

4.2.2 Prototype design

Nordic Thingy: 52 is a microcontroller which is great prototyping platform that provides us with great support in the development of our product and serves the purpose for audio communication between the clip and the hub. It is comprised of many components and includes various features that fulfills the need in our prototype design phase. The thingy requires power switch to be turned on to perform audio transmission. Our prototype will be focused on delivering the correct audio encoding to the hub from a single device that complies with our requirements specifications. The hardware block diagram below shows the interconnection between hardware components on Thingy and the key blocks that we want to incorporate in our prototype design are circled in red below [15].

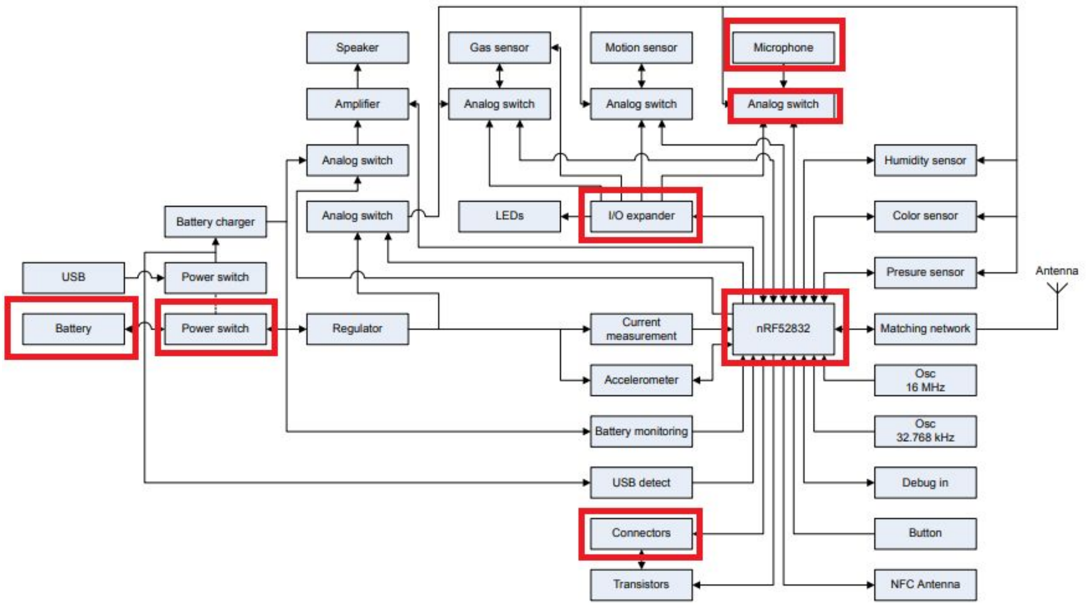


Figure 4.2.2.1: Thingy Hardware Block Diagram (Red symbolizes main block components) [15]

nRF52832 – The thingy is built around the nRF52832 Bluetooth 5 SoC. It is a powerful, flexible, ultra-low power SoC with a 32-bit ARM Cortex CPU that supports BLE which fulfills **Req 3.4.1.2.5-PT**. It provides a data rate between 1Mbps – 2Mbps over BLE, fast enough for data transmission as stated in **Req 3.4.1.2.3-PT**. Depending on the flash size it can store up to 32kB RAM for 256 kB flash and 64 kB RAM for 512 kB flash which is enough for data storage fulfilling **Req 3.4.1.2.4-PT**. While working on the prototype we realized that 8.3 mA transmission current stated in **Req 3.4.1.2.6-PT** can be reduced to 5.3 mA transmission current at 2.4 GHz providing low current consumption and thus, saving power [16].

I/O expander - The chip provides with a default 32 GPIO (General-purpose input/output) pins complying with **Req 3.4.1.2.8-FP** but the expander can be used to provide extended GPIOs. Most of the connection are internal. To connect the clip to the hub we will be using the connector block that has GPIOs for external hardware connection [15].

Digital Microphone – The microphone is the main component block that is needed to transfer audio to the hub which is done through an analog switch that will save power when microphone is not in use. It has a sensitivity of -96 dBm fulfilling the final product requirement in the prototype itself **Req 3.4.1.1.7-FP** [15]. The microphone in thingy streams the audio with the ADPCM codec but the hub requires the audio to be in PCM format to process audio as the cloud server communicating with hub supports PCM coding for speech to text conversion. Thus, we will use the ADPCM decoding algorithm to convert the audio to the required encoding format for audio to be processed correctly.

Power Supply – This chip offers power supply voltage in the range of 1.7V – 3.6V that satisfies both the microphone and microcontroller requirement **Req 3.4.1.1.1-PC** and **Req 3.4.1.2.1-PC** respectively [16]. Rechargeable lithium-ion battery of 1440 mAh is being used which acts as a power source and is connected to the power switch which must be turned on for the thingy to work and battery to charge. To save power an analog switch is used to turn off the power supply during sleep mode [15].

Button – The button will act upon the user input which needs to be pressed while recording audio in the digital microphone. Once the audio has been recorded the user can leave the button which will mark as the end of the data transmission.

4.2.3 Final product design

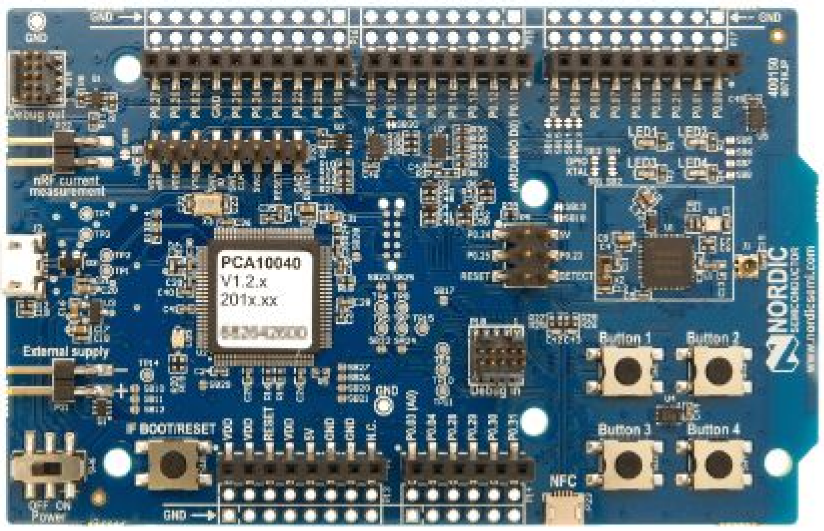


Figure 4.2.3.1: nRF52 Development Kit Board [17]

For our final product we will use the nRF52 development kit as shown in the figure above which will be used to program and develop the nRF52832 chip by writing our own firmware. To incorporate multiple users, we will be placing multiple nRF52832 chips (**Req 3.4.1.3.2-FP** and **Req 3.4.1.3.3-FP)** on our custom designed two-layer PCB board which will also include LEDs and buttons. A simple design for our custom PCB incorporating one button and one LED to connect one user is shown in the figure below.

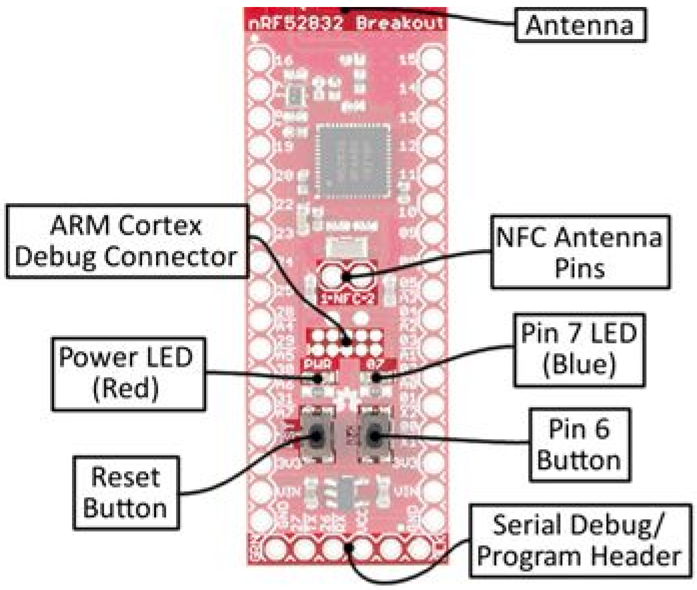


Figure 4.2.3.2: Custom PCB design with nRF52832 chip [19]

The reason for choosing two-layer PCB board is that it provides short return path for signals and also provides low resistance power supply to all sub-systems. Final design of the clip will incorporate all the functionalities of the prototype design with the following blocks modified as follows,

nRF52832 - This SoC functions as the brain of thingy which will be enhanced and modified by writing our own firmware and development code on the nRF52 development kit. It features 12-bit analog to digital converter (ADC) with 200 ksps (kilo samples per second) which should suffice our requirement **Req 3.4.1.2.7-FP** [16]**.**

I/O expander – Similar to the thingy 52 it supports GPIOs and has an integrated LED driver that supports intensity control, blink control and breathing control [3]. Connecter interface is used to access nRF52832 GPIOs which will be used to control the LEDs interaction [17].

LED's and Buttons – The clip will have a button associated with each individual user which when pressed will blink its corresponding LED on. The user will need to press the button to start recording audio into the mic which will send data simultaneously to the hub and once the user is done speaking, they can press the button again to stop the recording. A LED on signifies the audio transmission in progress. When the user presses the button, its state will change which will send a BLE notification making use of the BLE read and notify features.

To test our final product, we will connect a USB-C cable between the development board and the custom PCB board. In an efficient way jumper wire connection will be established while programming the chip with the development kit on the PCB as shown in the figure below. Our team will need to decide whether to solder the jumper wires for a secure connection and if it might cause any problem.

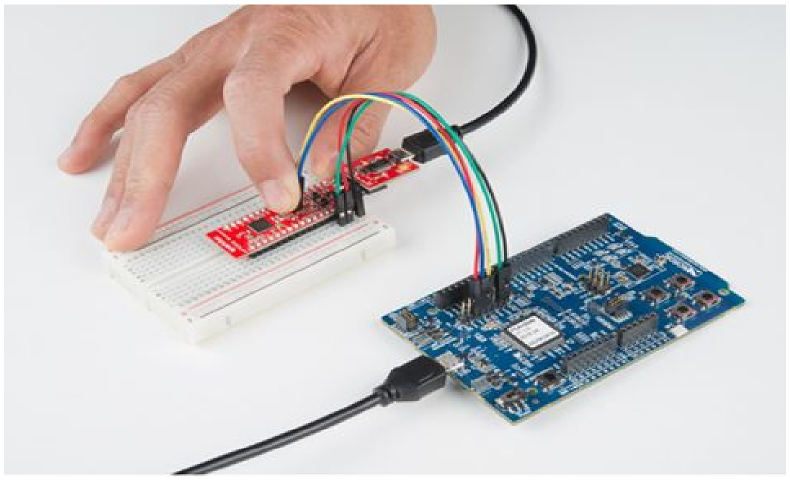


Figure 4.2.3.3: Programming chip with nRF5 kit on custom PCB [19]

4.2.4 Firmware Design

The thingy firmware is written with the help of the nRF5 software development kit (SDK) and is built on top of it utilizing most of its components that includes SDK peripheral drivers, SosftDevice and SDK HAL (Hardware Access Layers) as shown in the firmware architecture below [18]

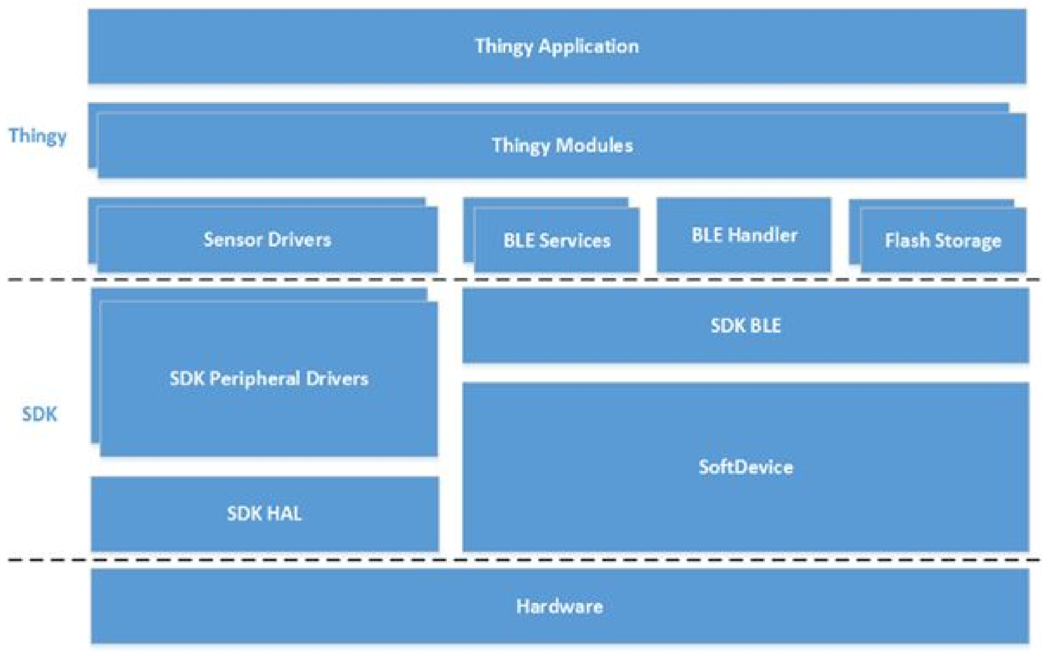


Figure 4.2.4: Firmware architecture framework [18]

The thingy contains the following modules on top of the nRF5 [18]:

BLE handler – It is responsible for handling effective Bluetooth communication and dispatch events to the processes.

Sensor drivers – These drivers are mostly used to perform operations such as enabling/disabling them, changing configuration or fetching data and are built on top of the hardware access layers.

Thingy modules – It controls the sensor drivers and BLE services and contains high level functionality and features. It also handles the storage of configuration and default configuration. The modules used in our device are:

I) BLE handling module
II) Sound module
III) User interface module
IV) Battery measurement module
V) Environment module

BLE services – Thingy modules use the BLE services for implementation and is discussed in detail below.

BLE services

Most of the BLE services are based on the simple GATT (Generic Attributes) service and includes the following services on the thingy firmware [18]

Configuration service – Configuration service handles all general configuration parameters that are not related to any particular module.

Environment service - The environment module is used for reading pressure, humidity, temperature, gas and light intensity sensors and sends data for configuration. It is built on the environment BLE service (ble_tes), Environment sensor drivers and environment flash storage.

User interface service – UI module handles the LEDs and buttons and runs on top of the UI BLE service (ble_uis), the LED and button drivers and UI flash storage. UI module sets the RGB LED value and button event notification via BLE.

Sound service – Sound module is responsible for handling both the speakers and the microphone and is built on top of the sound BLE service (ble_tss). Microphone characteristic can store maximum of 273 bytes in ADPCM mode.

Battery service – This service handles all the battery level information. It stores 1 byte of data and only notifies if there is a change in the battery level.

DFU service – This service is meant to provide device firmware updates on Thingy.

An example below shows the function used for setting the mic for data transmission using the sound BLE service [18]

ble_tss_mic_set()

```
        unit32_t ble_tss_mic_set (ble_tss_t* p_tss,
                                   unit8_t* p_data,
                                   unit16_t* size
                                   )
```

// This Function sends microphone input as microphone characteristic notification to the peer

Parameters:

[in] p_tss	Pointer to the Thingy sound service structure
[in] p_data	Pointer to the mic data
[in] size	Mic data size

Return Values:

NRF_Success	If the string is sent correctly otherwise an error code is returned

Firmware for Thingy is written by referencing The Thingy module API's and BLE services API.

5. Conclusion

MeetAssist provides a medium for storing verbal communication exchanged in a meeting by using speech to text AI in sync with speech differentiating microphone. This product provides a method for formatting and analyzing data which could be documented for the purpose of checking the productivity of a meeting. Through the use of a microphone component *MeetAssist* will differentiate between speakers in a meeting which could be vital in segregating meeting objectives once the meeting has been conducted. Our web app will offer the customer an elegant means to edit and manage their data.

MeetAssist has three major components therefore the Design constraints specified in this document have been decided after carefully considering the compatibility of each component to another.The Design Specifications were divided into two stages i.e, Prototype and the Final Product. The prototype has been designed to cover all the basic features of MeetAssist meeting the minimum viable product criteria, the Final Product will be built with enhanced functionality and interactive UI. Software being a core component of the Product was divided into interdependent components i.e., Cloud Services using Azure, Web Portal Database and Artificial Intelligence component comprising of Azure Cognitive Services, thereby covering all the critical features of the software component of the product.

Hardware Design was divided into subsections such as Electrical, Firmware , Functional and Microprocessor. The prototype stage uses Raspberry Pi Model 3+ for the hub component due to its versatility and meeting the Hardware, Electrical and Firmware requirements set by the Team. The Nordic Thingy 52 prototype board is selected for the clip due to meeting the core requirement of having a microprocessor which supports BLE module, microphone and being compact in size. Thus the Final Product will be built upon the prototype in which all the basic features will be implemented. A PCB with custom firmware based on the prototype will be designed for the Clip along with external casing, and Raspberry Pi with functionality to connect with multiple clips with external casing will be used for the Hub in the final stage.

6. Glossary

RESTful - Representable state transfer
AI - Artificial Intelligence
CRUD operations - Create, Read, Update, Delete Operations
LTS- Long time support
MVC - Model, View and controller
UI - User Interface
ES6 - Ecmascript 6
Ecmascript - European Computer Manufacturers Association Script
UML - Unified modeling language
JS - Javascript
SQL - standardized query language
SSH - Secure Shell
HTTP - Hypertext Transfer Protocol
RFC - Request for comments
DB - Database
POSIX - Portable Operating System Interface
BLE - Bluetooth Low Energy
Wifi - Wireless Fidelity
USB - Universal Serial Bus
GPIO - General Purpose Input Output
ARM - Acorn RISC Machine
LPPDDR2 - Low-Power Double Data Rate (Ver 2)
SDRAM -Synchronous Dynamic Random Access Memory
IEEE - Institute of Electrical and Electronics Engineers
RPi - Raspberry Pi
OS - Operating System
API - Application Program Interface
ADPCM - Adaptive differential pulse-code modulation
PCM - pulse-code modulation
IoT - Internet of Things
I2C - I-squared-C, is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial computer bus
PCB - Printed Circuit Board
LED - Light Emitting Diode
HAL - Hardware Access Layer
GATT - Generic Attributes
SDK - Software Development Kit

7. References

[1] H. Admiraal, “Technical design in UML for AngularJS applications.” 2015

[2] P. Chandrayan, “All About Node.Js You Wanted To Know ?,” *codeburst*, 29-Oct-2017. [Online]. Available: https://codeburst.io/all-about-node-js-you-wanted-to-know-25f3374e0be7. [Accessed: 25-Mar-2019]

[3] “Cognitive Services,” *Microsoft Azure*. [Online]. Available: https://azure.microsoft.com/en-ca/services/cognitive-services/. [Accessed: 28-Mar-2019].

[4] L. Fu and J. Dai, “A Speech Recognition Based on Quantum Neural Networks Trained by IPSO,” *2009 International Conference on Artificial Intelligence and Computational Intelligence*, 2009.

[5] *Angular*. [Online]. Available: https://angular.io/guide/styleguide. [Accessed: 28-Mar-2019].

[6] “All About Node.Js You Wanted To Know ?,” *codeburst*, 29-Oct-2017. [Online]. Available: https://codeburst.io/all-about-node-js-you-wanted-to-know-25f3374e0be7. [Accessed: 28-Mar-2019].

[7] “AngularJS,” *AngularJS Tutorial: Routing III - 2018*. [Online]. Available: https://www.bogotobogo.com/AngularJS/AngularJS_Routing_C.php. [Accessed: 28-Mar-2019].

[8]“Power Supply,” *Power Supply - Raspberry Pi Documentation*. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md. [Accessed: 24-Mar-2019].

[9]*Raspberry Pi Documentation*. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/frequency-management.md. [Accessed: 24-Mar-2019].

[10]G. Vijayaraghavan, M. Brown, and M. Barnes, “Electrical noise and mitigation - Part 1: Noise definition, categories and measurement,” *EETimes*, 16-Dec-2008. [Online]. Available: https://www.eetimes.com/document.asp?doc_id=1274125#. [Accessed: 24-Mar-2019].

[11]Static.raspberrypi.org. (2019). [online] Available at: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf [Accessed 24 Mar. 2019].

[12]GitHub. (2019). *NordicPlayground/Nordic-Thingy52-Nodejs*. [online] Available at: https://github.com/NordicPlayground/Nordic-Thingy52-Nodejs [Accessed 26 Mar. 2019].

[13]*Nordic Semiconductor Infocenter*. [Online]. Available: http://infocenter.nordicsemi.com/index.jsp?topic=/com.nordic.infocenter.rds/dita/rds/designs/thingy/intro/frontpage.html. [Accessed: 24-Mar-2019].

[14]*Nordic Semiconductor Infocenter*. [Online]. Available: https://infocenter.nordicsemi.com/index.jsp?topic=/com.nordic.infocenter.rds/dita/rds/designs/thingy/hw_description/hw_figures.html. [Accessed: 24-Mar-2019].

[15]N. Semiconduictor , "Nordic Thingy 52: User Guide v1.1." 27-Sep-2017

[16]*Nordic Semiconductor Infocenter*. [Online]. Available: http://infocenter.nordicsemi.com/index.jsp?topic=/com.nordic.infocenter.nrf52/dita/nrf52/chips/nrf52832_ps.html. [Accessed: 24-Mar-2019]

[17]N. Semiconduictor , "nRF52832 Development Kit: User Guide v1.2." 15-Feb-2017

[18]BLE services," *Nordic Thingy:52 v2.2.0 : Firmware architecture*. [Online]. Available: https://nordicsemiconductor.github.io/Nordic-Thingy52-FW/documentation/firmware_architecture.html. [Accessed: 25-Mar-2019]

[19]*nRF52832 Breakout Board Hookup Guide*. [Online]. Available: https://learn.sparkfun.com/tutorials/nrf52832-breakout-board-hookup-guide?_ga=2.73747354.980765483.1510236640-313485059.1503660044. [Accessed: 25-Mar-2019].

Appendix A - Test Plans

MeetAssist aims to provide a means to users to record and analyse meeting minutes, this process should be seamless and the user should be able to execute the functions of MeetAssist with minimal effort. To ensure the quality of the product this Appendix lists the Tests and expected results. This document should be used as a tool to check the product before the product is shipped to the customer. The test plans have been divided into sections and subdivided into components to ensure all major features of the Product are covered. These tests will cover the Prototype Stage and can be replicated for the Product Stage. Important note since the Product Stage is still in pre-development stage the workflow of the test and/or Expected result might change based on the implementation.

A.1 Electrical Tests

Component	Test	Expected Result	P/F
Hub	Plug in 5V DC / 2.5 A micro-usb power supply to the Rpi, using digital Multimeter check the micro-usb outlet for the current reading.	Multimeter should read 700 - 1000mA. No leakage current is expected in normal conditions (5 µA expected over 85℃)	
Hub	Plug in 5V DC / 2.5 A micro-usb power supply to the Rpi, check GPIO pin current reading	Individual GPIO pins will draw maximum of 16mA	
Clip	Turn on the switch for the Clip and check current reading (Idle state)	Multimeter should read a few hundred nA	
Clip	Turn on the switch for the Clip and connect to the HUB check current reading (Connected state)	Multimeter should read 10 to 30 mA	
Clip	Drain the Battery to minimum and recharge the battery	The battery should recharge and the device should function properly	

Table A.1 - Electrical Tests for hardware components

Comments

A.2 Hardware Tests

Component	**Test**	**Expected Result**	**P/F**
Hub	Power the Hub and let it run for a long period of time (2 hours)*	Voltage supply Remains Stable	
Hub	Power on the Hub check LED light	The LED light on the Hub should indicate green light	
Clip	Toggle the on/off switch	The Clip should turn on when switch is on (Blue Blinking light) And off when switch is turned off	
Clip	Connect Clip to Hub, Press the Microphone Button	The Voice Data should be transmitted	
Clip	Turn the clip switch on and let it stay idle for a long time (2 hours).	The clip should actively listen for incoming BLE connection and remain in the on state	
Clip	Turn on the switch for the Clip check the LED light	The LED light on the Clip should indicate Blue static light	
Clip	Turn on the switch for the Clip and connect to the HUB check the LED light	The LED light on the Clip should turn static green	

Table A.2 - Hardware Tests consisting of hub and clip

Comments

*Note Time estimated for the test based on average maximum meeting time for a corporate meeting

A.3 Software Tests

Component	**Test**	**Expected Result**	**P/F**
Overall Software	Check for Code Coverage - Unit Tests	Code Coverage should not be less than 80%	
Web Portal	Access the URL for MeetAssist web app	The Home Page should show up with authentication section	
Web Portal	Connect the Hub and the clip and record meeting minutes	The Web portal should show real time recorded data	
Final Product Tests			
Web Portal	Create a new user account	The user should successfully be able to create a user account	
Web Portal	Login with a Test Account	Login successful - User Page should show with Past User meetings and a sidebar with different options	
Web Portal	Click on different Sections of the User Account page - Past Meetings	This should show the user for the recorded meetings and option to make notes	
Web Portal	Click on different Sections of the User Account page - Analytics	This should show the user the Sentiment Analysis done on the Recorded meeting	
Web Portal	Click on the Tutorial Section	Tutorial Page will load with tutorials on setting up and using MeetAssist	
Web Portal	Click on the Hamburger sign besides the sidebar	The sidebar should hide/show depending on the previous state	

Table A.3 - Software Tests for Prototype and Final Product

Comments

A.4 Integration Tests

Component	**Test**	**Expected Result**	**P/F**
Clip/Hub	Power on the Hub, Power on the clip	The clip should connect to the Hub indicating a static green light on the LED	
Clip/Hub	Hub and clip connected, move the clip away from the Hub upto 10 meters apart	The connection between the two components should remain strong	
Hub/Wifi	Power on the Hub and connect it to the organisations wifi network	The Hub should connect to the Wifi	
Clip/Hub/Web Portal	Transmit Voice Data	Voice Data is Actively Transmitted through the three components	
Clip/Hub/Web Portal	Transmit Voice Data	Text Data Transmitted matches the voice data spoken by the user	
Final Product Test			
Clip/Hub	Turn Hub on , Turn Multiple Clips on (upto 8)	Hub connects to each clip status on each clip shows green static light	

Table A.4 - Integration tests for Overall Product

Comments

Appendix B - CAD Designs

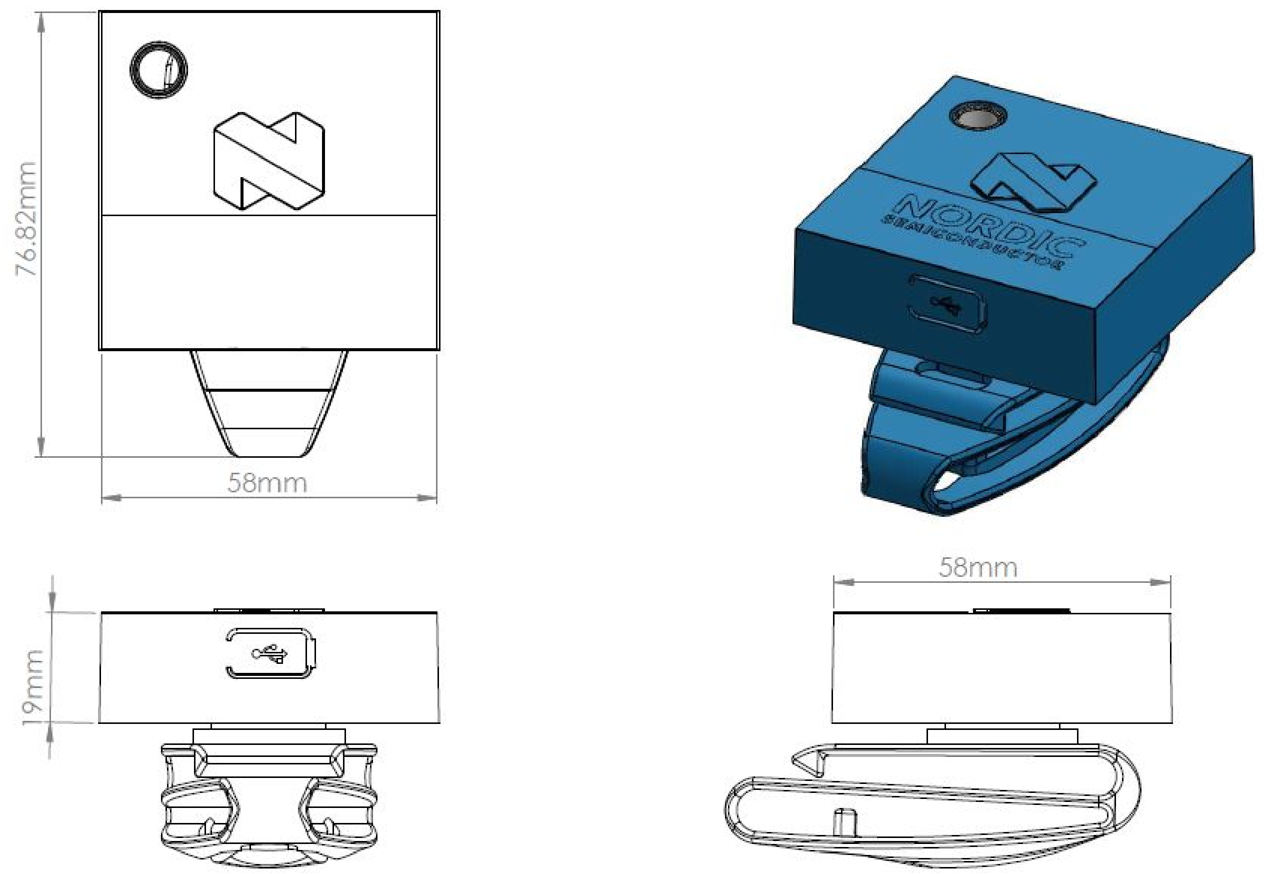


Figure B.1 - The prototype version of the clip

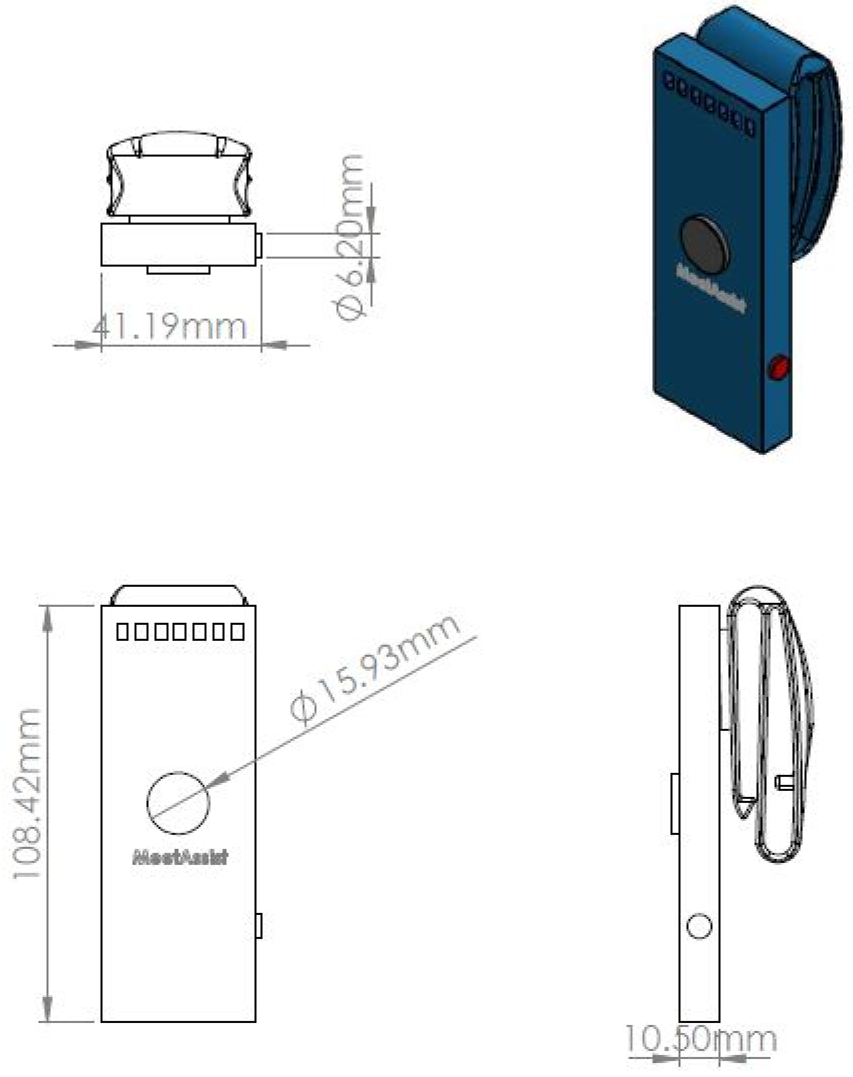


Figure B.2 - The final version of the clip

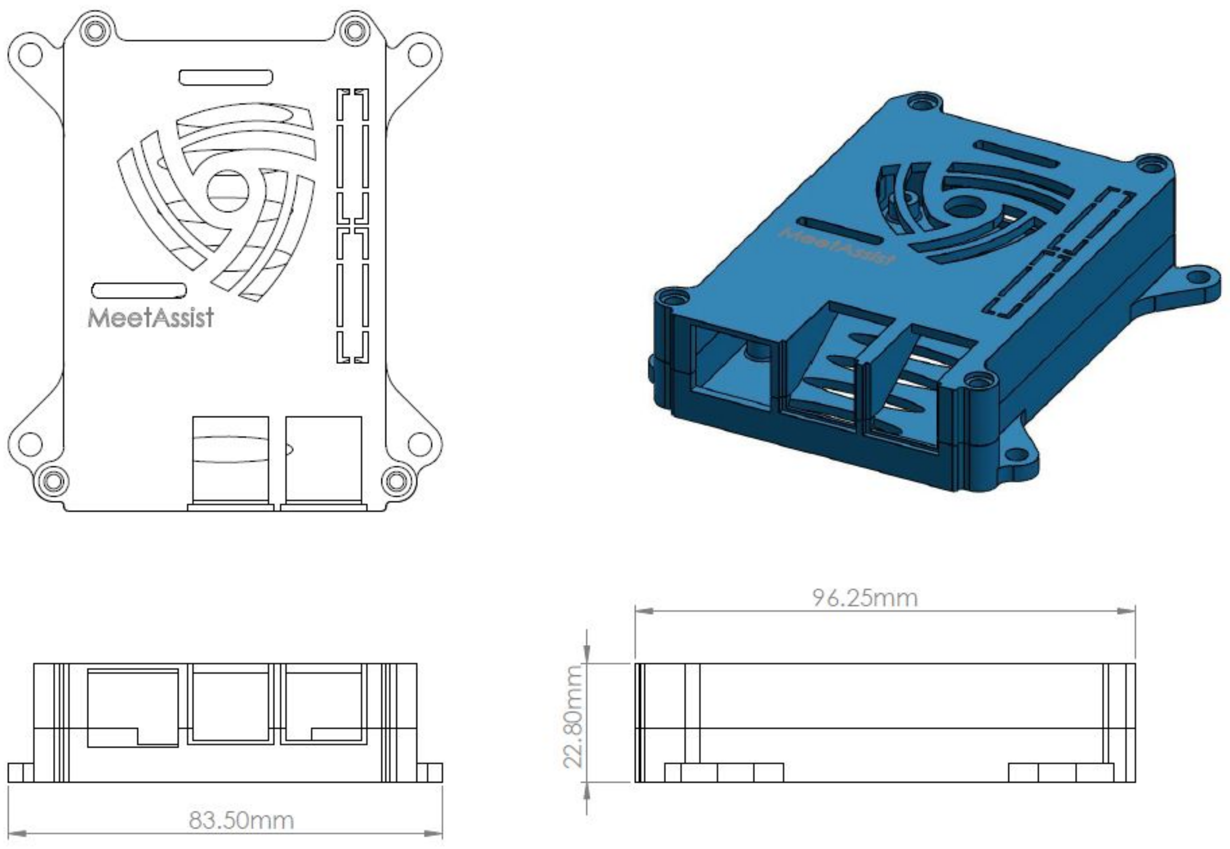


Figure B.3 - The casing of the hub

Appendix C – User Interface & Appearance Design

C.1 Introduction

The MeetAssist by Lazy Tech is a solution to the problem of lack of productivity in team meetings. This solution is comprised of two hardware components and a unifying software component, namely the clip, the hub and the web application. While the clip attaches to the meeting attendees' clothing, the hub rests upon a central desk around which the attendees are sitting. The design of the clip focuses on comfort and appearance, especially when it is to be attached to clothing. Similarly the design of the hub should emphasize comfort with soothing edges and a friendly appearance. The design for the web application will heavily rely upon user friendliness, functionality and ease of use.

C.1.1 Purpose

This Appendix illustrates the software and hardware design of the MeetAssist and further expands on the reasonings behind design decisions on hardware characteristics and software features.

C.1.2 Scope

The Appendix will elaborate on multiple topics:

1. User Analysis
2. Technical Analysis
3. Graphical Presentation
4. Engineering Standards
5. Analytical Usability Testing
6. Empirical Usability Testing

C.2 User Analysis

The intended users of the product package are those who attend in-office meetings. The clip's design is considered in a way to be comfortable to clip onto shirts or lapels of suits for both men and women. The design of our product does not consider remote attendees but such a consideration will be made in the future iterations of our product as the team understands the importance of the increase and growth in modern way of working remotely. The clip's design will be as compact as possible with respect to dimensions of the board as the clip attaches to clothing. It is important to acknowledge the fashion statement any attachable device would make when in essence it becomes a wearable technology and such a statement will be considered in the slick design of the clip.

The hub is a box-shaped with a modern twist to it where the covering is similar to that of a smart home speaker. The reason for such a design is to consider future applications of adding AI powered assistant to the meeting product or have the option of integrating currently existing AI-powered assistants to the product. The hub does not require constant user interaction which makes the matter of design much simpler.

The web application is geared towards usage by two types of users. First and primary usage will be by all meeting attendees who:

- ➔ are curious or require a refresher on the results of the meeting
- ➔ want detailed transcription of the meeting
- ➔ want a summary of the discussion in the meeting

The secondary users of the web applications are the managers or meeting owners who are curious about where their meeting productivity is going. They can access analytic results and team health reports and statistics that identify ways to get more out of the meetings.

C.3 Technical Analysis

Technical analysis section is broken down to and is agreeable with Don Norman's "Seven elements of UI Interaction" as listed below:

1. Discoverability
2. Feedback
3. Conceptual models
4. Affordances
5. Signifiers
6. Mappings
7. Constraints

C.3.1 Discoverability

Discoverability, sometimes referred to as learnability, determines the level of ease at which the user demonstrates their capability to identify/locate the UI components in their first interaction with the product [1]. At Lazy Tech, the team believes that if there is less interaction with the device, the higher the level of discoverability in the sense that there will be a gentle or almost no learning curve. As a result, the hub has almost no interaction other than being plugged in and hooked up to an internet connection. The initial connection setup will be programmed on the hub when the user puts an order to purchase the device where they will be prompted to provide their WiFi information.

The clip on the other hand has two buttons: a button that turns the clip on or off and another that signals the hub that the user is about to speak. More complex than the clip, the web application has multiple dimensions that the user would need to consider in order to understand the full functionality of the product. With such a scenario, the Lazy Tech team has diligently chosen an open source framework called Material UI that presents web application modules similar to that of Google products. Such a decision by the team is to increase the likelihood of easy learnability as Google products are among the most popular free tools available online. Furthermore, there will be a tutorial page provided to the users upon their first interactions to ensure immediate understanding of the graphical interface. Regardless, the team will emphasize simplicity as the main theme of design, almost eliminating the need for a tutorial page.

C.3.2 Feedback

A cohesive feedback system that alerts the users about functionality of the product is an essential measure of the product's user friendliness. Feedback should give the user the signals that are necessary to understand the state of the product, whether everything is functional or there has been an issue or error in the system that requires user's attention. There's a LED light that indicates whether the clip is powered on, not connected to the hub or potentially non-functional. Similarly, another LED light on the hub indicates whether the hub is on, off or connected to the WiFi.

C.3.3 Conceptual models

Similar experiences in a system's design provide the user a sense of familiarity with the product that allow the user to set their expectations and define their behaviour with the product. In MeetAssist, the hub is designed similar to that of a smart mini-speaker while the clip is similar to that of a paper clip. In addition, the web app uses Material UI framework that visualizes modules, buttons and graphs similar to that of Google's products, the most widely used set of online free tools. Such considerations in conceptual model of Lazy Tech's product gives users a familiar experience thus reducing steep learning curves.

C.3.4 Affordances

Affordances are the product of characteristics, the feels and the looks of a product that determine how users interact with it. The hub has a modern feel and look to it with minimal interactivity while the clip is obvious and self-explanatory as it can be clipped onto clothing.

The web application, being the most informative and interactive part of the product, will make use of the simplest design, titles and buttons and provide the users with an interface that puts emphasis on each feature:

- ➔ Past Meetings
- ➔ Analytics
- ➔ About Tutorials

C.3.5 Signifiers

When affordances fall short of fulfilling their purpose, items that could be part of the feedback system of the product need to step in and redirect the user to continue operating the device. These items are the signifiers. Each hardware component has an LED light as a signifier. The hub's LED displays static red when it is not connected to internet or when there has been an error in processing data. Flashing yellow light on the hub indicates the device has successfully connected and is operational. Appearance of any light is indicator that the device is powered on. No light indicates no power or no activity by the hub.

The clip's signifiers will be developed in two phases of prototype and final product. In the prototype version of the clip, a static blue LED indicates that the device is powered on. When it is connecting to the hub via bluetooth, it will have a blinking blue light and once successfully connected, the LED light will turn static green. In case of any errors, a red LED light will provide user the necessary feedback. In the final product version of the clip, a static and constant green light indicates that the device is turned on. In the process of connecting to hub, the clip's LED light blinks blue until it turns into a static blue light when connected. Blinking red light indicates potential connectivity issues or device malfunction.

C.3.6 Mappings

Another significant element of UI interactions is mapping which indicates how the placement of an element such as a LED light relates to the status of the device. Design considerations such as specific placement of the LED light on the hub and the clip or the text or the character that determines the functionality of the button or a signifier are examples of applying mapping. MeetAssists LED lights and button indicators will follow standard mapping schemes, assisting the user to know the status of the hub and the clip. Mapping conventions will also be heavily relied on in Web App's UI, allowing the user to interact with the tool intuitively and on instinct, rather than trial and error.

In the prototype version of the clip, a switch will turn the device on and off with a 0 and 1 as indicators of on and off. In the final product version, rather than a switch, a power button will be placed on the side of the device with a power icon imprinted on the button. No characters or text will be supplied in correlation to LED lights on either the hub or the clip in order to simplify the mappings and avoid overcrowding the device with instructive measures.

C.3.7 Constraints

To avoid unintended behaviours by the device, certain constraints must be put into place that limit the user's interactivity with the device. Largest constraints have been put on the hub where there has been the least amount of interactivity with the device as there are no buttons and only a LED light. The clip also has a single button on it, limiting any confusion or possibility of unintended behaviour by the user. Standard constraints are being put into place for the UI design of the web app, where the user is not confused about how to interact with familiar elements such as menu items, hamburger button, buttons, opening and closing modules and etc.

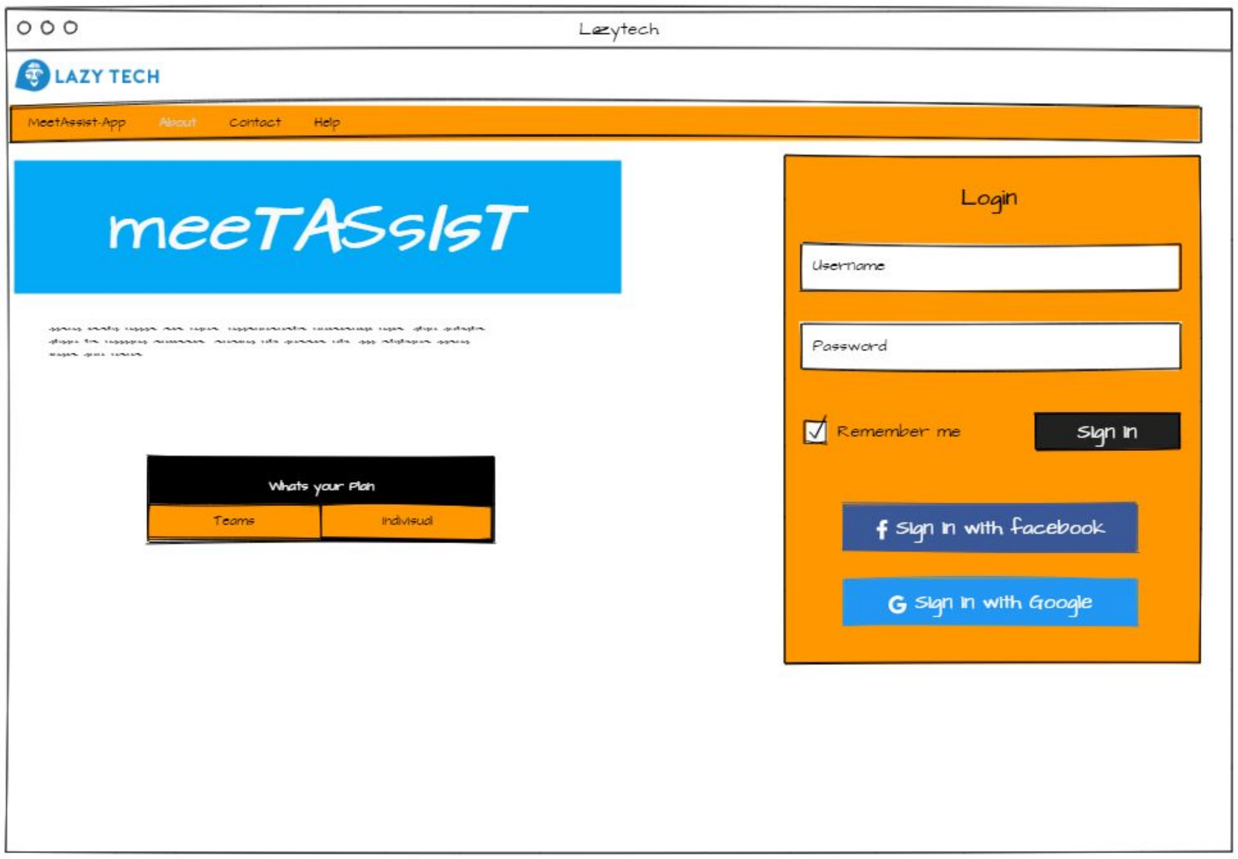
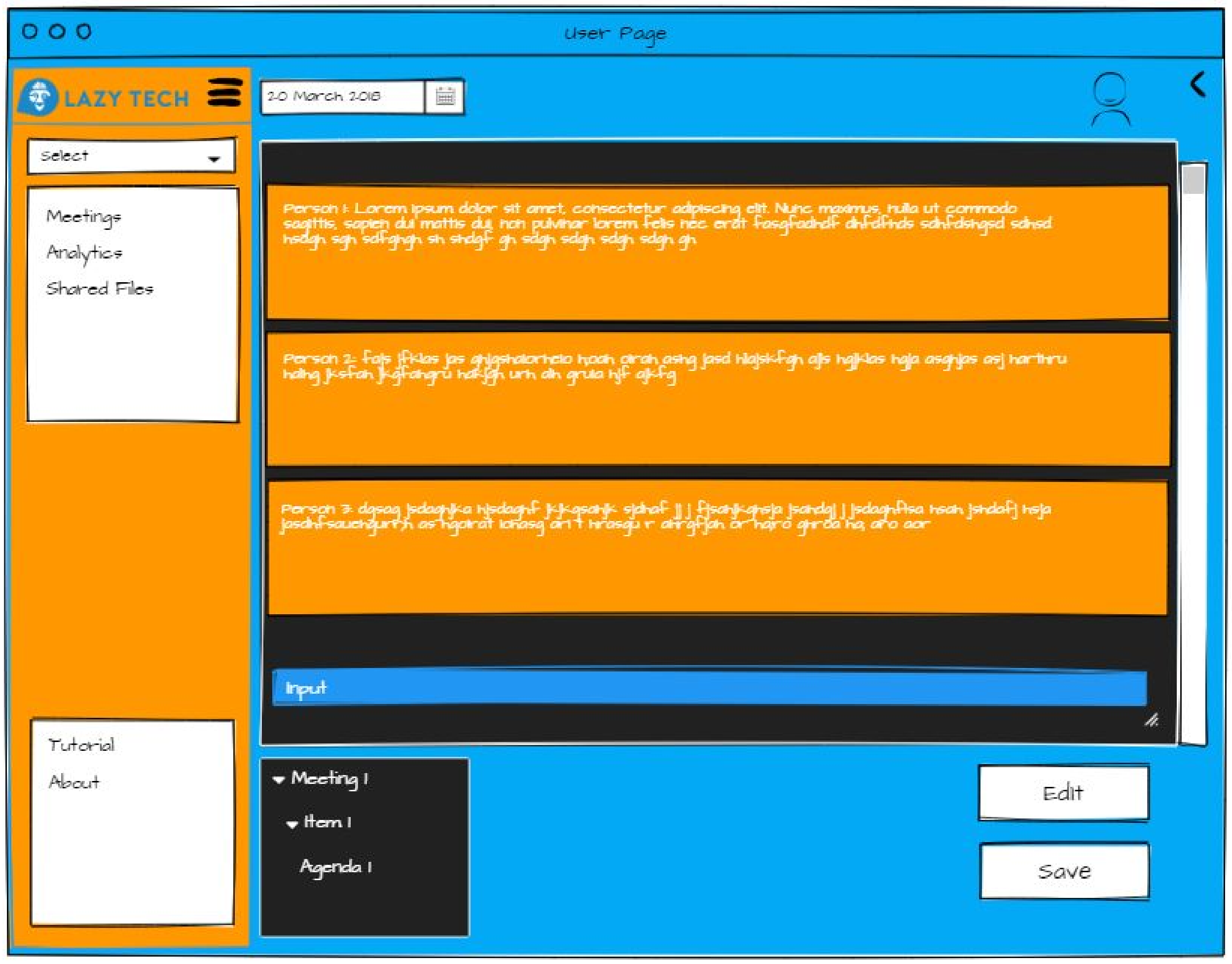


Figure C.3.7.1 - Website wireframe design of the home page



Person 1: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat fasgfadhdf dhfdfhds sdhfdshgsd sdhsd hsdgh sgh sdfghgh sh shdgf gh sdgh sdgh sdgh sdgh gh

Person 2: fals jfklas jas ghjgshdlorhelo hgah argh asng jasd hlajskfgh ajs hajklas haja asghjas asj harthru halhg jksfah jkgfahgru hakfgh urh ah gruia hjf ajkfg

Person 3: dgsag jsdaghjka hjsdaghf jkjkgsahjk sjdhaf jjj fjsahjkghsja jsahdajj jsdaghfisa hsah jshdafj hsja jasdhfsauehgurth as hgarat lahasg ar t hrasgu r ahrgfjah ar haro ghrda ha, aro aar

Figure C.3.7.2 - Website wireframe design of the dashboard

Figures in Appendix B provide an illustration of the full 3D model along with dimensions on three sides of each hardware component in prototype and final development phases: the hub and the clip.

C.4 Engineering Standards

In development of any product, standards set the expectation of the quality, performance and safety of devices. A table has been constructed that lists the necessary standards that the Lazy Tech team will adhere to in order to develop the hub, clip and the web application in accordance to standardization organizations such as IEEE, IEC and ISO.

Standard Code	**Description**
IEEE 1621-2004	IEEE Standard for User Interface Elements in

	Power Control of Electronic Devices Employed in Office/Consumer Environments [2]
IEEE 1012-2012	IEEE Standard for System and Software Verification and Validation [3]
IEEE P360	Standard for Wearable Consumer Electronic Devices - Overview and Architecture [4]
ISO/IEC 21823-1:2019	Internet of things (IoT) -- Interoperability for internet of things systems [5]
ISO 24624:2016	Language resource management -- Transcription of spoken language [6]

Table C.4.1:Standards of ISO, IEC and IEEE

C.5 Analytical Usability Testing

In this stage of the testing, the design and user interface of the hardware and software components will be evaluated heuristicly by the design team. Each member of the design team will examine the hub, the clip and the user interface of the web application and make notes of issues with compliance, usability and learnability. Let us not forget that the usability testing for the web application will occur after the proof of concept development phase of the product.

Below are the test requirements that will be evaluated:

Proof of Concept:

Hub

1. The hub turns on with LED light indicator after being plugged into a power source.
2. The hub attempts to connect to the WiFi with a blinking LED light indicator.
3. The hub automatically connects to the WiFi, while presenting a static LED light.

Clip

1. The clip should display no light when the device is not powered on.
2. The clip powers on after power button is clicked with a LED light indicator.
3. The clip LED light clearly indicates successful connection to the hub.

Prototype:

Hub

1. The hub indicates errors with a red LED light.

Clip

1. Once the button on the clip is pressed, the LED light indicates the user is allowed to speak.
2. Once the button on the clip is pressed, the LED light indicates it is not time for the user to speak yet

Web Application

1. The web app provides the user with an authentication method that requires username/password
2. The web app includes a comprehensive and easy-to-understand dashboard
3. The web app includes the list of meetings that have already occurred.
4. The web app has a transcription of the meetings that have already occurred.
5. The web app has analytics data on team health status.

C.6 Empirical Usability Testing

Internal Testing

The team will stage a meeting where multiple speakers participate by attaching the clip to their outfits and attempt hold conversations regarding the product itself. The team will take this opportunity to measure the success rate of the hardware functionality of the hub and the clip as well as the software. The performance of the software will be evaluated by how well the voice-to-text engine will work, how well the web app displays the list of meetings and how data is gathered to provide analytics to the end user.

End User Testing

A select number of individuals from the group member's social circles will be participating in a user acceptance testing phase. In this phase, users will attempt to interact with the product without prior knowledge by following the instructions below:

- ➔ Power on the clip
- ➔ Attach the clip to your shirt or collar
- ➔ Engage in a conversation with a specific topic determined in advance of the meeting
- ➔ If you would like to participate in the discussion, press the button on the clip
- ➔ Once you are done speaking, press the button on the clip again
- ➔ Refer to real-time transcription of your words on a screen
- ➔ At the end of the meeting, navigate to the web application to find more information about your meeting

Following the activity, the users will be provided a survey that should provide the team with necessary data to make adjustments to the product.

The survey questions will be similar to the list below:

1. How easy was it to attach the clip to your clothing?
2. Was working with the clip in association with the hub convenient to your conversations? Please provide a few sentences.
3. How comfortable were you in participating in the conversation while interacting with the clip?
4. Was having real-time transcription of your meeting useful to you? If not, please provide us with a few reasons.
5. How would you use the web application's analytics to improve your next meeting?
6. What other information would you like to know about your meeting?

C.7 Conclusion

A major cause of commercial failure of electronic products is design. Poor design often leads to user frustration and inability to operate the device and ultimately to a financial disaster for a technology company. That is why the Lazy Tech team has taken the necessary steps to outline the seven UI design principles and elaborate on how they will create a user-friendly device with almost zero learning curve. All components of the product, the hub, the clip and the web app have been analyzed, looked over and planned to be tested so that the end-user would have a comfortable experience in attending and participating in an office meeting. The team aims to cover majority of the hardware usability goals in the proof of concept phase of development while developing an advanced web app that meets modern UI requirements in the prototype phase of development. The objective of the team overall for hardware and software is simplicity along with full functionality. It is inevitable though that the team will continue to strive to improve the design of the product as technology advances over time.

C.8 References

[1] M. Rouse, “discoverability (in ux design),” 2018. [Online]. Available: http://whatis.techtarget.com/definition/discoverability-in-UX-design

[2] Standards.ieee.org. (2019). *IEEE 1621-2004 - IEEE Standard for User Interface Elements in Power Control of Electronic Devices Employed in Office/Consumer Environments*. [online] Available at: https://standards.ieee.org/standard/1621-2004.html [Accessed 28 Mar. 2019]

[3] Standards.ieee.org. (2019). *IEEE 1012-2012 - IEEE Standard for System and Software Verification and Validation*. [online] Available at: https://standards.ieee.org/standard/1012-2012.html [Accessed 28 Mar. 2019].

[4] Standards.ieee.org. (2019). *P360 - Standard for Wearable Consumer Electronic Devices - Overview and Architecture*. [online] Available at: https://standards.ieee.org/project/360.html [Accessed 28 Mar. 2019].

[5] 21823-1:2019, I. (2019). *ISO/IEC 21823-1:2019*. [online] ISO. Available at: https://www.iso.org/standard/71885.html [Accessed 28 Mar. 2019].

[6] 24624:2016, I. (2019). *ISO 24624:2016*. [online] ISO. Available at: https://www.iso.org/standard/37338.html [Accessed 28 Mar. 2019].

Appendix D : Requirement Specification

D.1 Functional general requirements

Requirement ID	General Functional Description
Req 3.1.1.1-PT	Clip must record meeting minutes audio with sensitivity up to 105 db
Req 3.1.1.2-PT	Clip must transmit recordings to the Hub via Bluetooth low energy
Req 3.1.1.3-PT	Hub must connect with clip and receive data via Bluetooth low energy
Req 3.1.1.4-FP	Hub must transmit data over the internet to cloud for analytics
Req 3.1.1.5-PC	Software must be able to do speech to text transcription
Req 3.1.1.6-PT	Cloud services must translate audio recording to text
Req 3.1.1.7-FP	Cloud services must open up REST apis to public on authentication
Req 3.1.1.8-PT	Web portal must display meetings transcription
Req 3.1.1.9-FP	Web portal must have a user interface (UI) for editing meeting notes

Table D.1 A list of General Functional Requirements

D.2 Non-functional General requirements

Requirement ID	General Non-functional Description
Req 3.1.2.1-FP	Clip shall be a wearable device that could be clipped on shirt or t-shirt
Req 3.1.2.1-F	Clip battery shall last more than 2 hours
Req 3.1.2.2-FP	Clip shall be of a size 60mm in all dimensions or less
Req 3.1.2.1-PT	Hub shall be powered by the power adapter
Req 3.1.2.3-PT	Hub shall connect to wifi with user wifi credential
Req 3.1.2.4-FP	User shall have a modern web browser

Table D.2 A list of Non-functional General Requirements

D.3 Hub Physical requirements

Requirement ID	Hub Physical Description
Req 3.2.1.1-FP	Hub shall be of a size 90 mm × 90 mm, excluding protruding connectors
Req 3.2.1.2-PC	Hub shall have convenient power connector port
Req 3.2.1.3-FP	Hub shall not weigh more than 400gm
Req 3.2.1.4 -FP	Hub shall be fabricated with ESD free material

Table D.3 A list of Hub Physical Requirements

D.4 Clip Physical requirements

Requirement ID	Clip Physical Description
Req 3.2.2.1-FP	Clip shall be fabricated by ESD free material
Req 3.2.2.2-FP	Clip must weigh under 50 gms
Req 3.2.2.3-FP	Clip shall have button to power on and off device

Table D.4 A list of Clip Physical Requirements

D.5 Audio Processing

Requirement ID	Audio Processing Description
Req 3.3.1.1-PC	Input audio must be recorded on a local computer as a MP3/4 file
Req 3.3.1.2-PC	A sample audio must put Google API to test to verify that the API is functional in transcribing audio to text
Req 3.3.1.3-PT	Input recording must be uploaded onto cloud storage
Req 3.3.1.4-PT	Input recording must be transcribed by Google APIs on the cloud level
Req 3.3.1.5-FP	Speaker shall be identified by the software and clearly indicated in transcriptions
Req 3.3.1.6-FP	Transcribed text must be presentable to the user through GUI

Req 3.3.1.7-FP	Input recording must be available for playback through GUI

Table D.5 A list of Audio Processing Requirements

D.6 Cloud

Requirement ID	Cloud Description
Req 3.3.2.1-PT	Cloud server must clearly organize uploaded audio files in a date formatted folder structure
Req 3.3.2.2-PT	Cloud server must reserve relevant metadata of the audio files to help with organization of uploaded audio files
Req 3.3.2.3-PT	Cloud server must home the processing power for the web application while providing adequate storage for the recordings
Req 3.3.2.4-FP	Cloud server shall use data compression of audio files to optimize storage usage
Req 3.3.1.5-FP	Recordings must have an expiry date where they are deleted in order to avoid low storage space over time

Table D.6 A list of Cloud Requirements

D.7 Web Application

Requirement ID	Web Application Description
Req 3.3.3.1-PT	The user must login to the web app in order to interact with features within the web app
Req 3.3.3.2-PT	In order to start recording a meeting on the web app, the user shall click the record button
Req 3.3.3.3-PT	The web app shall provide real-time transcription of the speaker's words while clearly indicating a new speaker is speaking
Req 3.3.3.4-FP	The web app must provide the user with the option of downloading the audio recording as an acceptable audio format
Req 3.3.3.5-FP	The web app must provide the user with the option of downloading the transcription of the meeting

Req 3.3.3.6-FP	The user shall be provided with a UI that clearly indicates recordings based on meeting titles and dates in a chronological manner
Req 3.3.3.7-FP	The meeting details must be present within Web App's UI, clearing indicating date, time, location, title of the meeting and the participants
Req 3.3.3.8-FP	The meeting details must be editable upon user's wish

Table D.7 A list of Web Application Requirements

D.8 Clip

D.8.1 Microphone requirements

Requirement ID	Microphone Description
Req 3.4.1.1.1-PC	The microphone should operate between 1.6-3.6V power supply range
Req 3.4.1.1.2-PC	The microphone normal mode of operation must operate with 1.8V/1.2mA power supply
Req 3.4.1.1.3-PT	The microphone shall have a operating temperature range from -45 to 85℃
Req 3.4.1.1.4-PT	The microphone shall incorporate for the signal to noise ratio (SNR) of at least 65 dBA
Req 3.4.1.1.5-PT	The microphone shall have a frequency response between 45Hz-20kHz at -3dB point
Req 3.4.1.1.6-PT	The microphone must have a acoustic dynamic range of about 104 dB which would be the acceptable voice for recording
Req 3.4.1.1.7-FP	The microphone must have a sensitivity of -44 dBFS at 94 dB SPL (Sound Pressure Level)
Req 3.4.1.1.8-FP	The microphone shall have total harmonic distortion of about 0.35% at 105 dB SPL to reduce any echo
Req 3.4.1.1.9-FP	The microphone must have its latency less than 30 microseconds to reduce the signal processing time

Table D.8 A list of Microphone Requirements

D.8.2 Microcontroller requirements

Requirement ID	Microcontroller Description
Req 3.4.1.2.1-PC	The power supply operating range must lie between 1.8-3.8V
Req 3.4.1.2.2-PC	The operating temperature range must be between -40 to 85 ℃
Req 3.4.1.2.3-PT	The microcontroller must have a throughput of at least 1 mbps for a reasonable data transmission
Req 3.4.1.2.4-PT	The microcontroller shall have at least 32 kb RAM data memory to store voice data
Req 3.4.1.2.5-PT	The microcontroller should be bluetooth 4.2 compliant for fast data transmission to the hub
Req 3.4.1.2.6-PT	Low energy consumption shall require 8.2mA current to be transmitted at 2.4GHz
Req 3.4.1.2.7-FP	The microcontroller must have 12 bit analog to digital converter (ADC) with 1 Msps
Req 3.4.1.2.8-FP	The microcontroller must support GPIO (General-purpose input/output) pins to incorporate buttons and LED's
Req 3.4.1.2.9-FP	The microcontroller must support UART connection

Table D.8.2 A list of Microcontroller Requirements

D.8.3 General clip requirement

Requirement ID	General Clip Description
Req 3.4.1.3.1-PT	Clip must be able to record audio of at least one person and transmit data to the hub
Req 3.4.1.3.2-FP	Multiple clips should be able to communicate with the hub and transmit data of more than one person
Req 3.4.1.3.3-FP	Clip must synchronise data transmission with the LED blinks on the hub

Table D.8.3 A list of General Clip Requirements

D.9 Hub

D.9.1 Hub Electrical Requirements

Requirement ID	Electrical Description
Req 3.4.2.1.1- PC	Hub components shall be powered by a 5V/2.5A power unit
Req 3.4.2.1.2- PC	Hub components must be resistant to external sources of heat
Req 3.4.2.1.3 - PC	Hub components must be resistant to external sources of noise
Req 3.4.2.1.4 - FP	Loss in electrical power must not result in loss of stored data on the HUB

Table D.9.1 A list of Electrical Requirements

D.9.2 Microprocessor Requirements

Requirement ID	Microprocessor Description
Req 3.4.2.2.1- PC	The microprocessor shall have Bluetooth 4.2 module to receive data from the clip[6]
Req 3.4.2.2.2- PC	The microprocessor shall consist of Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps) to send data to test laptop[6]
Req 3.4.2.2.3- PC	The operating temperature must be kept between 0 - 50 ºC for the microprocessor
Req 3.4.2.2.4 - PC	The microprocessor must be powered by 5V/2.5A DC micro-usb
Req 3.4.2.2.5 - PC	The microprocessor must have 1GB LPDDR2 SDRAM to store the incoming data from the clip
Req 3.4.2.2.6 - PC	The microprocessor shall not exceed more than 85x56 mm in size
Req 3.4.2.2.7- FP	The microprocessor shall have GPIO pins to indicate the state of the HUB (ON/OFF)

Req 3.4.2.2.8- FP	The microprocessor must have standards to ensure there is no loss in transmission of data over the cloud
Req 3.4.2.2.9- FP	The microprocessor shall consist of 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN to send data to the cloud[9]

Table D.9.2 A list of Microprocessor Requirements

D.9.3 Firmware/Operating System

Requirement ID	Firmware/Operating System Description
Req 3.4.2.3.1- PC	The microprocessor shall use Raspbian Debian based operating system version 9
Req 3.4.2.3.2- PC	The microprocessor shall use Linux Kernel version 4.14
Req 3.4.2.3.3 - PC	The installed Firmware on the microprocessor must have the capability to upgrade for future performance improvements
Req 3.4.2.3.4 - PT	The firmware shall use TCP/IP protocol to transfer data over the internet
Req 3.4.2.3.5 - FP	The software running on the Hub must be advertisement free
Req 3.4.2.3.6- FP	The software written for the microprocessor must have functions to handle external interrupts and run time errors due to various components present in the microprocessor

Table D.9.3 A list of Firmware/Operating System Requirements

D.9.4 Microprocessor Functional Requirements

Requirement ID	Microprocessor Functional Description
Req 3.4.2.4.1 - PC	The Hub must support at least connection to one Bluetooth enabled clip for the exchange of voice data
Req 3.4.2.4.2 - PC	No unethical software shall be utilized to operate the HUB
Req 3.4.2.4.3 - PC	Open Source Projects and APIs must be used to reduce the overall cost of the project
Req 3.4.2.4.4 - FP	The Hub must support connection to eight Bluetooth enabled clips for the exchange of voice data

Req 3.4.2.4.5 - FP	The Hub must use data ensuring the privacy of the users in not compromised

Table D.9.4 A list of Microprocessors Functional Requirements