Ain Shams University
Faculty of Computer & Information Sciences
Computer Science Department

Desktop Anywhere

July 2024

Ain Shams University
Faculty of Computer & Information Sciences
Computer Science Department

Desktop Anywhere

This documentation was submitted as required for the degree of bachelor's in computer and information sciences, By

Bassant Hossam Mohamed [Computer Science Department] - 20201701232
Hossam Hatem Abdel-Baky [Computer Science Department] – 20201701235
Sara Mazhar Mahmoud [Computer Science Department] - 20201700337
Andrew Samir Barsoum [Computer Science Department] - 20201700156
Rehab Youssef Hagag [Computer Science Department] - 20201700255

Under Supervision of

Dr. Ahmed Salah
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.
TA. Zeina Rayan
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

July 2024

Acknowledgement

All praise and thanks to ALLAH, who provided us with the ability to complete this work. We hope to accept this work from us.

We are grateful to *our family*, *especially our parents* who are always providing help and support throughout the whole years of study. We hope we can give that back to them.

We also offer our sincerest gratitude to our supervisors, *Dr. Ahmed Salah*, *T.A. Zeina Rayan,* who have supported us throughout our thesis with their patience, knowledge, and experience.

Finally, we would thank our *technical mentors Eng. Manar Sultan and Eng. Reem Telbani* who gave us support and encouragement.

Abstract

The rapid advancement of technology has significantly impacted our lives, work, and interactions. It has significantly shaped human civilization, making repetitive tasks easier in terms of time, effort, or both.

Desktop device users face a variety of obstacles while using desktop computers, including Limited Physical Access as Users face the challenge of being physically present at their desktop computers to access or control them, inefficient File and Folder Retrieval in terms of time especially where huge data exists, time-consuming data management across many PCs, also users with disabilities may have difficulty typing, while others may struggle with manual activities such as application launches or shutdowns. Furthermore, individuals who do not properly schedule their work risk becoming overwhelmed, lowering pleasure and productivity.

These obstacles highlight the need for innovative solutions to improve user experience and efficiency. Also, as a response and a contribution to the global orientation, Desktop Anywhere is here. It is a software which developed as a comprehensive and integrated system consisting of two interconnected components: a mobile application and a desktop application communicating through a server. Desktop Anywhere is developed to enable users to access and control multiple personal desktop computers remotely. It aims to provide a seamless experience for users to interact with their desktop screens, manage files, execute tasks through virtual touchpad and keyboard, and utilize voice commands in Arabic, which is done by more than 95% accuracy, all through their mobile devices which will have the ability to be automatically connected to all paired devices without authentication. The primary goal is to bridge the gap between devices and redefine remote interactions, enhancing convenience, efficiency, and accessibility in the realm of digital connectivity.

Table of Contents

List of Figures

List of Abbreviations

- **API:** Application programming interface.
- **ASR:** Automatic Speech Recognition.
- **LAN:** Local area network.
- **MLM:** Masked Language Modeling.
- **IP**: Internet Protocol.
- **NAT**: Network Address Translation.
- **NSP:** Next Sentence Prediction.
- **NLU:** Natural Language Understanding.
- **NER:** Named Entity Recognition.
- **RDP:** Remote Desktop Protocol .
- **SOT**: start of token.
- **STUN** server: Session Traversal Utilities for NAT.
- **TURN** server: Traversal Using Relays around NAT.
- **VNC:** Virtual Network Computing.
- **WebRTC:** Web Real-Time Communication.
- **WER:** Word Error rate.
- **Wi-Fi:** Wireless Fidelity

Chapter 1
<Introduction>

Motivation

The project, "Desktop Anywhere," is a necessary response to the evolving landscape of desktop computing and the challenges faced by users in this field. The rapid advancement of technology has transformed the way we live and work, demanding innovative solutions to enhance user experience and productivity. Within this context, several factors in the field of desktop computing have motivated the idea behind this project.

Firstly, the increasing reliance on desktop computers as essential tools for work and personal tasks has highlighted the need for improved accessibility and flexibility. Users often face the inconvenience of being physically tied to their desktops, limiting their mobility, and hindering productivity. The project addresses this need by enabling remote access and control of desktop computers, empowering users to connect and manage their devices from anywhere.

Secondly, the growing complexity of digital workflows and data management has created challenges in locating and retrieving files and folders efficiently. Time-consuming searching and manual synchronization across multiple PCs have become major obstacles to productivity. The project aims to streamline these processes, providing users with seamless file and folder retrieval, improving efficiency, and saving valuable time.

Furthermore, the project recognizes the importance of inclusivity and accessibility in the digital realm. Users with disabilities face additional barriers when interacting with desktop devices, which can hinder their productivity and independence. By offering alternative methods of interaction, such as virtual touchpad, keyboard, and voice commands, the project caters to the needs of users with disabilities, promoting inclusivity and equal access to technology.

Moreover, the project aligns with the global trend of remote work and digital collaboration. As the world becomes more interconnected, the demand for effective remote access and control solutions has increased significantly. The project embraces this trend, providing users with the ability to seamlessly connect to and manage their desktops remotely, regardless of their physical location.

In conclusion, the project, "Desktop Anywhere," is a necessary and timely response to the challenges faced in the field of desktop computing. The need for improved accessibility, streamlined data management, inclusivity, and remote

connectivity has motivated the idea behind this project. By addressing these needs, the project enhances user experience, improves productivity, and aligns with the ongoing advancements in remote work and digital collaboration.

Problem Definition

Desktop device users face a range of challenges and problems that impact their user experience and productivity. These challenges include limited physical access, Users often need to be physically present at their desktop computers, which can be inconvenient and restrict flexibility, inefficient file and folder retrieval as Locating and retrieving files and folders can be time-consuming and frustrating, particularly when dealing with large amounts of data. Time-consuming data management across multiple PCs, difficulties for users with disabilities. Additionally, users who struggle with work scheduling may experience decreased productivity and feel overwhelmed. These problems highlight the need for solutions that enhance user experience, efficiency, and accessibility in the realm of desktop computing.

Objectives

- **Increase Search Efficiency:** Avoid large consumed retrieval time as we work one sync all files existed on the desktop to be existed and collected in a database then when user search on it we retrieve the targeted file from the synced database of the mobile app.
- **Solve problem of Limited Physical Access:** Users face the challenge of being physically present at their desktop computers to access or control them .Thus, we Enabled remote control of desktop devices, as our objective is to provide users with a virtual touchpad and keyboard interface, remote control, and voices commands features which eliminates the physical proximity constraint and enables users to interact with their desktops from a distance in very different ways and modes to enhance flexibility.
- **Enhance the Availability of data:** Achieved through data synchronization.
- **Break Language barrier:** About 420 million people speak Arabic worldwide. “By 2021” and still most of apps in the market target english speakers only. Thus, The objective is to cater to Arabic speakers who may have limited proficiency in other languages or face challenges in using

traditional input methods. By offering voice commands and a user-friendly interface, the project aims to make electronic devices more accessible and inclusive for Arabic speakers, empowering them to leverage technology effectively.

- **Enhance Managing Multi-tasks:** Can be overwhelming without efficient task scheduling, multidevice connection. Thus, we enabled the users with two features works together to solve this problem. The first one is the ability to connect to multiple devices concurrently. The second one is the ability to shcedule some commands.
- **Enhance productivity and efficiency through voice-enabled commands:** The objective is to simplify and expedite common tasks such as file/folder search, alarm setting, note taking, and device control (shutdown, sleep, restart, and app closure) by allowing users to perform these actions through voice commands. This reduces the reliance on manual input methods and streamlines the user experience.
- **Provide seamless integration with multiple devices:** The objective is to offer an autoconnection feature that facilitates easy pairing and connectivity with previously paired devices. This eliminates the need for repetitive setup processes and ensures a smooth transition between devices, enhancing convenience and user satisfaction.
- **Enhance user experience and satisfaction:** The objective is to create a system that is intuitive, accurate, and efficient in understanding and responding to voice commands. By leveraging advanced natural language processing techniques, the project aims to provide a seamless and enjoyable user experience, thereby improving overall satisfaction with electronic devices.

  By addressing these problems, the project aims to create a user-centric system that enhances the experience of Arabic speakers when interacting with electronic devices, ultimately improving productivity, accessibility, and satisfaction.

Time plan

ID	Task Name	Predecessor	Duration	Members
1	Survey and learning	----------	3 weeks	5 members
2	Requirements Specifications	----------	3 weeks	5 members
3	Project Analysis	----------	2 weeks	5 members
4	Project Architecture	----------	2 weeks	5 members
5	User Interface Design	----------	1 week	5 members
6	Mobile Application	----------	1 month	2 members
7	Desktop Application & API Server	----------	1 month	3 members
8	Testing (Task 6)	Mobile Application	1 week	2 members
9	Voice Command	Mobile Application, API Server, and Desktop Application.	3 weeks	5 members
10	Testing (Task 9)	Voic Command	1 week	2 members
11	Testing (Task 7)	Desktop Application &API Server	1 week	3 members

12	Integration	Desktop Application, Mobile Application, Voice Command and API Server	1 weeks	5 members
13	Scheduled Tasks via Voice	Desktop Application, Mobile Application, Voice Command and API Server	5 weeks	2 members
14	Virtual Touchpad/Keyboard	API server, mobile application, and desktop application	1 month	3 members
15	Testing (Task 13)	Desktop Application, Mobile Application, Scheduled Tasks via Voice, and API Server	1 week	2 members
16	Testing (Task 14)	Desktop Application, Mobile Application, Virtual Touchpad/Keyboard and API Server	1 week	3 members
17	Autoconnection	Desktop Application, Mobile Application and API Server	1 month	3 members
18	File and Folder Search via Voice	Desktop Application, Mobile Application, Voice Command and API Server	3 weeks	2 members
19	Testing (Task 17&18)	API Server, Mobile Application, Autoconnection, File and Folder Search, Voice Command and Desktop Application	1 week	5 members
20	Multi-Device Sync	API Server, Mobile Application and Desktop Application	3 weeks	5 members
21	Testing (whole project)	Whole Project	1 week	5 members

Scope and Document Organization

The approximate work involved to finish the project is divided into these five phases: -

1. Planning:
   - Collecting team members.
   - Selecting suitable idea
   - Setting a Gantt chart for the project.
   - Determining the resources of the team.

2. Analysis:
   - Collecting data about the project and the lack that made us in a need to Our system.
   - Determining the functional and non-functional requirements.

3. Designing:
   Determining the diagrams to be carried out within the project: -
   - Use-case Diagram.
   - System Architecture.
   - Sequence Diagrams.
   - Block Diagrams.

4. Implementation:
   - Generating Dataset.
   - Model Coding.
     - Whisper Model.
     - Arabert Model for Classification task.
     - Arabert Model for NER task.
   - Mobile App Implementation:
     - Design App Interface.
     - App Development.
   - Desktop App Implementation:
     - Design App Interface.
     - App Development.
   - Server Implementation.
   - Integration.

5. Testing:
    - Functional Testing.
    - Non-functional Testing.

6. Deployment.

7. Documentation:
The documentation should mainly include these main chapters: -

- Background: In this chapter, we are going to discuss Background/Literature Review that includes citation of theoretical background, related work and results enhancement should be included.
- Project planning: In this chapter, we are going to discuss and go deeper in how we plan the project and show the steps and the instructions that we have followed to plan the System.
- System Analysis: In this chapter, we are going to discuss and go deeper into a detailed understanding of the business needs, Studying and analyzing the current systems, Existing Systems limitations and details, and Defining new system objectives.
- System Design: In this chapter, we are going to discuss and go deeper in the System design and present its diagrams.
- System Implementation: In this chapter we're going to discuss and go deeper in our system implementation, and present its code and the algorithms used to build it.
- System Testing and Deployment: In this chapter we're going to discuss and go deeper in our system testing, and present the types of testing to be used and test cases we examined our application through.
- User Manual: : In this chapter we're going to discuss how to operate the project along with screen shots of the project representing all steps, in another terms This chapter will also include an "Installation Guide" that would describe how to install the program, and all required third party tools that needs to be available for the project to run.
- Results and Discussions: includes the collected result and/or achieved final product capabilities.
- Conclusions and recommendations: In this chapter we're going to discuss a breif about the project, and recommendations for future work.

General constraints

- Learning new technologies may take a lot of time.
- Limited Free Advanced Learning resources.
- Underestimating the objectives that may not lead to realistic or achievable function.
- No avaliable dataset exists, No avaliable resources to collect the dataset, So we have to generate our dataset

Chapter 2
<Background>

Transformers: A Brief Overview

Transformers are a groundbreaking model architecture introduced by Vaswani et al. in their 2017 paper, "Attention Is All You Need." They have revolutionized natural language processing (NLP) and other fields due to their efficiency and ability to handle long-range dependencies in data.

Key Features of Transformers:

1. **Attention Mechanism**: Enables the model to weigh the importance of different words in a sequence, allowing it to capture context effectively.
2. **Self-Attention**: Each word in the input sequence can attend to all other words, capturing relationships and dependencies.
3. **Parallel Processing**: Unlike recurrent neural networks (RNNs), transformers process words simultaneously, leading to faster computation.
4. **Encoder-Decoder Structure**: Typically consists of an encoder to process input sequences and a decoder to generate output sequences.

Transformers have become the backbone of many advanced NLP models, including BERT due to their robust performance in various tasks such as translation, summarization, and text generation.

Whisper: Built on Transformers

Whisper is an advanced speech recognition model developed by OpenAI, leveraging the powerful transformer architecture. It excels in converting spoken language into written text, enabling accurate and efficient transcription.

How Whisper Uses Transformers:

1. **Attention to Detail**: Whisper uses transformers to focus on different parts of the audio signal, ensuring that even subtle nuances in speech are captured.
2. **Contextual Understanding**: The self-attention mechanism allows Whisper to understand the context of spoken words, improving the accuracy of transcriptions.
3. **Efficiency**: By processing input audio in parallel, Whisper can quickly and accurately transcribe speech, making it highly effective for real-time applications.

In summary, transformers provide the foundational architecture for Whisper, enabling it to achieve state-of-the-art performance in speech recognition. Their ability to handle complex dependencies and process information efficiently makes them ideal for such advanced applications

Why we choose Whisper

We aim to develop a multilingual model capable of accurately recognizing voice commands given in a mix of English and Arabic or Arabic only within the same sentence. Whisper, developed by OpenAI, stands out as a state-of-the-art speech recognition model with several key advantages over traditional systems:

1. **Unified Model for Multiple Tasks**:
    - Whisper is designed to handle various speech processing tasks such as transcription, translation, voice activity detection, alignment, and language identification within a single model. This reduces the complexity of having multiple specialized systems and streamlines the overall process.
2. **Minimal Pre-processing**:
    - Whisper models predict raw text transcripts without significant standardization, avoiding the need for a separate inverse text normalization step. This simplifies the speech recognition pipeline and reduces preprocessing overhead.
3. **Robust to Diverse Audio Sources**:
    - The training dataset for Whisper includes a broad distribution of audio from various environments, recording setups, speakers, and languages. This diversity helps the model to be more robust and perform well across different real-world conditions.
4. **Effective Handling of Machine-Generated Transcripts**:
    - Whisper employs several heuristics to detect and remove machine-generated transcripts from the training dataset. This ensures that the model is trained on high-quality human-generated transcripts, enhancing its performance and reliability.
5. **Contextual Understanding**:
    - The decoder in Whisper is trained to condition on the history of the transcript text, enabling it to use longer-range text context to resolve ambiguous audio. This improves the accuracy and coherence of transcriptions.
6. **Advanced Tokenization and Language Handling**:
    - Whisper uses a byte-level BPE text tokenizer, like GPT-2, which is refit for multilingual models to avoid excessive fragmentation. This

allows effective handling of multiple languages and ensures the model can process diverse linguistic inputs accurately.

7. **Simplified Pipeline for Timestamp Prediction**:
    - Whisper integrates timestamp prediction within the transcription process, providing precise time annotations for spoken words. This feature is useful for applications requiring time-aligned transcripts, such as subtitling and video editing.
8. **Scalable and Reliable Architecture**:
    - Built on the well-validated encoder-decoder Transformer architecture, Whisper scales reliably for large-scale supervised pre-training, ensuring robust performance across extensive datasets.
9. **Improved Data Quality**:
    - Automated filtering methods improve transcript quality by removing subpar and machine-generated transcripts. Additionally, audio language detection ensures that the spoken language matches the transcript, further enhancing data integrity.
10. **Efficient Training and Deployment**:
    - By consolidating multiple tasks into a single model and reducing preprocessing steps, Whisper offers efficient training and deployment, making it a practical solution for various speech recognition applications
11. **Multilingual Capabilities:**
    - Whisper excels in recognizing and processing multiple languages, making it ideal for handling mixed-language commands seamlessly.
12. **High Accuracy:**
    - The model's advanced architecture ensures high accuracy in transcription, even with complex and diverse linguistic inputs.
13. **Zero-shot Performance:**
    - Whisper demonstrates strong zero-shot capabilities, enabling it to perform well on languages and tasks it has not been explicitly trained on, which is beneficial for mixed-language scenarios.
14. **Scalability:**
    - The model scales efficiently, allowing for training and deployment in different environments while maintaining performance.

Detailed comparison of effective robustness across various datasets. Although both models perform within 0.1% of each other on LibriSpeech, a zero-shot Whisper model performs much better on other datasets than expected for its LibriSpeech performance and makes 55.2% less errors on average. Results reported in word error rate (WER) for both models after applying our text normalizer.

Dataset	wav2vec 2.0 Large (no LM)	Whisper Large V2	RER (%)
LibriSpeech Clean	**2.7**	**2.7**	0.0
Artie	24.5	**6.2**	74.7
Common Voice	29.9	**9.0**	69.9
Fleurs En	14.6	**4.4**	69.9
Tedlium	10.5	**4.0**	61.9
CHiME6	65.8	**25.5**	61.2
VoxPopuli En	17.9	**7.3**	59.2
CORAAL	35.6	**16.2**	54.5
AMI IHM	37.0	**16.9**	54.3
Switchboard	28.3	**13.8**	51.2
CallHome	34.8	**17.6**	49.4
WSJ	7.7	**3.9**	49.4
AMI SDM1	67.6	**36.4**	46.2
LibriSpeech Other	6.2	**5.2**	16.1
Average	29.3	**12.8**	55.2

Table 2.1 effective robustness [1]A. Radford, J. Kim, T. Xu, G. Brockman, C. Mcleavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision." Available: https://arxiv.org/pdf/2212.04356

Whisper is competitive with state-of-the-art commercial and open-source ASR systems in long-form transcription. The distribution of word error rates from six ASR systems on seven long-form datasets are compared, where the input lengths range from a few minutes to a few hours. The boxes show the quartiles of per-example WERs, and the per-dataset aggregate WERs are annotated on

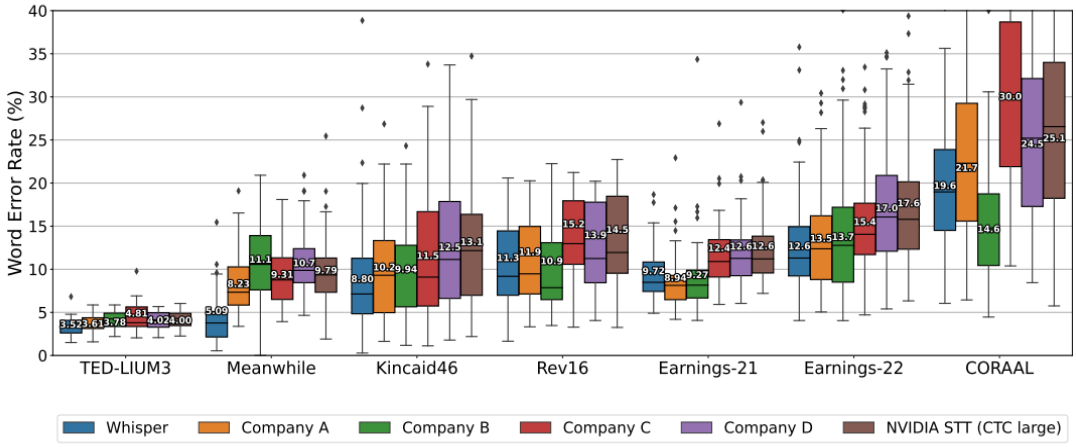


Figure 2.1 The distribution of word error rates from six ASR systems [1]A. Radford, J. Kim, T. Xu, G. Brockman, C. Mcleavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision." Available: https://arxiv.org/pdf/2212.04356

each box. Our model outperforms the best open-source model (NVIDIA STT) on all datasets, and in most cases, commercial ASR systems as well.

Whisper's performance is close to that of professional human transcribers. This plot shows the WER distributions of 25 recordings from the Kincaid46 dataset transcribed by Whisper, the same 4 commercial ASR systems from Figure 6 (A-D), one computer-assisted human transcription service (E) and 4 human transcription services (F-I). The box plot is superimposed with dots indicating the WERs on individual recordings, and the aggregate WER over the 25 recordings are annotated on each box.

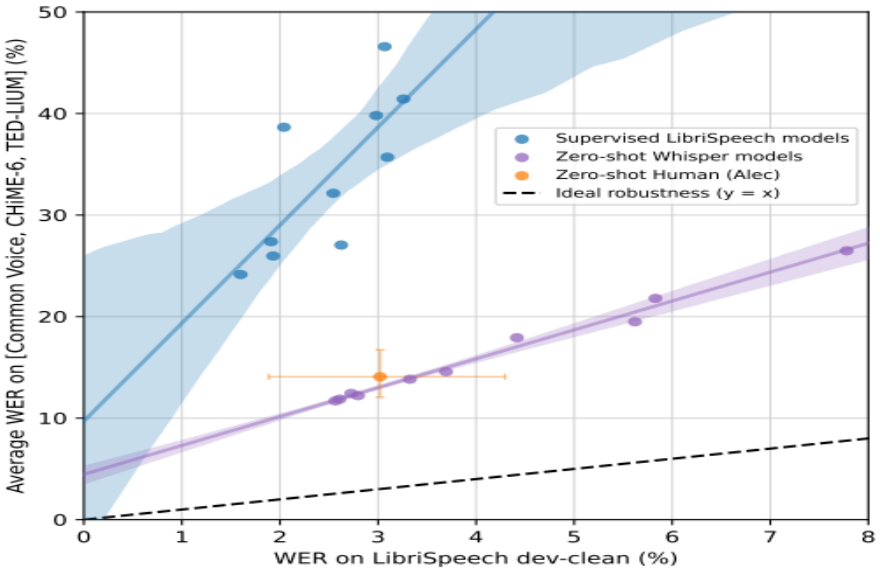


Figure 2.2 Whisper's performance to professional human transcribers A. Radford, J. Kim, T. Xu, G. Brockman, C. Mcleavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision." Available: https://arxiv.org/pdf/2212.04356

Multilingual speech recognition performance. Zeroshot Whisper improves performance on Multilingual LibriSpeech (MLS) but is still significantly behind both Maestro, XLS-R, and mSLAM on VoxPopuli.

Model	MLS	VoxPopuli
VP-10K + FT	-	15.3
XLS-R (1B)	10.9	10.6
mSLAM-CTC (2B)	9.7	9.1
Maestro	-	**8.1**
Zero-Shot Whisper	**7.3**	13.6

Table 2.2 [1]A. Radford, J. Kim, T. Xu, G. Brockman, C. Mcleavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision." Available: https://arxiv.org/pdf/2212.04356

Correlation of pre-training supervision amount with downstream speech recognition performance. The amount of pre-training speech recognition data

for a given language is very predictive of zero-shot performance on that language in Fleurs.

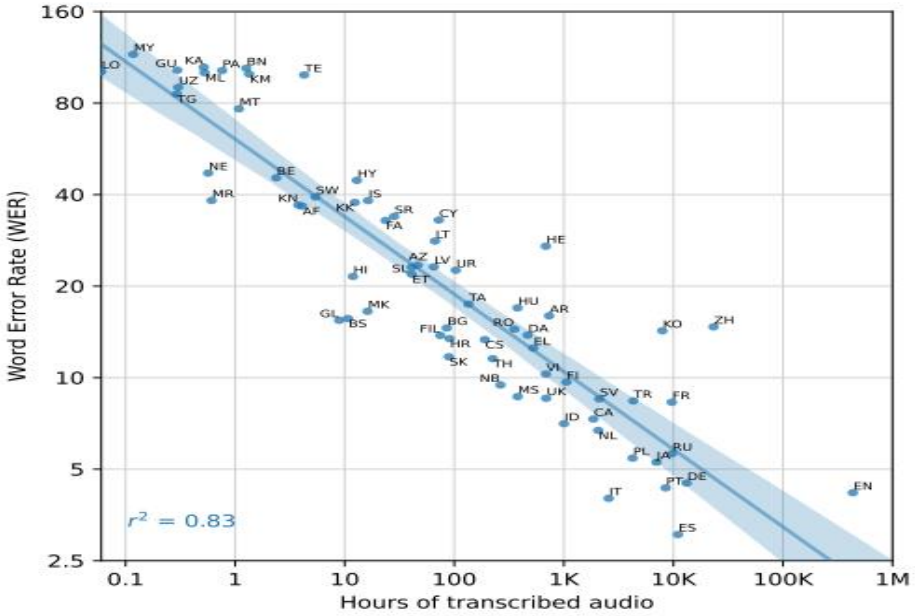


Figure 23 Whisper's performance to professional human transcribers A. Radford, J. Kim, T. Xu, G. Brockman, C. Mcleavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision." Available: https://arxiv.org/pdf/2212.04356

Whisper: Approach

Data Processing:

- **Minimal Pre-processing**:
  - Whisper models predict raw text transcripts without significant standardization.
  - Avoids the need for a separate inverse text normalization step.
- **Dataset Construction**:
  - Compiled from audio paired with transcripts available on the internet.
  - Covers a wide range of environments, setups, speakers, and languages.
- **Transcript Quality**:
  - High variability in transcript quality, necessitating automated filtering.
  - Developed heuristics to detect and remove machine-generated transcripts.

- **Machine-Generated Transcript Detection**:
  - Identified through characteristics such as lack of punctuation, uniform case (all-uppercase/lowercase), and simple rule-based text normalization.
- **Language Matching**:
  - Used an audio language detector to ensure spoken language matches the transcript.
  - Exceptions made for English transcripts, using them for speech translation training.
- **De-duplication**:
  - Applied fuzzy de-duping to reduce duplication and automatically generated content.
- **Segmentation**:
  - Audio files segmented into 30-second intervals, paired with corresponding transcript segments.
  - Included non-speech segments with sub-sampled probability for voice activity detection training.
- **Manual Inspection**:
  - Identified and removed low-quality data sources through inspection based on error rate and source size.
  - Removed poorly transcribed or misaligned transcripts and undetected low-quality machine-generated captions.
- **Evaluation Dataset De-duplication**:
  - Performed de-duplication between training and evaluation datasets, particularly TED-LIUM 3, to avoid contamination.

Whisper Model Architecture:

- **Objective**:
  - Focus on studying large-scale supervised pre-training for speech recognition.
  - Utilizes an off-the-shelf encoder-decoder Transformer architecture to ensure findings are not confounded by model improvements.
- **Audio Pre-processing**:
  - All audio resampled to 16,000 Hz.
  - Compute an 80-channel log-magnitude Mel spectrogram on 25-millisecond windows with a 10-millisecond stride.
  - Feature normalization by scaling input to be between -1 and 1 with approximately zero mean across the dataset.

- **Encoder**:
  - Processes input with a small stem consisting of two convolution layers with a filter width of 3 and GELU activation.
  - The second convolution layer has a stride of two.
  - Sinusoidal position embeddings are added to the stem output.
  - Transformer blocks applied using pre-activation residual blocks.
  - Final layer normalization applied to the encoder output.
- **Decoder**:
  - Uses learned position embeddings.
  - Employs tied input-output token representations.
  - Shares the same width and number of transformer blocks as the encoder.
- **Tokenizer**:
  - Uses the same byte-level BPE text tokenizer as GPT-2 for English-only models.
  - Refit vocabulary for multilingual models to avoid excessive fragmentation in non-English languages, while keeping the same vocabulary size.

Multitask Format in Whisper Model

1. **Unified Model for Multiple Tasks**:
   - Aim to reduce complexity by having a single model handle the entire speech processing pipeline.
   - Incorporates tasks such as transcription, translation, voice activity detection, alignment, and language identification.
2. **Task Specification**:
   - Use a simple sequence of input tokens to specify tasks and conditioning information for the decoder.
   - The decoder is an audio-conditional language model that also conditions the transcript's text history.
3. **Contextual Conditioning**:
   - With some probability, the transcript text preceding the current audio segment is added to the decoder’s context.
   - This helps the model use longer-range text context to resolve ambiguous audio.
4. **Prediction Process**:
   - Begin prediction with a <|startoftranscript|> token.
   - Predict the language being spoken using unique tokens for each language (99 total).
   - If no speech is detected, predict a <|nospeech|> token.

- Specify the task with either a <|transcribe|> or <|translate|> token.
- Indicate whether to predict timestamps with a <|timestamps|> token.

5. **Timestamp Prediction**:
   - Quantize times to the nearest 20 milliseconds, matching the native time resolution of Whisper models.
   - Add start and end time tokens to the vocabulary.
   - Interleave time tokens with caption tokens:
     - Predict start time token before the caption's text.
     - Predict end time token after the caption's text.
   - For partially included transcript segments, predict only the start time token in timestamp mode or truncate the audio.
6. **Training Process**:
   - Mask out the training loss over the previous context text.
   - Train the model to predict all other tokens.

Overview of approach. A sequence-to-sequence Transformer model is trained on many different speeches processing tasks, including multilingual speech recognition, speech translation, spoken language identification, and voice activity detection. All these tasks are jointly represented as a sequence of tokens to be predicted by the decoder, allowing for a single model to replace many different stages of a traditional speech processing pipeline. The multitask training format uses a set of special tokens that serve as task specifiers or classification targets, as further explained.

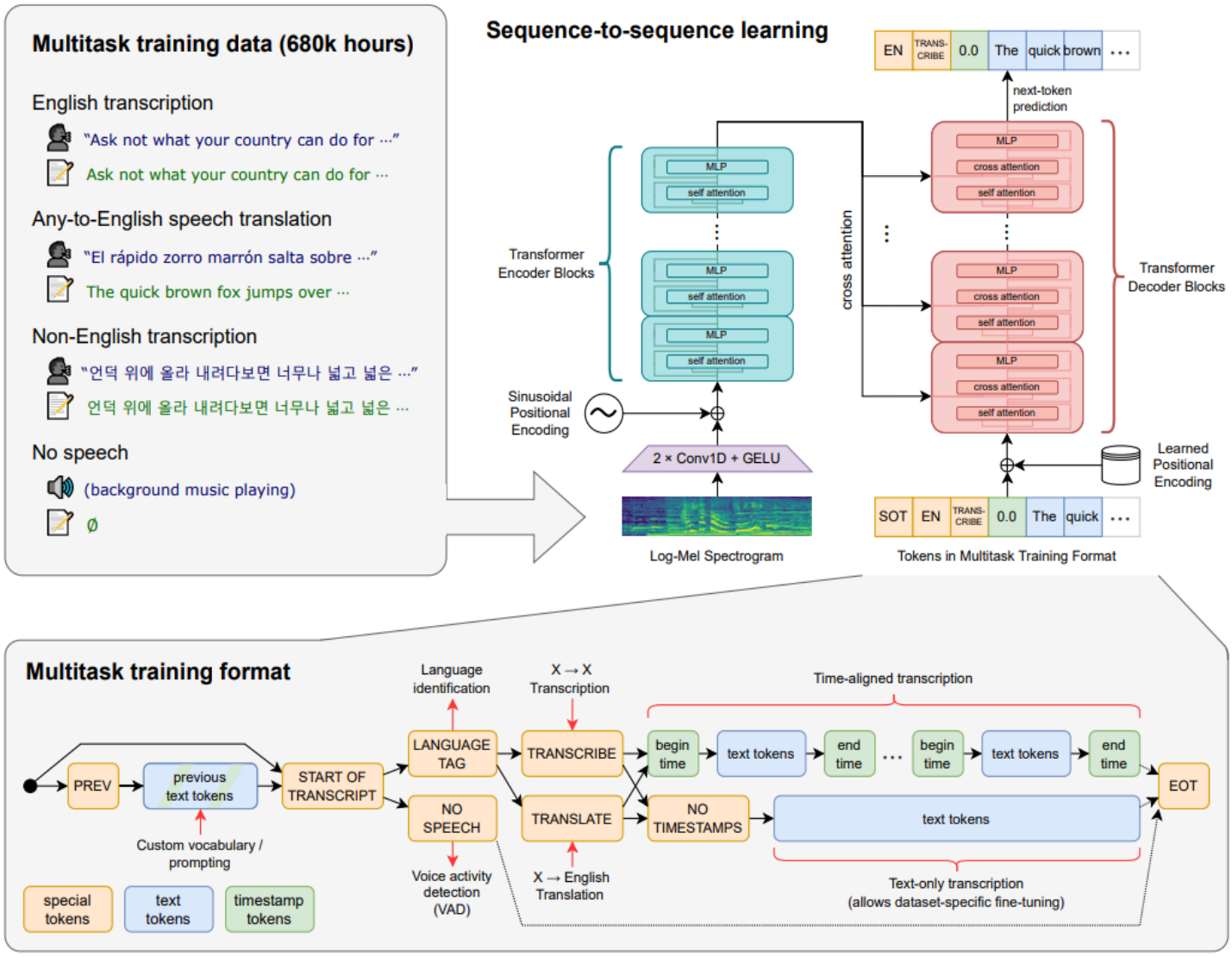


Figure 2.4- Overview of whisper approach A. Radford, J. Kim, T. Xu, G. Brockman, C. Mcleavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision." Available: https://arxiv.org/pdf/2212.04356

Training Details of Whisper Models

1. **Model Variety**:
    - Suite of models trained in various sizes to study scaling properties.

2. **Training Techniques**:
    - Data parallelism across accelerators.
    - Use of FP16 (16-bit floating point) with dynamic loss scaling and activation checkpointing.
3. **Optimizer and Techniques**:
    - Trained with AdamW optimizer.
    - Gradient norm clipping applied.
    - Linear learning rate decay to zero after initial warmup over 2048 updates.
4. **Training Parameters**:
    - Batch size: 256 segments.
    - Training duration: 220 updates (2-3 passes over the dataset).
5. **Overfitting Management**:
    - Overfitting is not a significant concern due to limited training epochs.
    - No data augmentation or regularization used.
    - Rely on large dataset diversity for generalization and robustness.

Whisper Performance under Noise:

- Under low noise (40 dB SNR), many models outperform Whisper's zero-shot performance.
- As noise intensity increases, all models degrade in performance.

Whisper Robustness:

- Whisper outperforms other models under intense noise conditions (SNR below 10 dB), especially in natural noisy environments like pubs.
- Demonstrates Whisper's robustness to real-world noise scenarios.

Whisper compared with human performance:

- **Irreducible Error Factors:**
    - Ambiguous/indistinct speech and labeling errors contribute to irreducible error in datasets.
- **Human Performance Comparison:**
    - Selected 25 recordings from the Kincaid46 dataset.
    - Used 5 transcription services (4 human, 1 computer-assisted) for obtaining transcripts.

- **Recording Conditions:**
    - Variety of conditions: scripted/unscripted broadcasts, telephone/VoIP calls, and meetings.
- **Performance Metrics:**
    - Distribution of per-example WERs and aggregate WER analyzed across 25 recordings.
- **Results:**
    - The computer-assisted transcription service had the lowest aggregate WER, 1.15% better than Whisper's.
    - Pure-human performance was only slightly better than Whisper's.
- **Conclusion:**
    - Whisper's English ASR performance is very close to human-level accuracy, though not perfect.

Zero-shot Whisper models close the gap to human robustness. Despite matching or outperforming a human on LibriSpeech dev-clean, supervised LibriSpeech models make roughly twice as many errors as a human on other datasets demonstrating their brittleness and lack of robustness. The estimated robustness frontier of zero-shot Whisper models, however, includes the 95% confidence interval for this human.

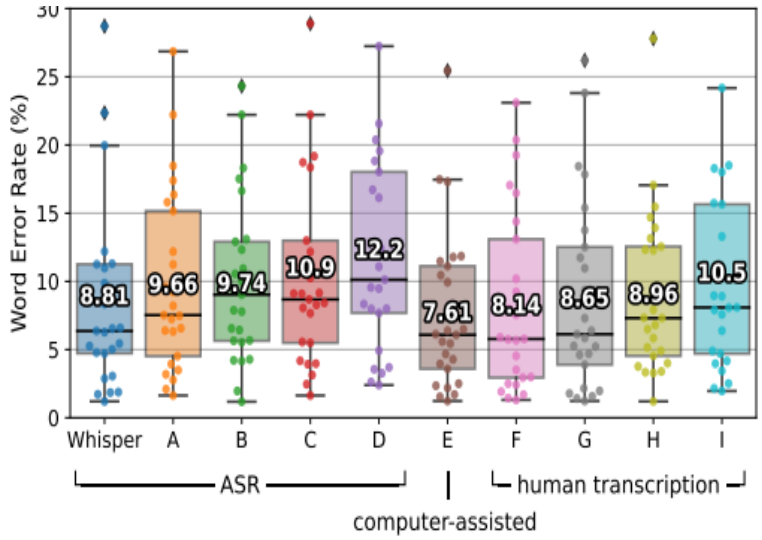


Figure 2.5- Zero-shot Whisper models close the gap to human robustnessA. Radford, J. Kim, T. Xu, G. Brockman, C. Mcleavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision." Available: https://arxiv.org/pdf/2212.04356

Brief Overview of Word Error Rate (WER)

Definition: Word Error Rate (WER) is a common metric used to evaluate the performance of speech recognition systems. It measures the differences between the predicted transcription and the reference (correct) transcription.

Calculation: WER is calculated using the formula:

$$\text{WER} = \frac{S+D+I}{N}$$

Where:

- S = Number of substitutions (incorrect words)
- D = Number of deletions (missing words)
- I = Number of insertions (extra words)
- N = Total number of words in the reference transcription

Process:

1. **Substitutions:** Words in the predicted transcription that are incorrect compared to the reference transcription.
2. **Deletions:** Words that are missing in the predicted transcription but present in the reference transcription.
3. **Insertions:** Extra words in the predicted transcription that are not present in the reference transcription.
4. **Normalization:** Often, both the predicted and reference transcriptions are preprocessed to standardize formatting, such as removing punctuation and converting to lowercase, to ensure fair comparison.

AraBERT

Pretrained contextualized text representation models have driven significant advancements in Natural Language Understanding (NLU) tasks, achieving state-of-the-art performance across various applications. Early models like Word2vec and GloVe captured distributed syntactic and semantic properties of words but lacked contextual embedding. This gap was filled by models like ELMO, which generated contextualized representations.

Recent focus has shifted to fine-tuning large pretrained language models for specific NLU tasks, leveraging unsupervised pre-training. However, this approach requires massive corpora and substantial computational resources,

restricting its availability mainly to English and a few other languages. Multilingual models, designed to represent over 100 languages, often fall behind single-language models due to limited data and language-specific vocabularies.

Arabic, with its unique morphological and syntactic structure, presents additional challenges for multilingual models. To address this, we introduce ARABERT, a BERT transformer model pretrained specifically for Arabic. Evaluated on Sentiment Analysis, Named Entity Recognition, and Question Answering tasks, ARABERT achieves state-of-the-art performance on most datasets, outperforming previous multilingual and single-language approaches.

ARABERT: Methodology

1. **Objective:**
    - Develop an Arabic language representation model, ARABERT, to improve the state-of-the-art in several Arabic NLU tasks.
2. **Model Foundation:**
    - Based on the BERT model, a stacked Bidirectional Transformer Encoder (Devlin et al., 2018).
    - Utilizes BERT-base configuration:
        - 12 encoder blocks.
        - 768 hidden dimensions.
        - 12 attention heads.
        - 512 maximum sequence length.
        - ~110M parameters.
3. **Preprocessing:**
    - Introduced additional preprocessing specific to the Arabic language prior to pretraining.
4. **Components:**
    - Pre-training setup.
    - Pre-training dataset for ARABERT.
    - Arabic-specific preprocessing.
    - Fine-tuning process.

Pre-training Setup

- **Masked Language Modeling (MLM):**
    - 15% of input tokens are selected for replacement.
    - Replacements: 80% [MASK] token, 10% random token, 10% original token.

    - Whole-word masking improves the task by predicting the entire word.
- **Next Sentence Prediction (NSP):**
    - Helps the model understand relationships between sentences, useful for tasks like Question Answering.

Pre-training Dataset

- **Sources:**
    - Arabic Wikipedia Dumps.
    - Manually scraped Arabic news websites.
    - 1.5 billion words Arabic Corpus (El-Khair, 2016).
    - OSIAN: Open-Source International Arabic News Corpus (Zeroual et al., 2019).
- **Dataset Size:**
    - 70 million sentences, approximately 24GB of text.
    - Covers diverse topics and regions in the Arab world.
    - Preserved words with Latin characters to avoid information loss.

Sub-Word Units Segmentation

- **Lexical Sparsity:**
    - Arabic has complex concatenative morphology, causing redundancy.
- **Segmentation:**
    - Used Farasa (Abdelali et al., 2016) for segmenting words into stems, prefixes, and suffixes.
    - Example: "Alloga" becomes "Al+log+a".
- **Sentence Piece Tokenizer:**
    - Trained in unigram mode on segmented dataset.
    - Created a subword vocabulary of ~60K tokens.
    - Evaluated impact by training another version on non-segmented text (AraBERTv0.1).
    - Final vocabulary size: 64k tokens, with 4K unused tokens for further pre-training.

Fine-tuning

- **Sequence Classification:**
    - Used final hidden state of the "[CLS]" token.
    - Added a feed-forward layer with Softmax for probability distribution over classes.

    - Jointly trained classifier and pre-trained model weights to maximize log-probability of correct class.
- **Named Entity Recognition (NER):**
    - Used IOB2 format: "B" for beginning, "I" for inside, "O" for outside named entities.
    - Treated as a multi-class classification process.
    - Only input the first sub-token of each word after using AraBERT tokenizer

By integrating specific preprocessing steps, large pre-training datasets, and effective tokenization strategies, ARABERT is fine-tuned for Arabic NLU tasks, delivering state-of-the-art performance in sequence classification and named entity recognition.

Performance of AraBERT on Arabic downstream tasks compared to mBERT and previous state of the art system

Task	metric	prev. SOTA	mBERT	AraBERTv0.1/ v1
SA (HARD)	Acc.	95.7*	95.7	**96.2** / 96.1
SA (ASTD)	Acc.	86.5*	80.1	92.2 / **92.6**
SA (ArsenTD-Lev)	Acc.	52.4*	51.0	58.9 / **59.4**
SA (AJGT)	Acc.	92.6**	83.6	93.1 / **93.8**
SA (LABR)	Acc.	**87.5**†	83.0	85.9 / 86.7
NER (ANERcorp)	macro-F1	81.7††	78.4	**84.2** / 81.9
QA (ARCD)	Exact Match	mBERT	**34.2**	30.1 / 30.6
	macro-F1		61.3	61.2 / **62.7**
	Sent. Match		90.0	**93.0** / 92.0

* (ElJundi et al., 2019)
** (Dahou et al., 2019b)
† (Dahou et al., 2019b)
†† Previous state of the art performance by BiLSTM-CRF model

Table 2.3- Performance of AraBERT on Arabic downstream tasks compared to mBERT and previous state of the art system W. Antoun, F. Baly, and H. Hajj, "AraBERT: Transformer-based Model for Arabic Language Understanding," 2020. Accessed: Jun. 26, 2024. [Online]. Available: https://arxiv.org/pdf/2003.00104v2

Results in Table 1 show that AraBERTv0.1 improved results by 2.53 points in F1 score scoring 84.2 compared with the Bi-LSTM-CRF model, making AraBERT the new state-of-the-art for NER on AN-ERcorp. Testing AraBERT with tokenized suffixes and prefixes showed results similar to that of the Bi-LSTM-CRF model. We believe that the reason this happened is that the start token (B-label) is referenced to the suffixes most of the time. An example of this, "الجامعه" with a label B-ORG becomes "ال", "جامعه"with labels B-ORG, I-ORG respectively, providing misleading starting cues to the model. Testing multilingual BERT, it proved inefficient as we got results lower than the baseline model.

How AraBERT works on classification task

Text classification involves assigning a text sample to one or more predefined categories. AraBERT v2 excels at this by leveraging its pre-trained knowledge of the Arabic language. Here's how it works:

- **Text Preprocessing:** The input text is first preprocessed, which might involve cleaning, tokenization (breaking the text into smaller units), and potentially applying other techniques.
- **Embedding Generation:** AraBERT v2 then takes the preprocessed text and generates numerical representations (embeddings) for each token. These embeddings capture the meaning and context of the words within the text.
- **Classification Layer:** Finally, a classification layer on top of AraBERT v2 analyzes the generated embeddings and predicts the most likely category for the text based on the training it received.

How AraBERT works on Named Entity Recognition task:

Named Entity Recognition (NER) is a subtask of Information Extraction (IE) that focuses on identifying and classifying specific types of entities mentioned within a text. These entities can be people, organizations, locations, dates, monetary values, percentages, etc.

AraBERT-v2 comes into play as a powerful tool specifically designed for tackling NER tasks involving Arabic text. It leverages the strengths of the BERT architecture and is pre-trained on a massive dataset of Arabic text. This pre-training allows AraBERT-v2 to understand the nuances of the Arabic language and identify named entities effectively.

Here's a breakdown of how AraBERT-v2 tackles NER:

1. **Text Preprocessing:** Like text classification, the input Arabic text undergoes preprocessing steps like cleaning and tokenization.
2. **Embedding Generation:** AraBERT-v2 then generates numerical representations (embeddings) for each token in the text. These embeddings capture the meaning and context of the words, crucial for recognizing entities.
3. **NER Layer:** On top of AraBERT-v2, a specific NER layer takes the generated embeddings and predicts the most likely entity type (e.g., person, location) for each token. This prediction is based on the context of the surrounding words and the pre-trained knowledge of AraBERT-v2.

Chapter 3

<Project planning>

Feasibility Study

A Feasibility study is used to determine if a business or a specific project is achievable, so for determining the achievability of our project we will go deeper in the following points: -

1.1.1 Market Analysis

- Our app exists in the market, especially in the Remote Access software market.

- We can make money from our Website using ADS or from locking some features for premium accounts to encourge subsrciption.

Competitive Edge of Desktop Anywhere: Desktop Anywhere offers a unique set of features that position it competitively in the remote desktop access market. In the following points we will illustrate them.

Addressing User Pain Points:

- **Untethered Access:** Unlike traditional remote desktop solutions, Desktop Anywhere eliminates the need for physical proximity to access personal computers. This empowers users with ultimate flexibility and convenience.

- **Streamlined File Management:** Desktop Anywhere tackles the inefficiency of file retrieval, especially with large datasets. Users can access and manage files across multiple desktops seamlessly, saving valuable time and effort.

- **Enhanced Accessibility:** Desktop Anywhere caters to users with disabilities by offering alternative input methods like virtual touchpads, keyboards, and voice commands (focusing on Arabic for wider user base). This promotes inclusivity and empowers a broader audience.

- **Improved Productivity and Organization:** Voice-activated scheduling and task management features built within Desktop Anywhere help users stay organized and avoid work overload. These fosters increased productivity and a more enjoyable work experience.

Technological Innovation:

- **Integrated Mobile App:** The seamless integration of a mobile application with desktop software sets Desktop Anywhere apart. This user-friendly approach allows for remote access and control directly from mobile devices, a ubiquitous technology in today's world.

- **Automatic Device Pairing:** The ability to automatically connect to paired devices without constant authentication simplifies the user experience and strengthens security measures.

Thus, Desktop Anywhere can establish itself as a frontrunner in the remote desktop access market, offering a solution that prioritizes convenience, efficiency, accessibility, and a user-centric approach.

1.1.2 Organizational Analysis:

- The number of members in our system: - 5 members.

- The coverage we need to generate and support our services:
  - We develop and upgrade the application by the 5 developers members.

1.1.3 Operational Analysis:

- In our project there are some rules to the members such as:
  - Commitment to the deadline for delivering tasks.
- We deliver our service by using the internet.

Estimated Cost

A cost estimate approximates the cost of a program, project or operation. The cost estimate is the product of the cost estimating process and our estimated cost for this project comes as following: -

Currenly, our cost is 0 LE. As we depends on online meetings, using free server, free limited resources to fine tune the models. But in case of project deployement to be used in the market, server subscription will be a must, also buying better tools to maintain and develop new features in the system is going to be a necessary step, but the cost of this step can’t be determinded now as the prices are not stable.

Gantt Chart

ID	Name	Start Date	End Date	Duration	Progress %	Dependency	Resources	Color
1	Desktop Anywhere	Sep 20, 2023	May 30, 2024	182 days	56			
2	Survey and Learning	Sep 20, 2023	Oct 18, 2023	21 days	80		5 Member	121
3	Requirments Specifications	Sep 20, 2023	Oct 18, 2023	21 days	50	2FS-21 days	5 Member	211
4	Project Analysis	Oct 10, 2023	Oct 24, 2023	11 days	20	3FS-7 days	5 Member	301
5	project Design	Oct 25, 2023	Nov 24, 2023	23 days	57			331
6	project Architecture	Oct 25, 2023	Nov 08, 2023	11 days	57		5 Member	
7	User Interface Design	Nov 17, 2023	Nov 24, 2023	6 days	57		5 Member	
8	Project Implementation	Oct 10, 2023	Dec 22, 2023	54 days	57			181
9	Mobile App	Oct 10, 2023	Nov 07, 2023	21 days	57		2 Member	
10	API Server & Desktop App	Oct 10, 2023	Nov 07, 2023	21 days	57		3 Member	
11	Testing (Mobile)	Nov 17, 2023	Nov 24, 2023	6 days	57		2 Member	
12	Voice Command	Nov 17, 2023	Dec 08, 2023	16 days	57		3 Member	
13	Voice Command (part2)	Nov 24, 2023	Dec 08, 2023	11 days	57		2 Member	
14	Testing (voice command)	Dec 08, 2023	Dec 15, 2023	6 days	57		2 Member	
15	Testing (Desktop & API)	Dec 08, 2023	Dec 15, 2023	6 days	57		3 Member	
16	Integration	Dec 15, 2023	Dec 22, 2023	6 days	57		5 Member	
17	Main Modules	Jan 26, 2024	May 21, 2024	83 days	57			271
18	Scheduled Tasks via Voice	Jan 26, 2024	Mar 01, 2024	26 days	57		2 Member	
19	Virtual Touchpad / Keyboard	Jan 26, 2024	Feb 23, 2024	21 days	57		3 Member	
20	Testing (Scheduled task)	Mar 01, 2024	Mar 08, 2024	6 days	57		2 Member	
21	Testing (Virtual)	Feb 23, 2024	Mar 01, 2024	6 days	57		3 Member	
22	Auto Connection	Mar 04, 2024	Mar 29, 2024	20 days	57		3 Member	
23	File & Folder Search	Mar 08, 2024	Mar 29, 2024	16 days	57		2 Member	
24	Testing (Auto connection & File-folder)	Apr 05, 2024	Apr 12, 2024	6 days	57		5 Member	
25	Multi Devices Sync	Apr 15, 2024	May 09, 2024	16 days	57		5 Member	
26	Final Testing	May 07, 2024	May 21, 2024	11 days	57		5 Member	
27	Project Documentation	Oct 25, 2023	May 30, 2024	157 days	57		5 Member	91

Table 3.1

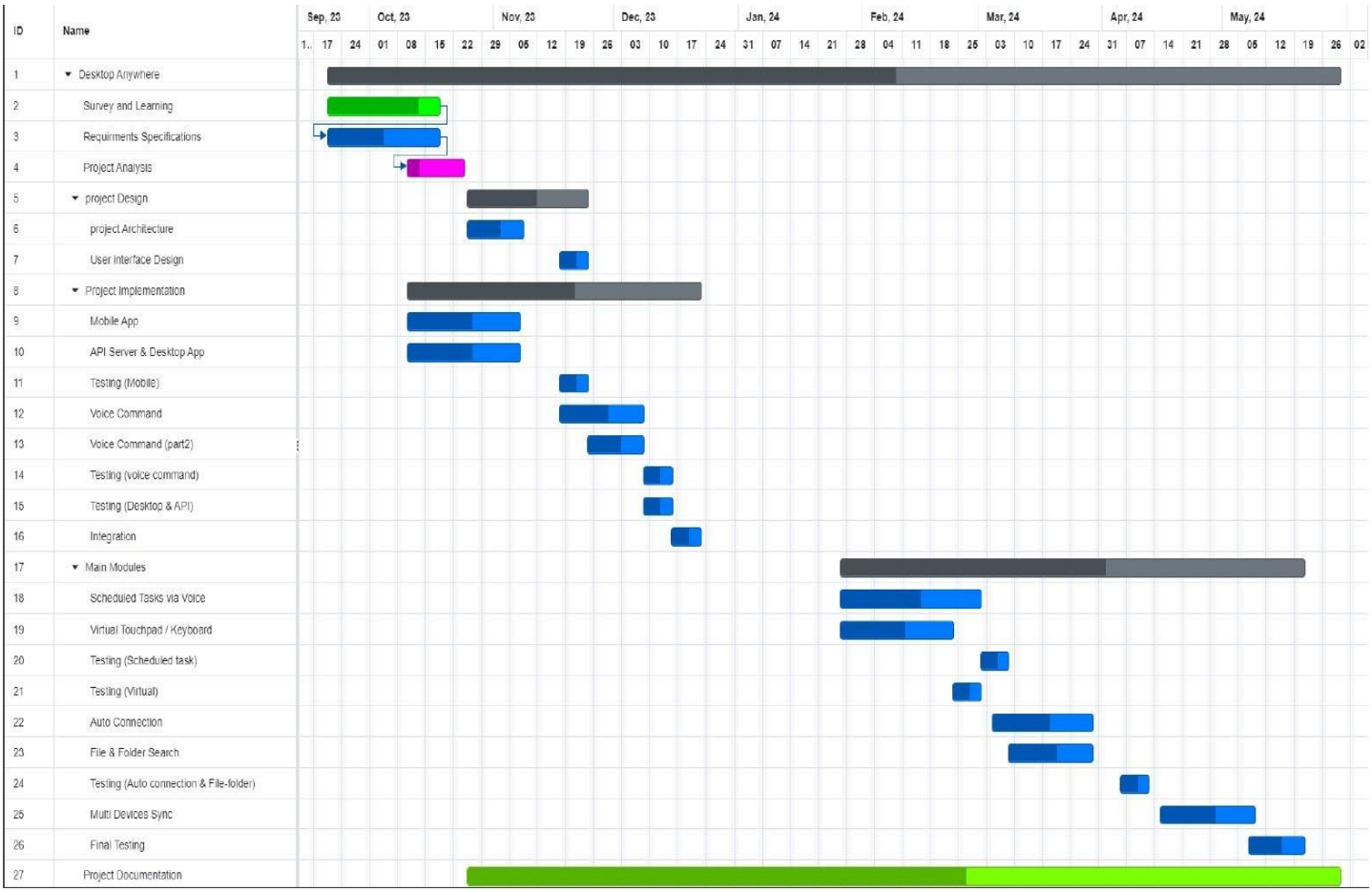


Figure 3.1

Chapter 4

<System Analysis>

Analysis and limitations for existing systems

App	Icon	Remote Control	Voice Command	Auto connection	Virtual Touchpad	Multiple Desktops
TeamViewer		✓	✗	✗	✗	✗
Any Desk		✓	✗	✗	✗	✓ Not Free
Chrome Remote Desktop		✓	✗	✗	✗	✗
Microsoft Remote Desktop		✓	✗	✗	✗	✓
Splashtop		✓	✗	✓	✓ In IOS only and not free.	✓
VNC Connect		✓	✗	✓	✓	✓

Table 4.1

TeamViewer

TeamViewer is a popular remote access and remote-control software that allows users to connect to and control remote computers and devices. It is designed for both personal and business use and offers a wide range of features to support remote work, collaboration, and support.

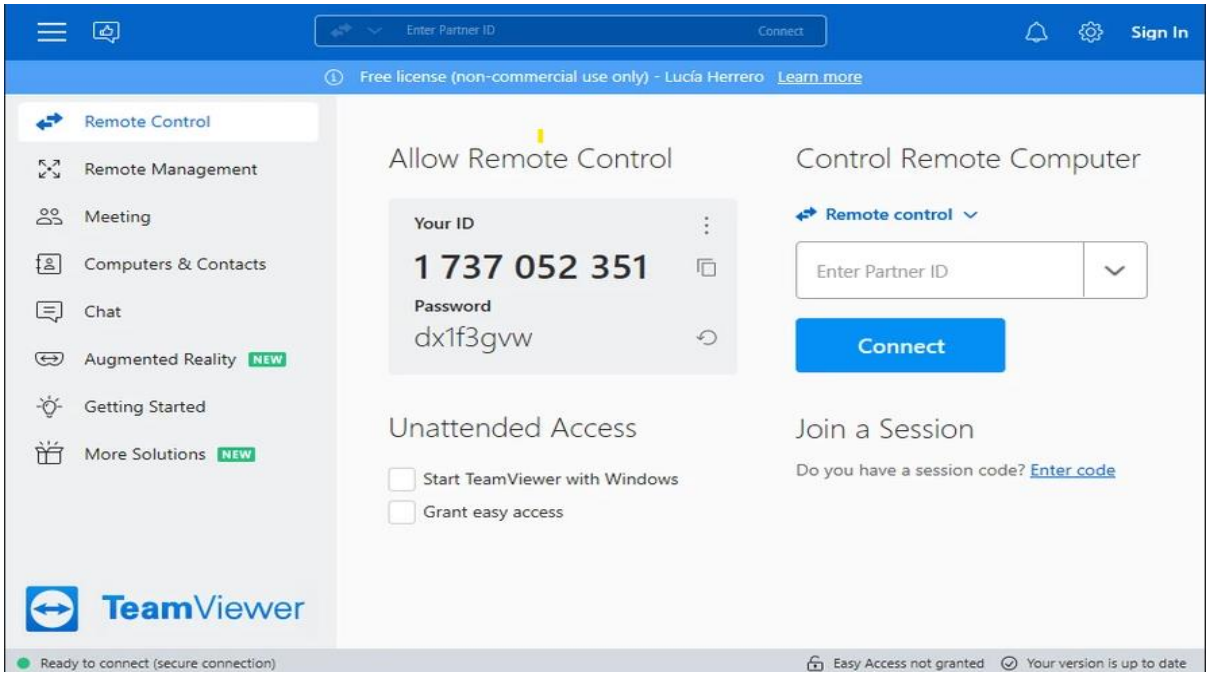


Figure 4.1

Any Desk

Any Desk is a fast and secure remote desktop application that allows you to access and control other computers from your own device. It offers features such as high-speed streaming, cross-platform compatibility, and secure connections. Any Desk is easy to set up and use and offers flexible license models and customization options.

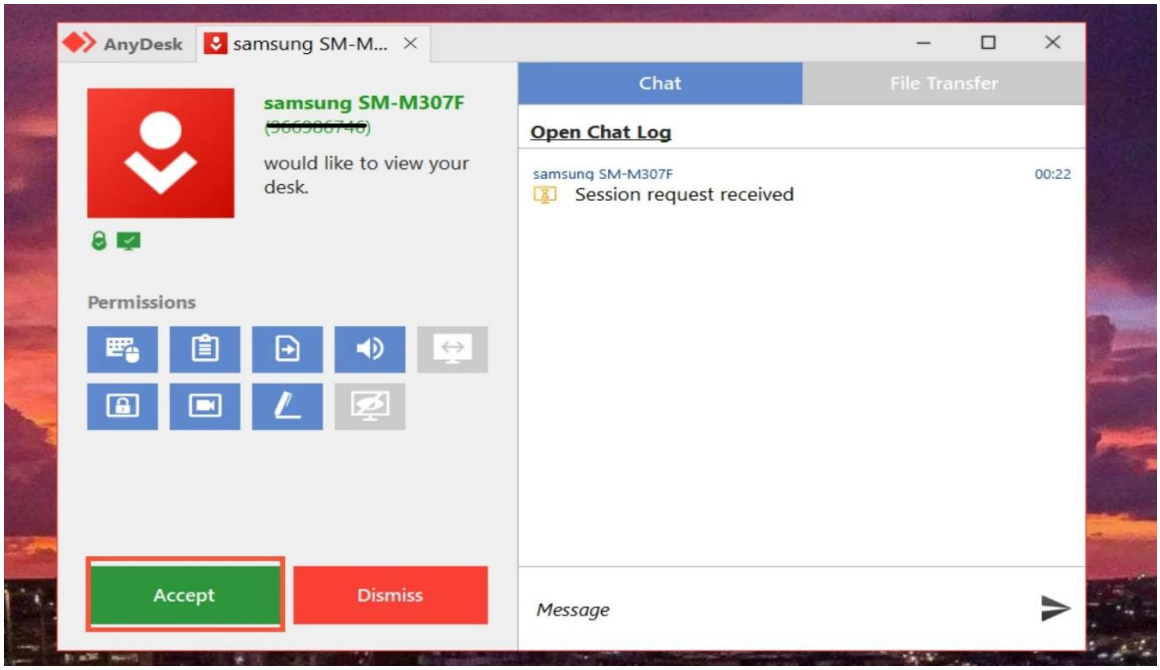


Figure 4.2

Chrome Remote Desktop

Chrome Remote Desktop is a remote desktop software tool developed by Google. It allows a user to remotely control another computer's desktop through a proprietary protocol called Chromoting. The tool consists of a server component for the host computer and a client component on the computer accessing the remote server. Chrome Remote Desktop uses a unique protocol, as opposed to using the common Remote Desktop Protocol (RDP) developed by Microsoft.

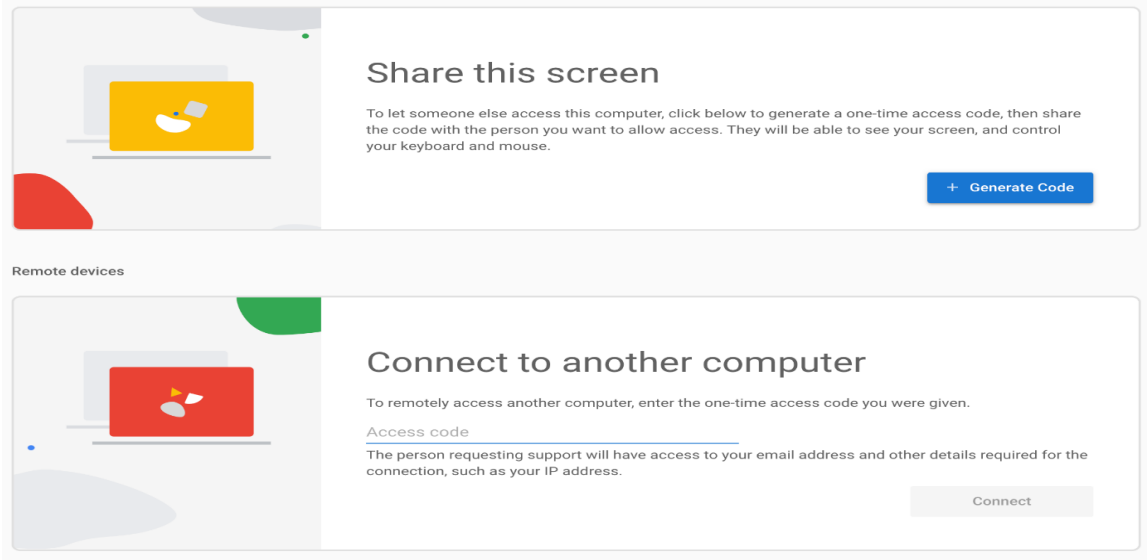


Figure 4.3

Microsoft Remote Desktop

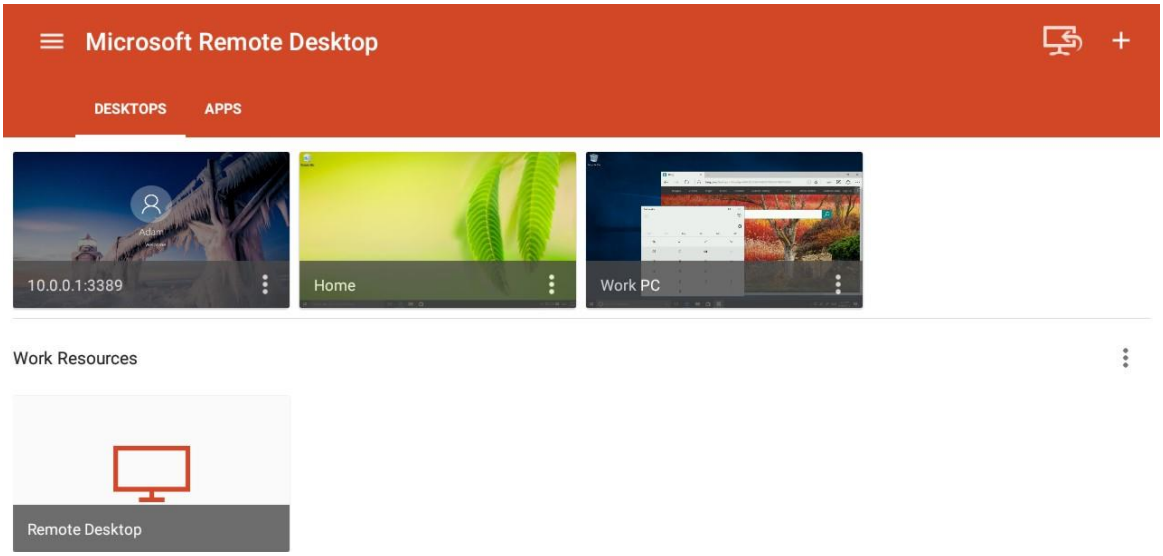


Figure 4.4

Microsoft Remote Desktop is a remote desktop application developed by Microsoft. It allows a user to remotely connect to and control another computer running the Remote Desktop Protocol (RDP). The application is available for Windows, macOS, iOS, and Android, and allows a user to connect to a remote computer and access its resources as if they were sitting in front of it.

Splashtop

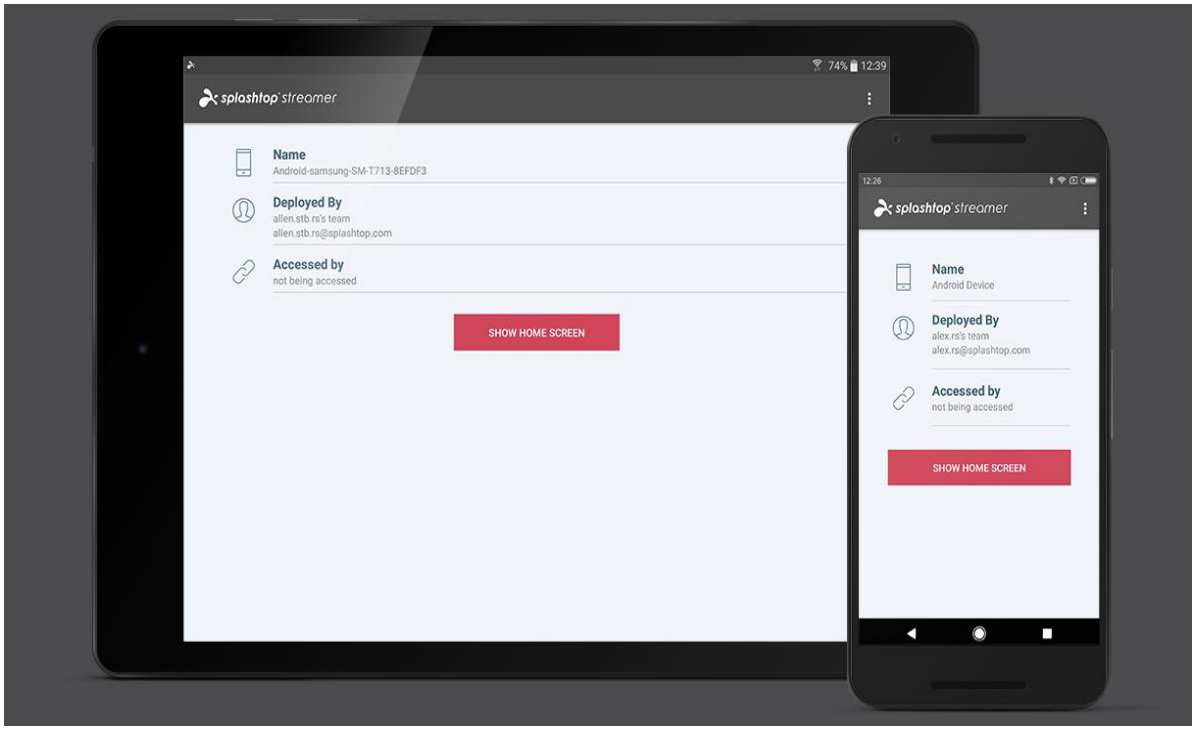


Figure 4.5

Splashtop is a remote access and remote support software that allows you to securely access and control other computers and devices from your own. It is available for Windows, macOS, iOS, and Android. Splashtop offers a few features, including the ability to connect to multiple remote desktops at once, use multiple monitors, and transfer files between the local and remote computers. It also supports remote audio and video streaming and allows a user to print remotely.

VNC Connect

VNC Connect is a remote access and remote desktop software developed by RealVNC. It allows a user to remotely connect to and control another computer or virtual desktop using the Virtual Network Computing (VNC) protocol. VNC Connect offers a few features, including the ability to connect to multiple remote desktops at once, use multiple monitors, and transfer files between the local and remote computers. It also supports remote audio and video streaming and allows a user to print remotely.

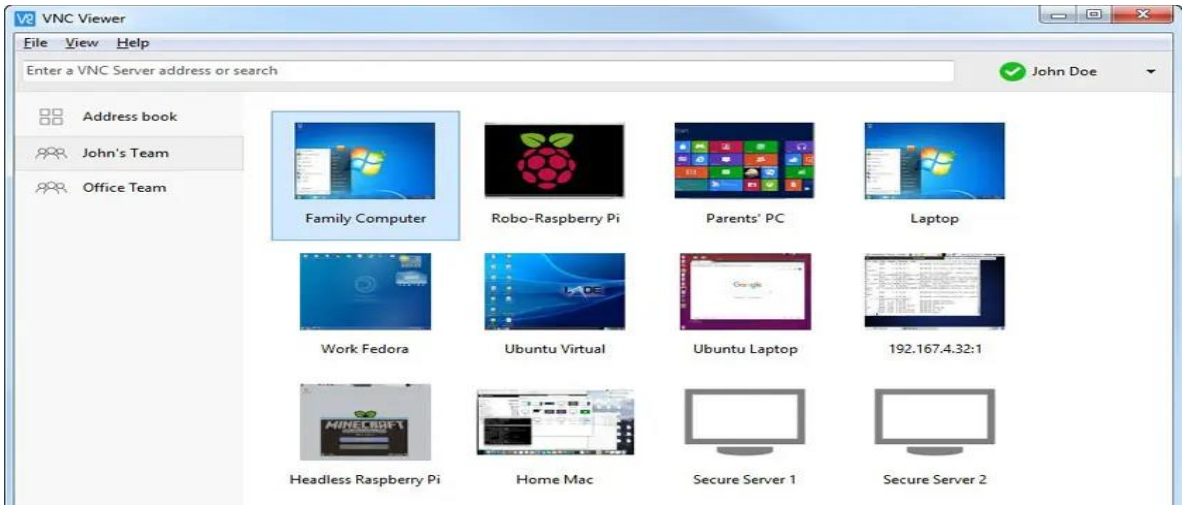


Figure 4.6

The Need for the new System

In today's fast-paced world, where time is of the essence and efficiency is paramount, the need for streamlined processes for accessing and managing desktop computers has never been greater. With professionals and individuals constantly on the move, the ability to seamlessly control multiple personal desktops remotely has evolved into a necessity rather than a luxury. Traditional methods of physically accessing desktops and navigating files and resources across multiple devices are not only time-consuming, but also inefficient. Furthermore, the need for accessibility is greater, particularly for those with motor impairments or disabilities, who face significant challenges when using traditional interfaces such as typing. By introducing "Desktop Anywhere," we hope to revolutionize remote desktop interactions by offering users a comprehensive solution that addresses these pressing needs.

We are not talking here about the limited accessibility only; Users face many other problems which will be discussed in more details in the following list of user challenges:

- File Management: Managing files and resources across multiple desktop computers can be complex and time-consuming, making it difficult for users to organize and retrieve essential documents efficiently.
- Frustrating File Retrieval: Navigating through vast amounts of data to find specific files or folders among multiple devices can be tedious and time-consuming, impacting productivity and workflow.
- Hands-Free Challenges: Interacting with the computer while engaged in other activities can be challenging without hands-free options, restricting users' ability to multitask effectively.
- Cumbersome Task Execution: Manually executing tasks like shutdown, restart, or application launch requires navigating through menus, which can be cumbersome and slow, impeding users' workflow.
- Overwhelming Multitasking: Simultaneously managing computer tasks and other real-world activities can be overwhelming without efficient task scheduling, leading to increased stress and reduced productivity.
- Inconvenient Device Integration: Users struggle with the inconvenience of having to authenticate frequently when connecting multiple devices, disrupting workflow and productivity.

In conclusion, Professional users and some traditional users face a vast challenge in their daily lives while using their desktop computers, these challenges must be addressed and solved and that’s why Desktop Anywhere is finally here.

Targeted users of the new system

In our project, the target users are individuals who rely on desktop computers as their primary computing devices for work, study, or personal use.

Also, individuals or professionals who require remote access to their desktop computers for various purposes, such as:

- Professionals who work remotely and need access to their desktop computers from different locations.
- Students who need to access their school or university computers from home or while on the go.

- Individuals who want to access their home computers from a mobile device while away from home.
- Users with disabilities, such as those with difficulties in typing or performing manual tasks.
- Users who struggle with the time-consuming process of navigating through files and folders, particularly when dealing with large data sets stored on their desktop computers.

Characteristics of Target Users:

Based on the project and the targeted user groups, the key knowledge, and characteristics that users should have to be able to easily interact with the Desktop Anywhere system are:

1. Technical Proficiency:
    - Basic understanding of desktop computer and mobile device operations.
    - Familiarity with file management and application usage on both desktop and mobile platforms.
    - Comfort with adopting and learning new technological solutions.
2. Digital Literacy:
    - Ability to navigate and utilize various digital interfaces, including mobile applications and remote desktop controls.
    - Willingness to learn and adapt to the unique features and functionalities of the Desktop Anywhere system.
3. Adaptability and Problem-Solving Skills:
    - Ability to quickly adapt to the remote interaction paradigm offered by Desktop Anywhere.
    - Willingness to troubleshoot and resolve any minor technical issues that may arise during the use of the system.
4. Motivation and Productivity Mindset:
    - Desire to improve their work-life balance and enhance their overall productivity.
    - Openness to explore and leverage new technologies to streamline their tasks and workflows.
5. Accessibility Awareness (for users with disabilities):
    - Understanding of their specific accessibility needs and requirements.

- Familiarity with assistive technologies and their integration with remote desktop solutions.

6. Language Proficiency (for voice command features):
    - Fluency in the primary language supported by the Desktop Anywhere voice command functionality, which is the Arabic language.

By ensuring that users possess these knowledge and characteristics, the Desktop Anywhere system can be more easily adopted and utilized, leading to a seamless and efficient remote interaction experience for the targeted user groups.

Analysis of the new system

1.1.4 User Requirements

Mandatory Requirements:

- Effortless Remote Desktop Access: Users need a seamless method to access and control their personal desktop computers remotely through the mobile application, enabling them to manage tasks from any location.
- Comprehensive File and Resource Management: Users require efficient file and resource management capabilities across multiple desktop computers to streamline organization and retrieval processes, enhancing productivity.
- Enhanced Accessibility Features: The system must offer accessibility features such as voice commands in Arabic to accommodate users with motor impairments or disabilities, ensuring inclusivity in desktop interactions.

Desirable Requirements:

- Convenient Multi-Device Sync: Users desire the ability to connect multiple devices from the same mobile for synchronized remote control, facilitating seamless multitasking and workflow management.
- Efficient Task Execution: Implementation of voice commands to execute tasks like shutdown, restart, or application launch after a specified duration, reducing the time and effort required for manual task execution.
- Automatic Device Connectivity: Users prefer the system to automatically connect to all paired devices without authentication, minimizing interruptions to workflow and enhancing overall user experience.

1.1.5 System Requirements:

Devices:

- Mobile Device: Smartphones or tablets.
- Computers: Windows PCs.

Operating system:

- Windows 7 and above.
- Android for mobile devices.

Internet Connection:

- A stable internet connection is required for accessing the remote desktop and transferring files.

Authentication Requirements:

- Users must have valid login credentials , IP and password, to access the remote desktop application.

Additional Requirements:

- Microphone and speaker if not already exists for voice command integration.

1.1.6 Domain Requirements:

Platform Compatibility:

- The system should be compatible with both mobile and desktop devices, supporting major operating systems such as Windows, Android.

Network Connectivity:

- Reliable internet connectivity is essential for remote desktop access and synchronization between devices.
- The system should support various network environments, including LAN, Wi-Fi, and mobile data connections.

Performance Optimization:

- Both the mobile and desktop applications should be optimized for performance, ensuring smooth remote desktop access and file management.

- Resource utilization should be efficient to minimize latency and maximize responsiveness during remote interactions.

Voice Recognition:

- The system should support voice recognition capabilities for executing commands and performing tasks using natural language input.
- Voice recognition functionality should be accurate and responsive.

User Interface:

- The user interface should be intuitive and user-friendly, providing easy navigation and seamless interaction across devices.
- Accessibility features should be incorporated to accommodate users with disabilities or special needs.

Data Storage and Backup:

- The system should include features for data storage and backup to ensure the safety and integrity of user files and resources.
- Backup mechanisms should be in place to prevent data loss in case of system failures or unexpected events.

Updates and Maintenance:

- Regular updates and maintenance should be provided to ensure compatibility with evolving technology standards and address security vulnerabilities.
- The system should include mechanisms for automatic updates and user notifications to keep users informed about new features and improvements.

Documentation and Support:

- Comprehensive documentation and user guides should be available to assist users in setting up and using the system effectively.
- Technical support channels should be established to address user inquiries, troubleshoot issues, and provide assistance as needed.

1.1.7 Functional Requirements:

1-Remote Desktop Access

Actor: User.

Preconditions:

- The user has installed the mobile application on their device.
- The user's personal desktop computer is connected to the internet.
- The user has the IP address and all authentication credentials necessary for the first time conncetion.

Description:

The user launches the mobile application on their device. Within the mobile application, the user navigates to the remote desktop access feature. The user enters the required data of the desktop computer they want to connect to, including the IP address and all authentication credentials necessary. The system verifies the entered data and establishes a secure connection between the mobile device and the specified desktop computer.

Postconditions:

- The user has successfully accessed and controlled their specified desktop computer remotely through the mobile application.
- The user can perform tasks and operations on the specified desktop computer as if they were physically present near the device.

2-Multi-Device Synchronization

Actor: User.

Preconditions:

- The user has installed the mobile application on multiple devices.
- The user has logged in to the mobile application using the same account on all devices.

Description:

The user launches the mobile application on each of their devices. Within the mobile application, the user navigates to the multi-device synchronization feature. The user selects the option to synchronize their devices. Each device operates independently, allowing the user to make changes specific to that device without affecting the others. Changes made on one device, such as cursor movements or file transfers, are confined to that device and do not affect the others. Synchronization ensures that each device provides its own unique remote-control experience tailored to the user's preferences.

Postconditions:

- The user's devices are synchronized for remote control, allowing them to seamlessly interact with their desktop computers across multiple devices simultaneously.

3-Virtual Touchpad and Keyboard

Actor: User.

Preconditions:

- The user has established a connection between the mobile application and their desktop computer.

Description:

Within the mobile application, the user accesses the virtual touchpad and keyboard feature. The virtual touchpad provides precise cursor control on the desktop screen, allowing the user to move the cursor and perform actions accurately. The virtual keyboard is displayed on the mobile device's screen, enabling the user to input text and commands remotely. Both the touchpad and keyboard should be responsive and intuitive, providing a seamless user experience.

Postconditions:

The user has successfully utilized the virtual touchpad and keyboard to interact with their desktop computer remotely from their mobile device.

4-Voice Command Integration

Actor: User

Preconditions:

- The user has installed the mobile application on their device.
- The user's device has a microphone and is connected to the internet.

Description:

The user activates the voice command feature within the mobile application. The user speaks voice commands in Arabic for various functions, such as searching for files or folders on the desktop and scheduling tasks. The system processes the voice input, recognizing only Arabic commands, and executes the corresponding actions on the desktop computer. Voice recognition is tailored to support Arabic language commands exclusively, ensuring compatibility with users' preferences.

Postconditions:

- The user has successfully utilized Arabic voice commands to perform functions on their desktop computer remotely from their mobile device.
- Voice recognition accurately interprets the user's Arabic commands, enhancing the user experience and efficiency.

5-Autoconnection

Actor: User.

Preconditions:

- The user has previously paired their mobile device with their desktop computer through the mobile application.
- Both the mobile device and the desktop computer are powered on and connected to the internet.

Description:

Upon launching the mobile application, the system automatically detects and connects to all paired desktop computers without requiring manual authentication steps from the user. The user is seamlessly granted access to

their desktop computers without the need for manual intervention or additional authentication procedures. The system maintains a secure connection between the mobile device and the desktop computer, ensuring data confidentiality and integrity during the autoconnection process.

Postconditions:

- The user has successfully accessed their paired desktop computers remotely without the need for manual authentication steps.

6-File and Folder Search

Actor: User.

Preconditions:

- The user has installed the mobile application on their device.
- The user's device has a microphone and is connected to the internet.
- Voice recognition features are enabled in the mobile application settings.
- The user's desktop computer is powered on and accessible remotely.

Description:

The user activates the file and folder search feature within the mobile application. Using voice recognition, the user verbally specifies the name of the file or folder he wants to search for on the desktop. The system processes the voice input, executing the search query in the specified partitions of the desktop computer. Relevant search results, matching the specified file or folder name, are retrieved, and displayed to the user on their mobile device. The user can view and access the search results, facilitating efficient navigation and retrieval of files and folders from the desktop.

Postconditions:

- The user has successfully utilized voice recognition to search for files or folders on their desktop computer remotely from their mobile device.
- The user can access the relevant search results and proceed with the desired actions, enhancing productivity and workflow efficiency.

7-Scheduled Tasks via Voice

Actor: User

Preconditions:

- The user has installed the mobile application on their device.
- The user's device has a microphone and is connected to the internet.
- The user's desktop computer is powered on and accessible remotely.

Description:

The user activates the scheduled tasks feature within the mobile application. Using voice commands, the user specifies the task they wish to schedule on the desktop, such as shutdown, sleep, restart, or closing specific running app. The user also specifies the desired duration or time frame after which the scheduled task should occur. The system processes the voice input, schedules the specified task on the desktop. The user receives confirmation that the task has been successfully scheduled and will be executed according to the specified parameters.

Postconditions:

- The user has successfully utilized voice commands to schedule tasks on their desktop computer remotely from their mobile device.
- The scheduled tasks are set to occur after the specified duration, enhancing user convenience and automation of desktop operations.

1.1.8 Non-functional Requirements:

1. **Performance:**
   Our remote desktop access application ensures swift response times for remote desktop access and file management tasks, delivering a seamless user experience. Response times remain consistent across various network conditions and device setups.

2. **Availability:**
   Our remote desktop access application is designed for continuous operation, ensuring that users can use it in any time and it always “uptime” is the amount of time that it is operational and available for use. This is specified because some systems are designed with expected downtime for activities like database upgrades and backups.

3. **Usability:**
   Our remote desktop access application boasts an intuitive user interface, requiring minimal training for users to navigate proficiently.

4. **Reliability:**
   Our remote desktop access application is built to maintain its performance over time. Unreliable software may fail frequently, but our application is designed to provide consistent and stable functionality. This is particularly important for remote access applications, where users rely on the system to maintain a connection to their desktop computers.

Our methodology

Our Product development cycle: Every product must start with an **idea**. At first, we had plenty of ideas for our Graduation project, but we selected the best of them, then we started to **research** deeper in the idea and read papers about what people did in the past in the same area. Then We decided to use agile methodology to **develop** and manage the progress of our Software Project. Each phase of the development had to be complete before the next phase could begin. The idea required that the requirements of a project must first be determined. We have completed all the project architecture and designs. The next step is followed by writing the code. The sequences continue in complete increments. And provide quick and unpredictable responses to the feedback we receive on our project, and to create opportunities to assess a project's direction during the development cycle. And we assess the project in regular meetings or sprints. Then we go to **the test** phase to examine the software system and check for the result, and to make sure that there are no changes that need to be made before it's launched.

So why did we choose Agile methodology to develop our software system?

- To deliver users satisfaction by delivering valuable software continuously.
- To accept changes of requirements no matter how early or late in the project.
- To deliver software that works within a shorter timescale.
- Information is best transferred between parties in face-to-face conversations and that appears in the scheduled meetings.
- Working software is the key measure of progress.
- As Self-organized team produce the best architecture, requirements, and design.

We revise in each meeting **what went right** in this sprint and **what went wrong** and **what wants to be changed** in the next sprint. And set **goals** for the next sprint.

Chapter 5

<System Design>

System Architecture

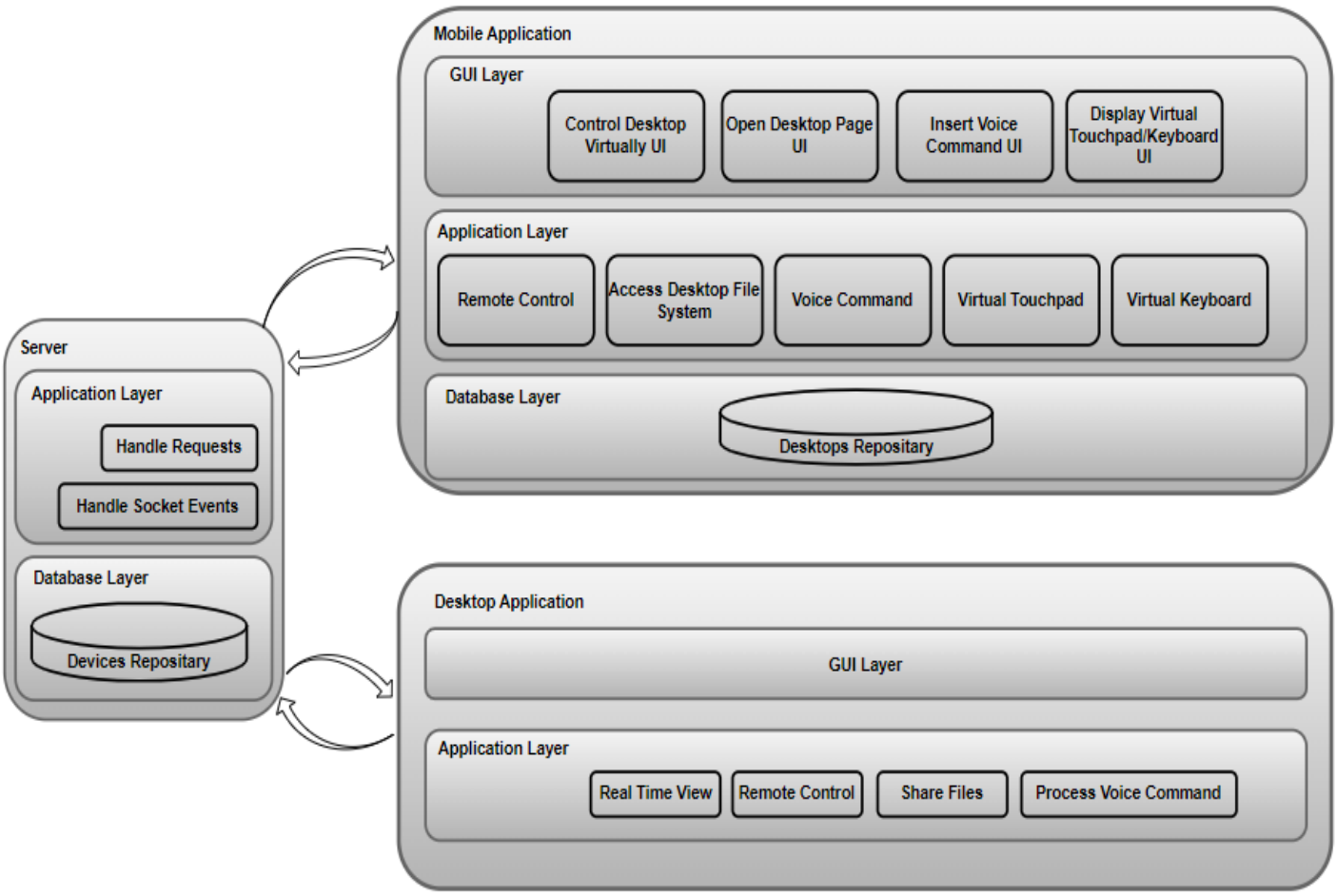


Figure 5.1- Desktop Anywhere System Architecture

Project Use case

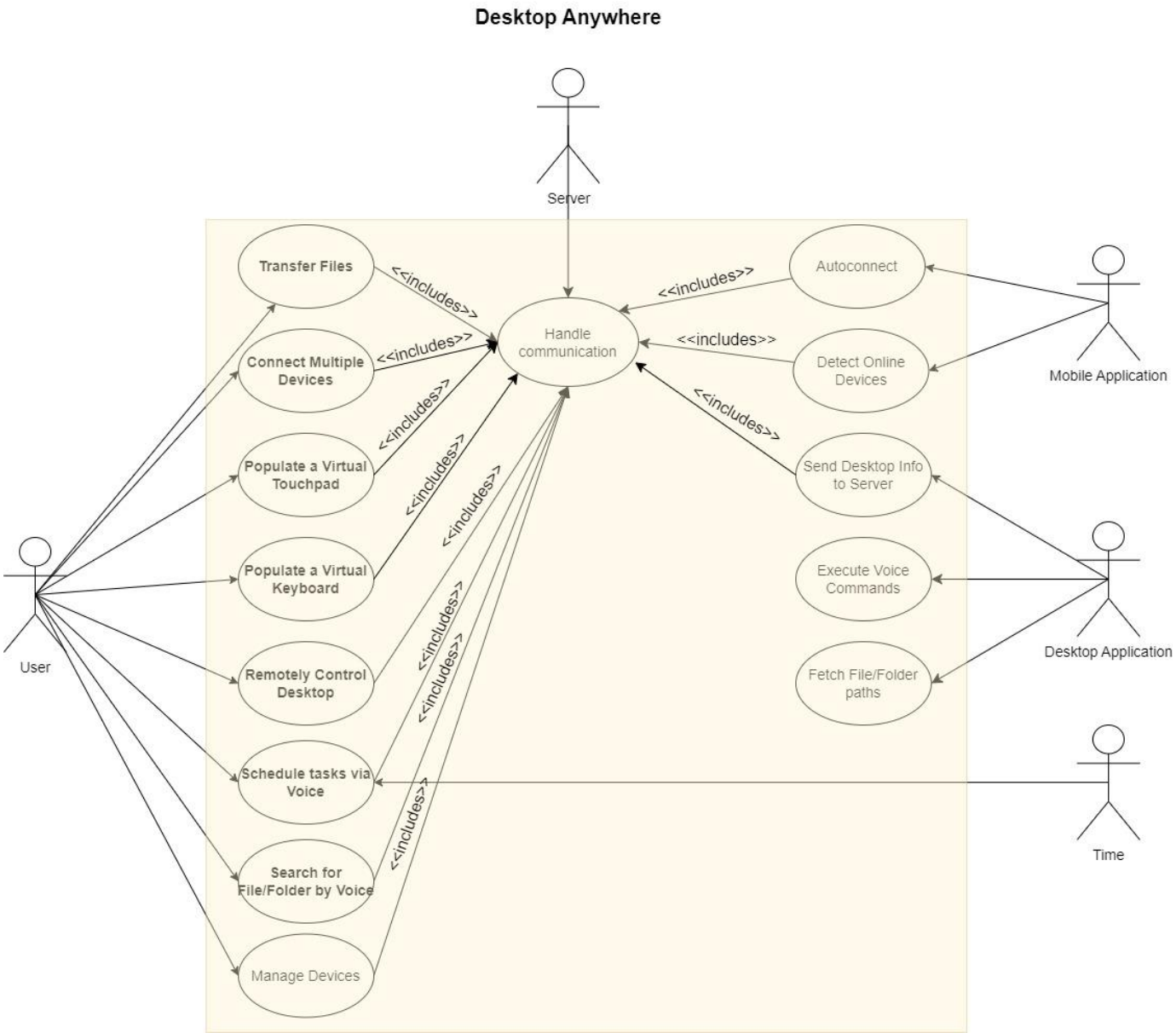


Figure 5.2- Desktop Anywhere Use Case

Whisper Architecture

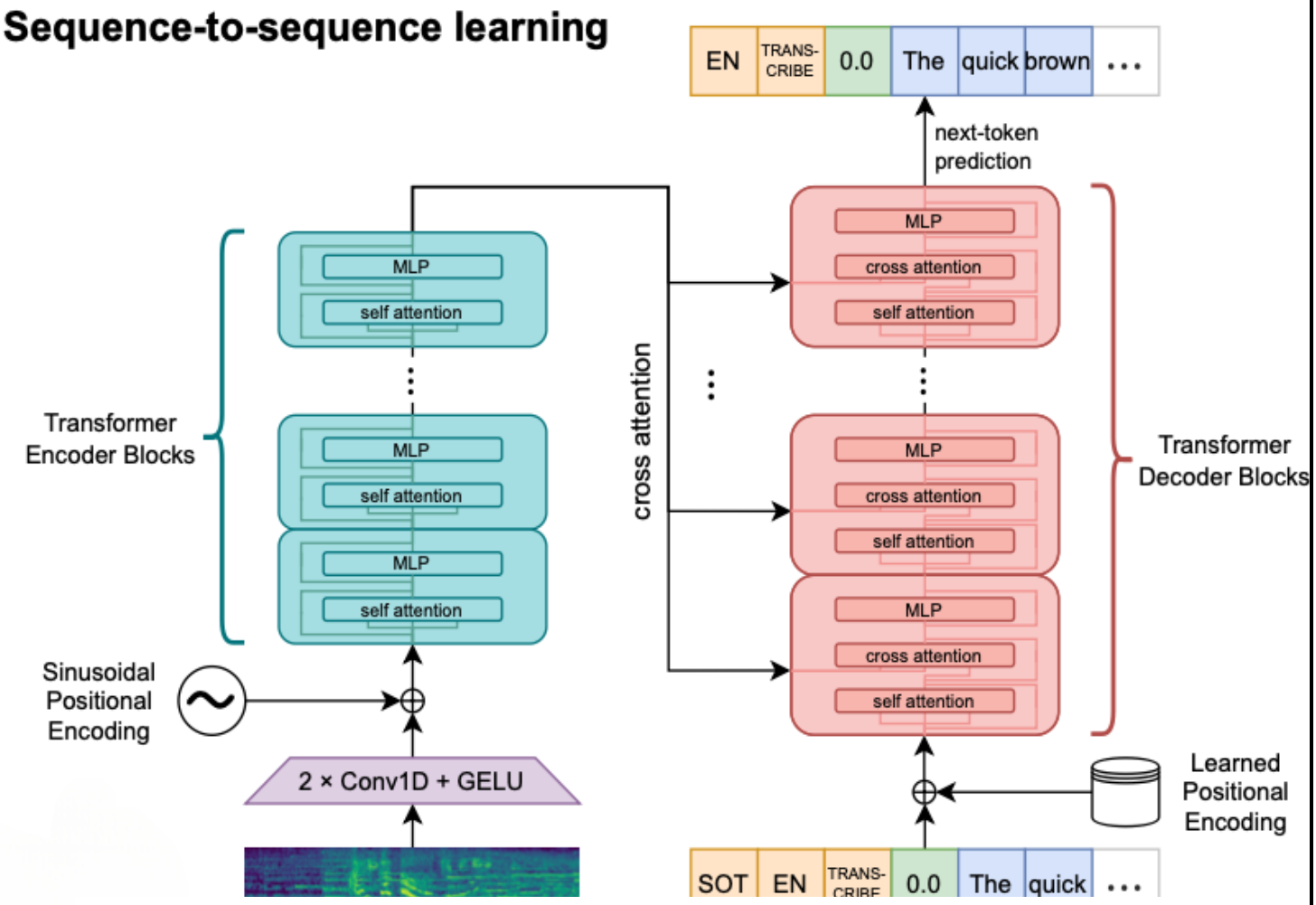


Figure 5.3- Whisper Architecture A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision," arXiv:2212.04356 [cs, eess], Dec. 2022, Available: https://arxiv.org/abs/2212.04356

Our Contribution in Whisper Fine-Tuning on Our Customized Dataset

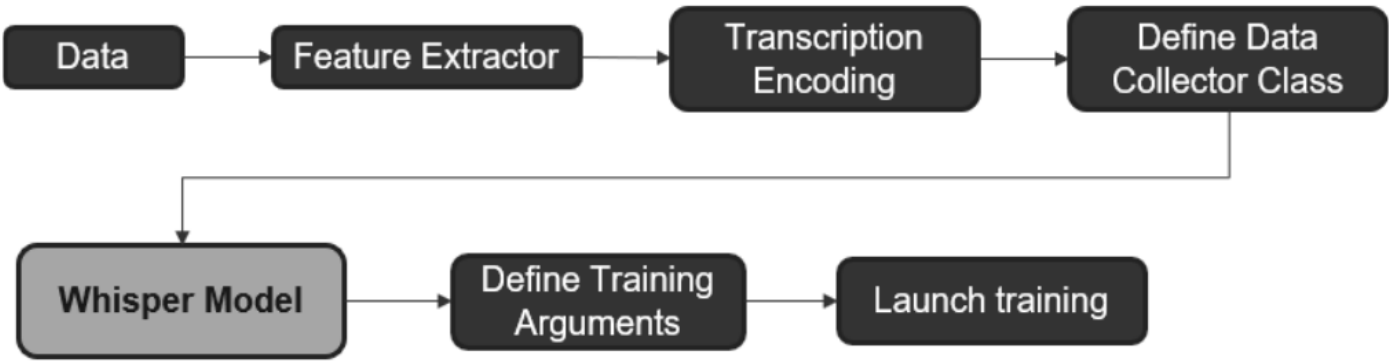


Figure 5.4- Whisper Fine-Tuning

AraBERT-V2 Architecture

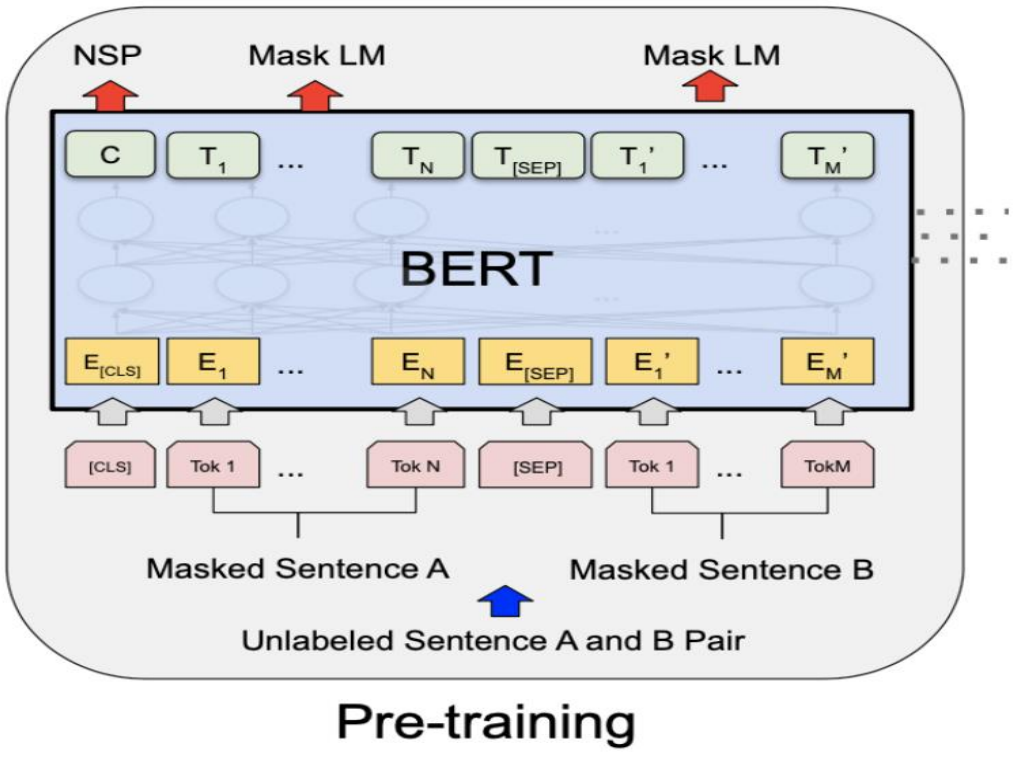


Figure 5.5-Arabert-V2 Training Architecture N. S. Chauhan, "Google BERT: Understanding the Architecture," The AI dream, Mar. 12, 2022. https://www.theaidream.com/post/google-bert-understanding-the-architecture

Multi-Class Classification AraBERT-V2

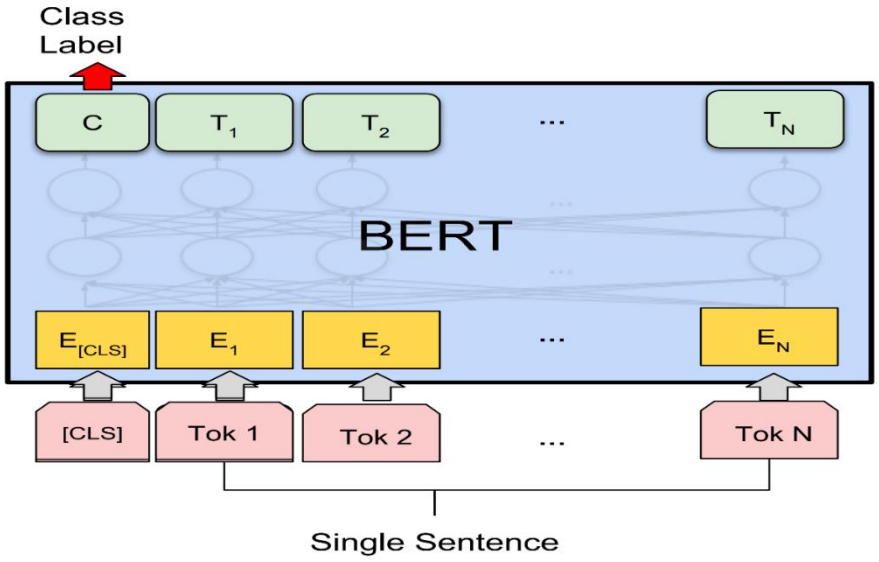


Figure 5.6-Arabert-V2 Multi-Class Classification D. Taunk, "How to fine-tune BERT on text classification task?," Analytics Vidhya, Jul. 28, 2023. https://medium.com/analytics-vidhya/how-to-fine-tune-bert-on-text-classification-task-723f82786f61

Named Entity Recognition AraBERT-V2

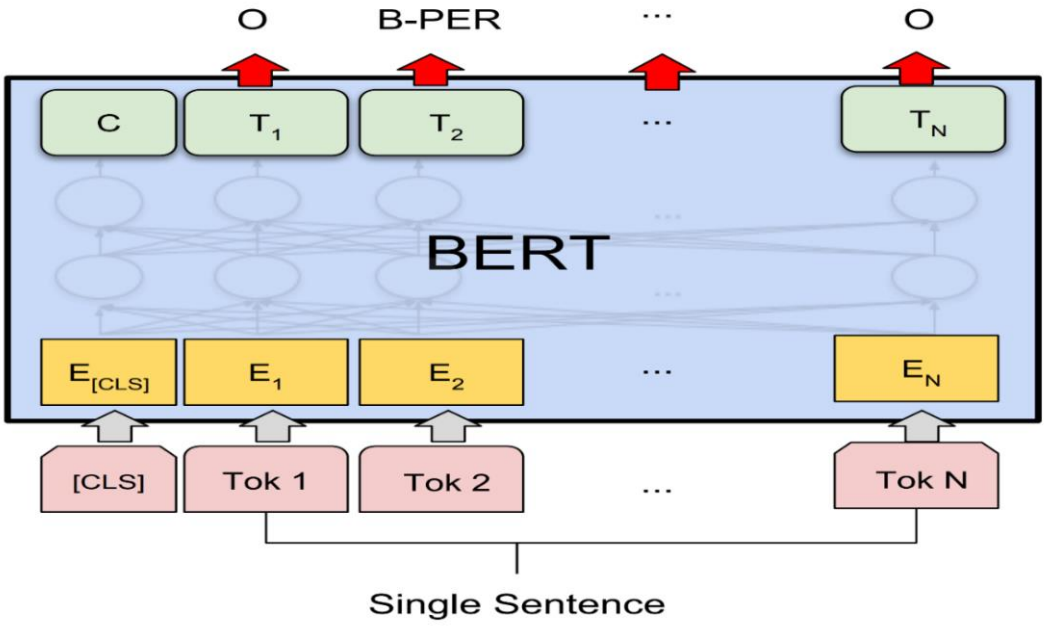


Figure 5.7-Arabert-V2 Named Entity Recognition walidamamou, "Master Named Entity Recognition with BERT in 2024," UBIAI, Dec. 28, 2023. https://ubiai.tools/mastering-named-entity-recognition-with-bert/

Our Contribution in Fine-Tuning AraBERT-V2

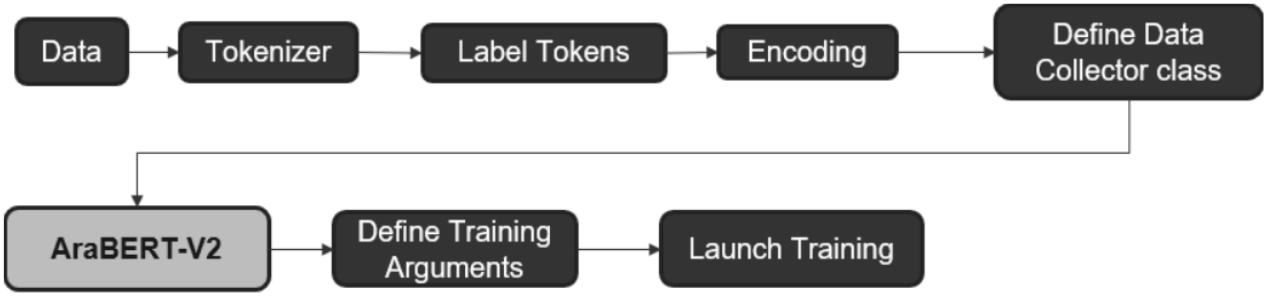


Figure 5.8-Arabert-V2 Fine-Tuning

Connection sequence diagram

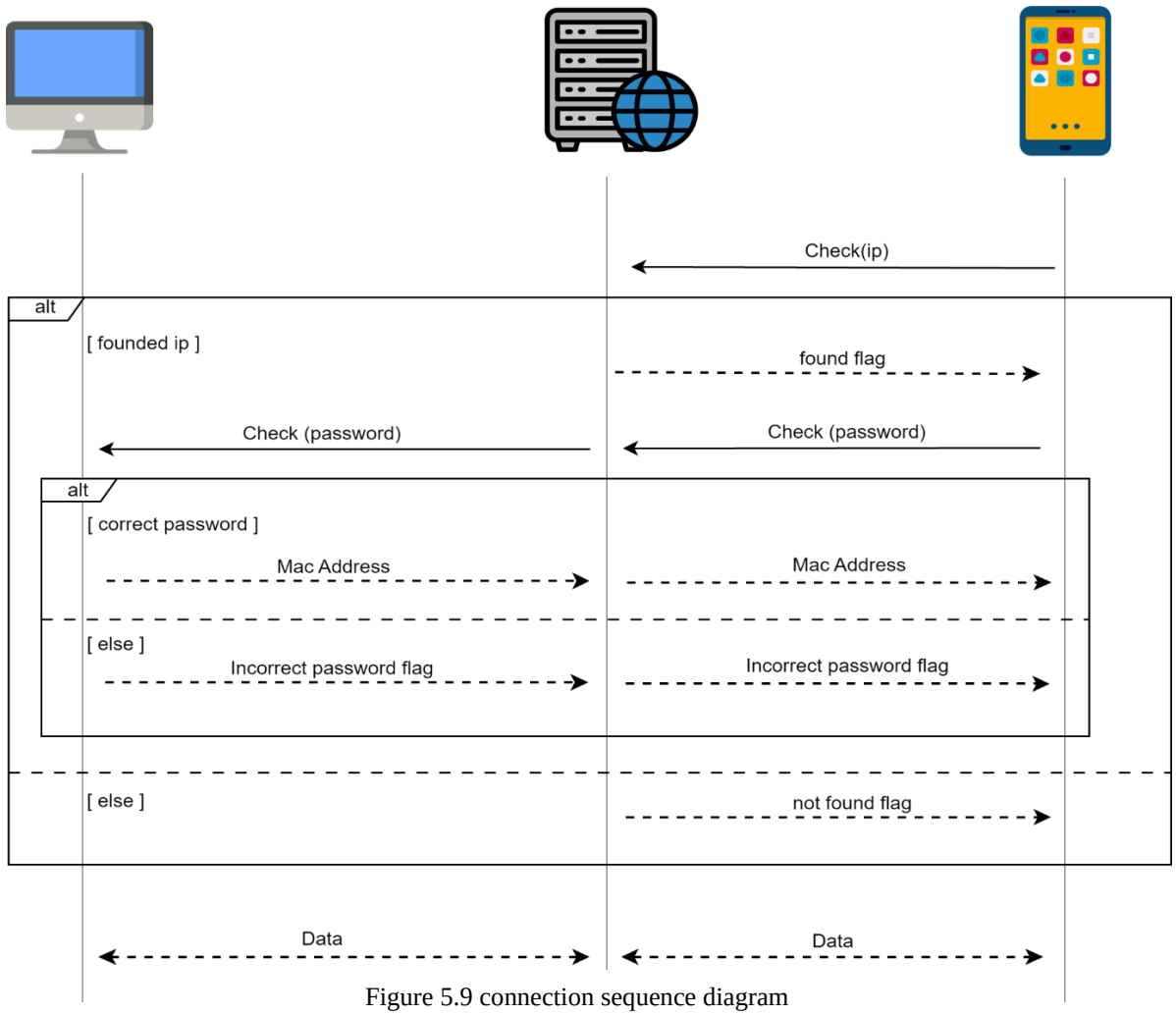


Figure 5.9 connection sequence diagram

Auto Connection sequence diagram

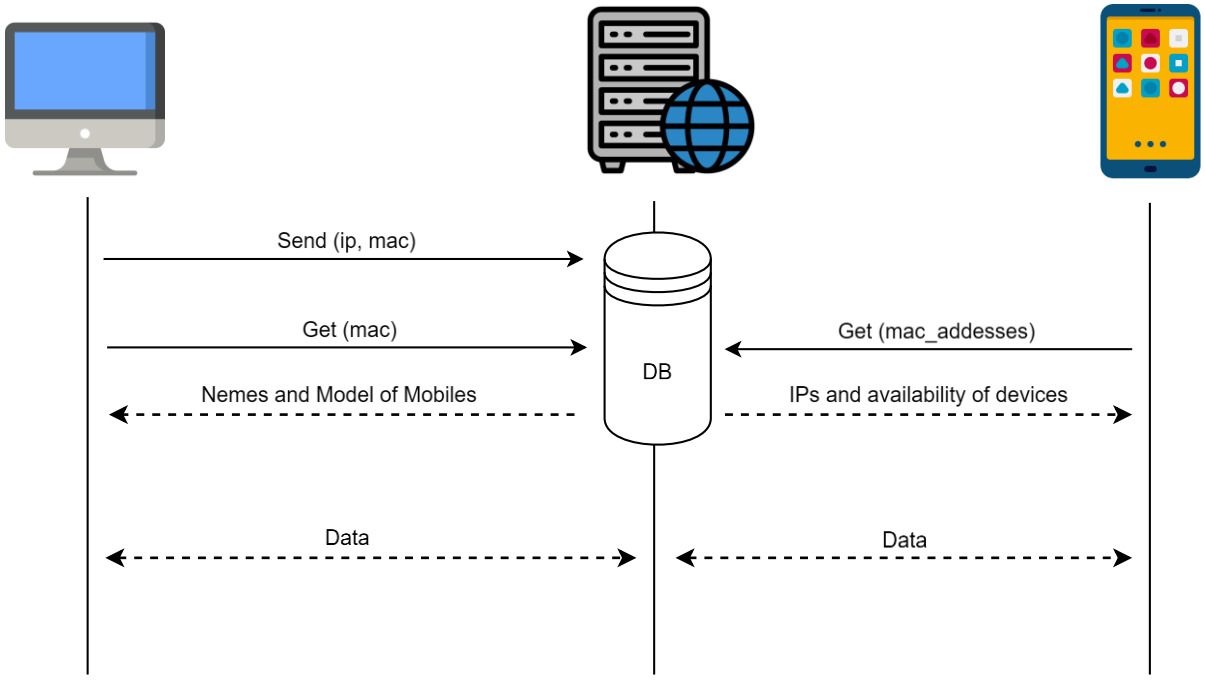


Figure 5.10 autoconnection sequence diagram

Real time view feature using WebRTC.

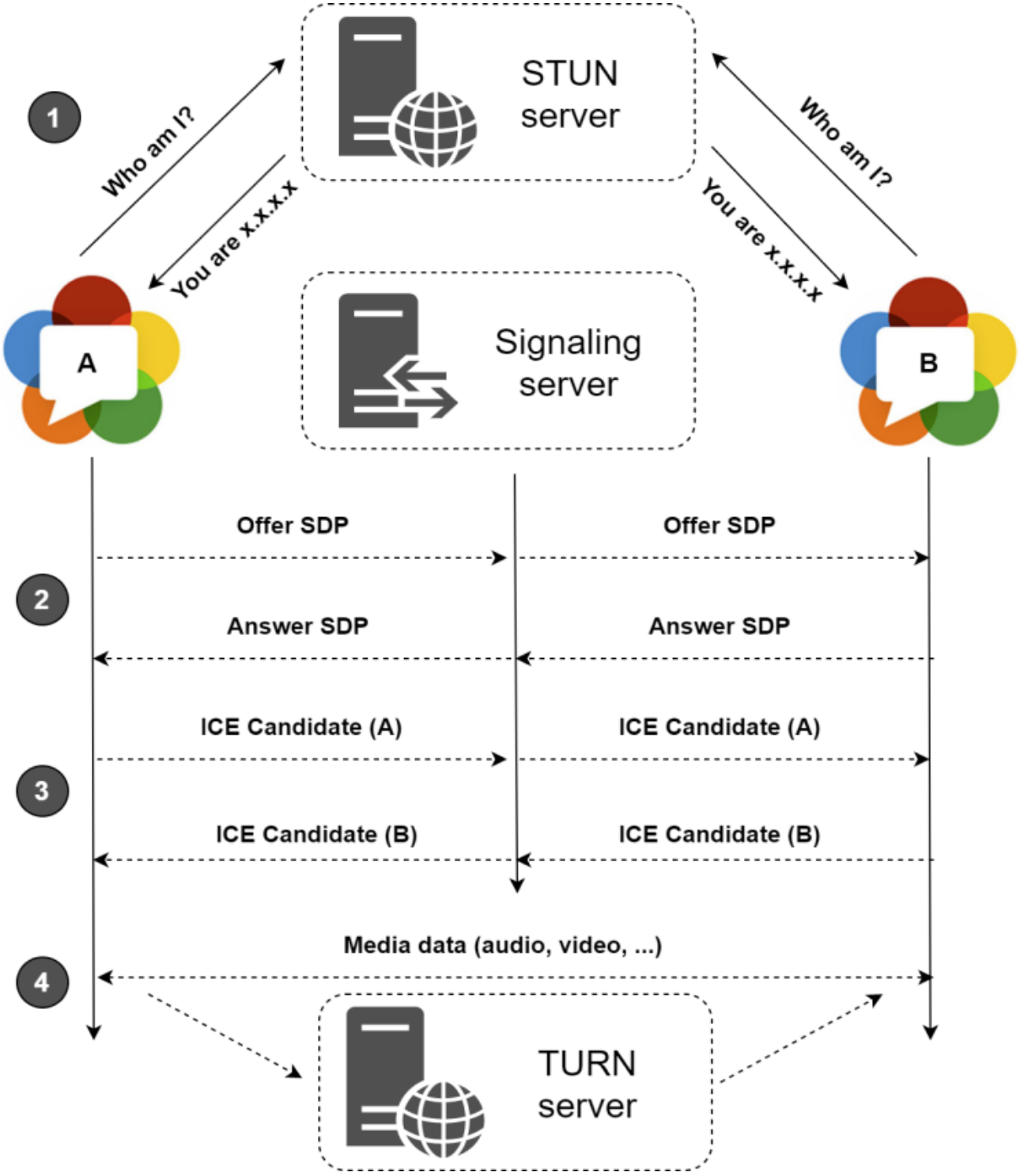


Figure 5.11 Realtime view [1] WebRTC, "WebRTC Home | WebRTC," *Webrtc.org*, 2017. https://webrtc.org/

Chapter 6 <Dataset>

The implementation step contains many sub-phases, and we are going to discuss the subphases that we have done till now.

These sub-phases:

1. Generating Dataset:
   - Whisper required dataset.
   - Arabert required dataset.

Before we go deeper in describing our dataset and its creation and evolution phases let's illustrate the Permitted Voice commands through our system. Users can:

- **Search:** The user can serach for a file/folder on his desktop.
- **Set :** The user can set the alarm or the stopwatch on his desktop device.
- **Add Note :** The user can add note on his desktop device.
- **Close :** The user can schedule shuting down the desktop or closing specific running app.
- **Restart :** The user can schedule restarting the desktop.
- **Sleep :** The user can schedule putting the desktop on sleep mode.

Dataset generation and evolution

In this section we will discuss the phases which the dataset went through, and the challenges we faces, our mistakes and the solutions.

Our initial task proved to be a challenge. We looked for a collection of data (dataset) containing this type of command on websites like Kaggle, but unfortunately, we couldn't find one. There wasn't a real-world situation (context) where this kind of command is commonly used, so we couldn't easily collect data from one. Since there wasn't any existing data available, we had to create our own dataset by generating the data ourselves.

Before we go for start creating the data we analyzed the needs of each model. Whisper needs: to fine tune whisper we want commands in form of voice records, Arabert needs: to fine tune Arabert we want commands in a text form.

Thus, we could start creating our dataset, we started with the text one then we used the sentences column to be recorded to get the data needed for Whisper. In our first trial we created the text dataset as following:

	Sentences	Class	Keyword	label
2	ابحث عن مجلد FCIS	1	FCIS	1
3	ابحث عن مجلد ImageProcessing	1	ImageProcessing	1
4	ابحث عن مجلد العمل	1	العمل	1
5	ابحث عن مجلد الرحلات	1	الرحلات	1
6	من فضلك ابحث عن مجلد album1	1	album1	1
7	من فضلك ابحث عن مجلد الصحافة	1	الصحافة	1
8	من فضلك ابحث عن مجلد العملي	1	العملي	1
9	من فضلك ابحث عن مجلد Math1	1	Math1	1
10	ابحث في الحاسوب عن مجلد Grades	1	Grades	1

Class
1-> search
2-> schedule
Label
1-> folder
2-> file

Table 6.1

In this format we followed the following structure:

- The first column is for sentences.
- The second column is for sentence class. 1 denotes to search class, 2 to schedule class, when we were creating this dataset we mistakly diveded the mentioned user permitted operations into two classes the search class and schedule class which contained all previous mentioned operations except search for file of folder.
- The third column contains one keyword which the sentence contains.
- The last column gives the keyword its label as we created in this phase only 800 row we had only 2 lables in this type of sentences, so we used in this coloumn 1 to refer to folder type and 2 to refer to file type.

We created 800 sentences following this structure, then the following problems arose:

- Multikeyword sentences are not handeled altough they are exsited in dataset. Actually this was the first problem faced us in generating our dataset, when we start creating the commands of setting the alarm we had 2 main keywords have to be labeled the first was the time to set the alarm and the second is whether its Am or Pm we tried to solve the problem without affecting the rest of the dataset as we chose to follow time as twenty-four hours it worked to solve the multikeyword problem in this sentence type but all of this was in vain as other sentences types still have multi-keywords like schecule closing a specific running app in a specific time as here we have to take the time and app name and they cannot be combined in one keyword in anyform.

- Dataset balance problem. Here was many subproblems existed. First of all, we were planning to create 800 sentence in a schedule class and 800 sentence in the search class. This seams to be balanced but going deeper in the details illustrated that we were wrong as the search class has two different types of commands Search for file/folder where the schedule class has 7 different commands under it, so if this will balance the classes it will not work in terms of learning the labels, the model will learn some

leables around 4 times the others, also we find that scheduling some of this commands isn't meaningul like schdeuling adding notes, why didn't write it at the same time you asked the device to write what makes you schedule writing it down? The same for setting the alarm or stopwatch why you will ask the computer to schdule setting the alarm at 12:00 after two hours? Thus, we found that we have to classify our dataset in a different way.

Let me introduce the data set after solving the mentioned problems and discuss the changes which have been made.

A	B	C
sentence	class	label
ابحث عن مجلد Image Processing	بحث	{"اسم العنصر":"Image Processing","نوع العنصر":"مجلد" }
ابحث في الحاسوب عن مجلد Grades	بحث	{"اسم العنصر":"Grades","نوع العنصر":"مجلد" }
جِد لي مجلد حسابات علمية	بحث	{"اسم العنصر":"حسابات علمية","نوع العنصر":"مجلد"}
في تمام العاشرة قم بغلق الجهاز	غلق	{"اسم العنصر":"الجهاز","وقت":"العاشرة" }
إنتظر حتي الساعة الثالثة عشر ثم أوقف تشغيل الجهاز	غلق	{"اسم العنصر":"الجهاز","وقت":"الساعة الثالثة عشر" }
بعد نصف ساعة أغلق الجهاز	غلق	{"اسم العنصر":"الجهاز","فترة":"نصف ساعة" }
بعد الساعة الثالثة عشر قم بإغلاق برنامج Blender	غلق	{"اسم العنصر":"Blender","وقت":"الساعة الثالثة عشر"}
أنهِ تطبيق الملاحة بعد إحدي عشر دقيقة	غلق	{"فترة":"إحدي عشر دقيقة", "اسم العنصر":"الملاحة"}
إنتظر ساعة ثم أعد تشغيل الجهاز	إعادة تشغيل الجهاز	{"اسم العنصر":"الجهاز","فترة":"ساعة" }
إنتظر خمسة عشر دقيقة ثم قم بإعادة تشغيل الجهاز	إعادة تشغيل الجهاز	{"اسم العنصر":"الجهاز","فترة":"خمسة عشر دقيقة" }
بمرور أربع ساعات وربع قم بإعادة تشغيل الجهاز	إعادة تشغيل الجهاز	{"اسم العنصر":"الجهاز","فترة":"أربع ساعات وربع" }
اضبط المنبه لينبهني بعد الساعة الرابعة والعشرين	ضبط	{"وقت":"الساعة الرابعة والعشرين","اسم العنصر":"المنبه"}
اضبط المنبه لينبهني بعد مرور ساعة كاملة	ضبط	{"فترة":"ساعة","اسم العنصر":"المنبه"}
اضبط المؤقت لينبهني بعد خمسة وثلاثين دقيقة	ضبط	{"فترة":"خمسة وثلاثين دقيقة","اسم العنصر":"المؤقت"}
اضبط المؤقت لينبهني بعد أربعين دقيقة	ضبط	{"فترة":"أربعين دقيقة","اسم العنصر":"المؤقت"}
قم بضبط المؤقت لمدة خمسون دقيقة	ضبط	{"فترة":"خمسون دقيقة","اسم العنصر":"المؤقت"}
قم بتعيين المؤقت لمدة خمسة وستون ساعة وسبعة عشر دقيقة	ضبط	{"فترة":"خمسة وستون ساعة و سبعة عشر دقيقة","اسم العنصر":"المؤقت"}
بمرور ثلث ساعة قم بوضع الجهاز في وضع السكون	إيقاف تشغيل الجهاز مؤقتاً	{"فترة":"ثلث ساعة"}
بمرور ست ساعات قم بوضع الجهاز في وضع السبات	إيقاف تشغيل الجهاز مؤقتاً	{"فترة":"ست ساعات"}
عِد ساعتين وربع ثم قم بإيقاف تشغيل الجهاز مؤقتاً	إيقاف تشغيل الجهاز مؤقتاً	{"فترة":"ساعتين وربع"}
اكتب في مذكراتي الآتي علينا قراءة النص بعناية قبل نشره	إضافة ملاحظة	{"الملاحظة":"علينا قراءة النص بعناية قبل نشره"}
أَضِف إلي ملاحظاتي لدي موعد عشاء في نفس يوم الاختبار	إضافة ملاحظة	{"الملاحظة":" لدي موعد عشاء في نفس يوم الاختبار"}
قم بإضافة الجملة التالية إلي مذكراتي كان الاجتماع مثمرًا وتم تغطية جميع النقاط المهمة	إضافة ملاحظة	{" الملاحظة":"كان الاجتماع مثمرًا وتم تغطية جميع النقاط المهمة"}

Table 6.2

First, we changed our structure, we followed the following described structure, the sentence is written in the first column and the second column represents the sentence class and the third contains list of keywords of this sentence each keyword will have its label in front of it. Thus, we handled multiple keywords sentences. What about the dataset balance and classes?

We chose to create 6 main classes which are displayed in the above figure, and they represent the main commands which are available to the user which we illustrated in the beginning, but why there are classes that contains 2 different command types under it? Because the increase of the number of classes equals decrease in the classifier accuracy. Thus, to reach the best values on your metrices you must minimize the number of classes to avoid overlapping classes problems, it's the problem where a classifier can't determine the sentence tends to be from which class as the classes are very similar and from different

perspectives you can say that this sentence can be from both classes, Thus, we grouped similar commands together under the same class.

To balance this dataset, we still have the problem which we mentioned before, we will mention it again but with an example from the current statues in the following figures.

{"فترة":"ساعة","اسم العنصر":"المنبه"}	ضبط	اضبط المنبه لينبهي بعد مرور ساعة كاملة
{"فترة":"35 دقيقة","اسم العنصر":"المؤقت"}	ضبط	اضبط المؤقت لينبهي بعد 35 دقيقة

Table 6.3

In this type of classes there are two different types of commands under the same class which means there are different values for our labels.

{"الملاحظة":" لدي موعد عشاء في نفس يوم الاختبار"}	اضافة ملاحظة	اضف الي ملاحظاتي لدي موعد عشاء في نفس يوم الاختبار

Table 6.4

In this type of classes there is only one type of commands under the same class which means there are no different values for labels in the same class.

And here is where the dataset balance problem arises, if we tried to give the classes the same number of sentences across the dataset, the labels would not be balanced and vice versa.

To solve this problem there were many options.

- As the NER task works on sentences and label column only, and Clssification task works on sentences and class column only we can separate the dataset on different sheets and balance the labels with the sentences and balance the classes with the sentences but we didn't prefer this solution as it wasn't the best from the structural prespective.
- Avoid the problem by dividing the classes contains 2 commands under it and rather than having search class we will have search for file class and search for folder class and so on on all classes, But as we mentioned before, the more the number of classes which we have the less the accuracy of the classifier becomes, Thus, it wan't a functional solution.
- Last solution and the used one is the Balance using Reasonable Range Approach.

Let's illustrate it in more details simply it states that giving the dataset equal numbers of sentences in each class is not easily done and not always achievable, Like our case here, Thus to balance the dataset try the following steps: take the average of number of sentences in your classes and determine a percentage in

which we are all as classes will be considered balanced together like 30%, put on which basis can you chose this percentage? It doesn't depend on whether its 40% or 10% as it doesn't mean that 10% is better than 40% it depends on the relation between the biggest class and the smallest one in your dataset is that a 10 times relation or 2-3 times relation? That's what determines if this percentage is suitable to be used to balance the classes or not. We used a 30% range where the largest class is 1.38:1 to the smallest class.

Is that enough to get a high accuracy from both models? Unfortunately, the answer is No! We noticed that the WER in Whisper model is still high. The reason was that whisper read converts the Arabic numbers from a record to a numeric form الثالثة عشر in the record will be converted to 13 not to الثالثة عشر as expected as when the model works on calculating the WER it will find that the result it generate compared to the input text which is recorded

	A	B	C
1	sentence	class	label
2	ابحث عن مجلد image processing	بحث	{"اسم العنصر":"image processing","نوع العنصر":"مجلد" }
3	ابحث في الحاسوب عن مجلد grades	بحث	{"اسم العنصر":"grades","نوع العنصر":"مجلد" }
4	جد لي مجلد حسابات علمية	بحث	{"اسم العنصر":"حسابات علمية","نوع العنصر":"مجلد"}
5	في تمام 10 قم بغلق الجهاز	غلق	{"اسم العنصر":"الجهاز","وقت":"10" }
6	انتظر حتي الساعة 13 ثم اوقف تشغيل الجهاز	غلق	{"اسم العنصر":"الجهاز","وقت":"الساعة 13" }
7	بعد نصف ساعة اغلق الجهاز	غلق	{"اسم العنصر":"الجهاز","فترة":"نصف ساعة" }
8	بعد الساعة 13 قم باغلاق برنامج blender	غلق	{"اسم العنصر":"blender","وقت":"الساعة 13"}
9	انتظر ساعة ثم اعد تشغيل الجهاز	اعادة تشغيل الجهاز	{"اسم العنصر":"الجهاز","فترة":"ساعة" }
10	انتظر 15 دقيقة ثم قم باعادة تشغيل الجهاز	اعادة تشغيل الجهاز	{"اسم العنصر":"الجهاز","فترة":"15 دقيقة" }
11	بمرور 4 ساعات قم باعادة تشغيل الجهاز	اعادة تشغيل الجهاز	{"اسم العنصر":"الجهاز","فترة":"4 ساعات" }
12	اضبط المنبه لينبهي بعد الساعة 24	ضبط	{"وقت":"الساعة 24","اسم العنصر":"المنبه"}
13	اضبط المنبه لينبهي بعد مرور ساعة كاملة	ضبط	{"فترة":"ساعة","اسم العنصر":"المنبه"}
14	اضبط المؤقت لينبهي بعد 35 دقيقة	ضبط	{"فترة":"35 دقيقة","اسم العنصر":"المؤقت"}
15	اضبط المؤقت لينبهي بعد 40 دقيقة	ضبط	{"فترة":"40 دقيقة","اسم العنصر":"المؤقت"}
16	قم بضبط المؤقت لمدة 50 دقيقة	ضبط	{"فترة":"50 دقيقة","اسم العنصر":"المؤقت"}
17	قم بتعيين المؤقت لمدة 65 ساعة و17 دقيقة	ضبط	{"فترة":"65 ساعة و17 دقيقة","اسم العنصر":"المؤقت"}
18	بمرور ثلث ساعة قم بوضع الجهاز في وضع السكون	ايقاف تشغيل الجهاز مؤقتاً	{"فترة":"ثلث ساعة"}
19	بمرور 6 ساعات قم بوضع الجهاز في وضع السبات	ايقاف تشغيل الجهاز مؤقتاً	{"فترة":"6 ساعات"}
20	عد ساعتين وربع ثم قم بايقاف تشغيل الجهاز مؤقتاً	ايقاف تشغيل الجهاز مؤقتاً	{"فترة":"ساعتين وربع"}
21	اكتب في مذكراتي الاتي علينا قراءة النص بعناية قبل نشره	اضافة ملاحظة	{"الملاحظة":"علينا قراءة النص بعناية قبل نشره"}
22	اضف الي ملاحظاتي لدي موعد عشاء في نفس يوم الاختبار	اضافة ملاحظة	{"الملاحظة":" لدي موعد عشاء في نفس يوم الاختبار"}
23	قم باضافة الجملة التالية الي مذكراتي كان الاجتماع مثمر وتم تغطية جميع النقاط المهمة	اضافة ملاحظة	{" الملاحظة":"كان الاجتماع مثمر وتم تغطية جميع النقاط المهمة"}

Table 6.5

When we reached this final version of our written dataset, we used it to create whisper required dataset. We used the sentences written here to be recorded and used to fine tune whisper with the resulted records. For better accuracy we followed the following criteria in creating the records, on average each 20 sentence is recorded by a unique different person to have a variety of pronunciation to avoid any errors when the product is deployed and used by real users.

Description of final version of Dataset

1932

Number of created **records**

1872

Number of created **rows**

6

Number of created **Classes**

3225

Number of existed **entities**

Entities distribution

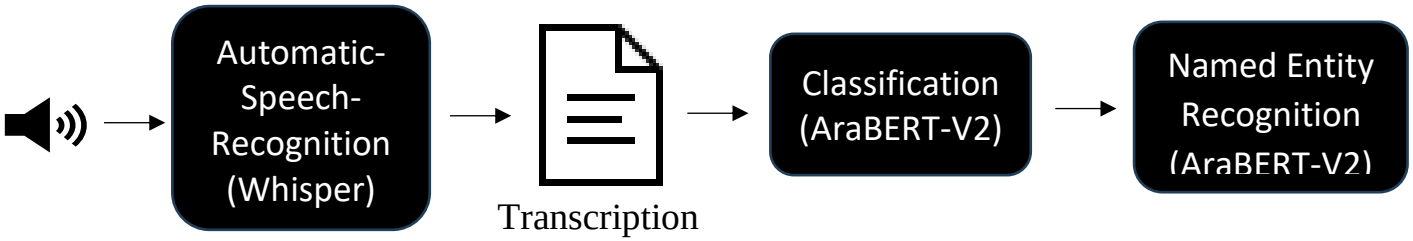
Distributed among 5 entities categories:
Time: 11.45%
Period: 27.34%
Note: 8.1%
Element's Name: 41.85%
Element's type: 11.25%

Chapter 7 <System Implementation>

After we have already illustrated our dataset, we now want to discuss the implementation of the following parts of our system:

2. Models Coding.
3. Mobile App Implementation.
4. Desktop App Implementation.
5. Server Implementation.
6. Integration.

We will start with models coding but before we go through how we implement the models here is an overview about the pipeline:



- **Receive Voice Command:**
  - We start by taking a voice command input from the user.
- **Transcribe Voice Command:**
  - The voice command is then passed to the Whisper model, which generates a transcription of the spoken words.
- **Classify Transcription**:
  - The transcribed text is sent to the classification model, AraBERT v2. This model determines the intent of the command and classifies it into one of the following categories: Search, Set, Add Note, Close, Restart, or Sleep.
- **Extract Important Information:**
  - the classified transcription is passed to a Named Entity Recognition (NER) model, also based on AraBERT v2. This model extracts important information relevant to the classified command. For example, if the command is classified as "Add Note," the NER model identifies and extracts the content of the note.
- This process involves the integration of voice recognition, text classification, and information extraction to efficiently handle and respond to user commands.

Whisper Model:

Model Description:

Whisper is a specific Automatic Speech Recognition (ASR) model. It is renowned for its high accuracy and its capability to handle various audio qualities, including those recorded in noisy environments. Whisper is particularly useful for transcribing audio into text with great precision.

Here's a simplified explanation of how audio-to-text technology, like Whisper, works:

- Audio Input:
  - The process begins with the audio file that you want to transcribe.
- Speech Recognition:
  - The audio data is then fed into the Whisper model. Whisper analyzes the audio, recognizing the speech patterns and phonemes (the basic units of sound) within the recording.
- Text Output:
  - Based on its analysis, the model generates a text transcript corresponding to the spoken words in the audio file.

Whisper is a multilingual model. For our use case, we want to handle sentences that contain a mix of Arabic and English. Therefore, each voice input fed into the model is a blend of both Arabic and English languages.

Model Implementation

1. We start by importing various libraries and modules required for the code to function, including dataclasses, torch, transformers, datasets, nltk, pydub, librosa, and others. These are essential for handling data, processing audio, and performing speech recognition tasks.

```
from dataclasses import dataclass
from typing import Any, Dict, List, Union
import torch
from huggingface_hub import HfApi, HfFolder
from transformers import WhisperFeatureExtractor, WhisperTokenizer, WhisperProcessor, WhisperForConditionalGeneration, Seq2SeqTrainingArguments, Seq2SeqTrainer
from datasets import load_dataset, DatasetDict, Audio, Dataset, ClassLabel
from pydub import AudioSegment
import os
import io
import pandas as pd
import evaluate
import re
import nltk
from nltk.tokenize import word_tokenize
import numpy as np
import torch
from transformers import AutoModelForSpeechSeq2Seq, AutoProcessor, pipeline
import csv
import pandas as pd
import jiwer
import re
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import librosa
```

Figure 7.1

2. Drive Mount and Model Initialization
    a. Mount Google Drive to access saved models and datasets. Initialize the Whisper model and pipeline for automatic speech recognition. Load the Whisper model, tokenizer, and feature extractor
3. Set up the ASR pipeline with the Whisper model for processing audio data.

```
pipe = pipeline(
    "automatic-speech-recognition",
    model=model,
    tokenizer=tokenizer,
    feature_extractor=feature_extractor,
    max_new_tokens=128,
    chunk_length_s=30,
    batch_size=16,
    return_timestamps=True,
    torch_dtype=torch_dtype,
    device=device,
)
```

Figure 7.2

4. Load audio files, process them with the Whisper pipeline, and save the recognized text to a CSV file.
5. Split the data into train and test
6. Preprocess the sentences by removing vowels and converting them to lowercase.

```
def tokenize_and_lowercase(sentence):
    # Tokenize the sentence into words
    words = word_tokenize(sentence)
    # Iterate through each word
    for i in range(len(words)):
        word = words[i]
        # Check if the word contains English alphabets using regex
        if re.search('[a-zA-Z]', word):
            # Convert the word to lowercase
            words[i] = word.lower()
    # Return the modified sentence
    return ' '.join(words)
```

Figure 7.3

```
def remove_vowels(sentence):
    # Define a regular expression pattern to match Arabic vowels
    arabic_vowels_pattern = "[\u064B-\u0652\u0640]"  # Range of Arabic vowels

    # Use the re.sub() function to replace vowels with an empty string
    sentence_without_vowels = re.sub(arabic_vowels_pattern, '', sentence)

    return sentence_without_vowels
```

Figure 7.4

7. Add period at the end of each sentence

```python
def add_period(sentence):
    # Check if the sentence already ends with a period
    if sentence.strip()[-1] != '.':
        return sentence.strip() + '.'
    else:
        return sentence.strip()
```

Figure 7.5

8. Convert the train and test datasets to Dataset Object
9. Set up feature extraction, tokenization, and dataset preparation for Whisper.

```python
# - Load Feature extractor
feature_extractor = WhisperFeatureExtractor.from_pretrained("openai/whisper-medium")

# - Load Tokenizer : WhisperTokenizer
tokenizer = WhisperTokenizer.from_pretrained("openai/whisper-medium", language="Arabic", task="transcribe")
```

Figure 7.6

10. Define a custom data collator for padding sequences during training.
    a. Extracts and pads the input audio features.
    b. Extracts and pads the label sequences.
    c. Replaces padding tokens in the labels with -100 to ensure they are ignored during loss computation.
    d. Removes the BOS token from the beginning of each label sequence if it was added during tokenization.
    e. Combines the padded input features and labels into a single batch dictionary.
    f. Returns the batch dictionary, ready for model input

```python
@dataclass
class DataCollatorSpeechSeq2SeqWithPadding:
    processor: Any

    def __call__(self, features: List[Dict[str, Union[List[int], torch.Tensor]]]) -> Dict[str, torch.Tensor]:
        # split inputs and labels since they have to be of different lengths and need different padding methods
        # first treat the audio inputs by simply returning torch tensors
        input_features = [{"input_features": feature["input_features"]} for feature in features]
        batch = self.processor.feature_extractor.pad(input_features, return_tensors="pt")

        # get the tokenized label sequences
        label_features = [{"input_ids": feature["labels"]} for feature in features]
        # pad the labels to max length
        labels_batch = self.processor.tokenizer.pad(label_features, return_tensors="pt")

        # replace padding with -100 to ignore loss correctly
        labels = labels_batch["input_ids"].masked_fill(labels_batch.attention_mask.ne(1), -100)

        # if bos token is appended in previous tokenization step,
        # cut bos token here as it's append later anyways
        if (labels[:, 0] == self.processor.tokenizer.bos_token_id).all().cpu().item():
            labels = labels[:, 1:]

        batch["labels"] = labels

        return batch
```

Figure 7.7

11. Define a function to compute Word Error Rate (WER) for evaluation
    a. Argument: The function takes one argument pred, which is an object containing the model's predictions and the true labels.
    b. pred_ids: This variable holds the predicted token IDs generated by the model.
    c. label_ids: This variable holds the true token IDs (ground truth labels) from the dataset.
    d. In the tokenization process, -100 is often used as a special token to indicate positions that should be ignored in the loss calculation (e.g., padding positions). However, when decoding tokens back into strings, these should be replaced with the actual padding token ID used by the tokenizer.
    e. tokenizer.batch_decode: This method converts a batch of token IDs back into strings.
    f. skip_special_tokens=True: This argument ensures that special tokens (like padding, start-of-sequence, end-of-sequence) are omitted from the decoded strings, focusing only on the actual content
    g. metric.compute: This method from the evaluate library computes the WER. It takes two arguments:
        i. predictions: The decoded prediction strings.
        ii. references: The decoded ground truth strings

h. Calculation: WER is typically a value between 0 and 1, representing the fraction of words that were incorrectly predicted. Multiplying by 100 converts this fraction into a percentage.
i. Output: A dictionary with a single key-value pair where the key is "wer" and the value is the computed WER percentage.

```python
def compute_metrics(pred):
    pred_ids = pred.predictions
    label_ids = pred.label_ids
    label_ids[label_ids == -100] = tokenizer.pad_token_id
    pred_str = tokenizer.batch_decode(pred_ids, skip_special_tokens=True)
    label_str = tokenizer.batch_decode(label_ids, skip_special_tokens=True)
    wer = 100 * metric.compute(predictions=pred_str, references=label_str)
    return {"wer": wer}
```

Figure 7.8

12.Load Whisper model

```python
# load a pretrained checkpoint
model = WhisperForConditionalGeneration.from_pretrained("openai/whisper-medium")

model.generation_config.language = "arabic"
model.generation_config.task = "transcribe"
model.generation_config.forced_decoder_ids = None
```

Figure 7.9

13.Define the Training arguments of the model

```python
training_args = Seq2SeqTrainingArguments(
    output_dir="./whisper-small-ar-en",
    per_device_train_batch_size=2,
    gradient_accumulation_steps=6,
    learning_rate=1e-5,
    warmup_steps=25,
    max_steps=200,
    gradient_checkpointing=True,
    fp16=False,
    evaluation_strategy="steps",
    per_device_eval_batch_size=2,
    predict_with_generate=True,
    generation_max_length=100,
    save_steps=50,
    eval_steps=50,
    logging_steps=50,
    report_to=["tensorboard"],
    load_best_model_at_end=True,
    metric_for_best_model="wer",
    greater_is_better=False,
    push_to_hub=False,
)

trainer = Seq2SeqTrainer(
    args=training_args,
    model=model,
    train_dataset=common_voice["train"],
    eval_dataset=common_voice["test"],
    data_collator=data_collator,
    compute_metrics=compute_metrics,
    tokenizer=processor.feature_extractor,
)


print("Training Started.")
trainer.train()
print("Training Ended.")
```

Figure 7.10

14.Launch the training

AraBERT-v2 Model for classification:

Model Implementation

1. We start by importing the necessary libraries like pandas, sklearn, transformers, datasets, torch.

```
import torch,os
import pandas as pd
from sklearn import preprocessing
import matplotlib.pyplot as plt
from transformers import pipeline, BertForSequenceClassification, BertTokenizerFast
from datasets import Dataset
import torch
```

Figure 7.11

2. Load the dataset
3. Label Encoding: Map labels to integers

```
NUM_LABELS = len(unique_values)
id2label = {idx: label for idx, label in enumerate(unique_values)}
label2id = {label: idx for idx, label in enumerate(unique_values)}
```

Figure 7.12

4. Tokenize Data: Load the tokenizer and model for the Arabic BERT.

```
tokenizer = BertTokenizerFast.from_pretrained("aubmindlab/bert-base-arabertv2", max_length=512)
```

Figure 7.13

5. Split Data: Split the data into training and testing sets
6. Encode Data: Tokenize and encode the training and testing texts.

```
train_encodings = tokenizer(train_texts, truncation=True , padding = True)
test_encodings  = tokenizer(test_texts, truncation=True , padding = True)
```

Figure 7.14

7. Create Dataset Class: Define a custom PyTorch dataset class to handle the encoded data and labels
    a. This CustomDataset class provides a standardized way to work with PyTorch datasets, making it easier to handle the data during training and evaluation.

```python
class CustomDataset(Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
        item['labels'] = torch.tensor(self.labels[idx])
        return item
```

Figure 7.15

8. Define Metrics: Define a function to compute accuracy, precision, recall, and F1 score.
    a. The compute_metrics function takes model predictions as input.
    b. It extracts true labels and predicted labels from the input.
    c. Using scikit-learn functions, it computes precision, recall, F1 score, and accuracy.
    d. The function returns these metrics as a dictionary.

```python
def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='macro')
    acc = accuracy_score(labels, preds)
    return {
        'Accuracy': acc,
        'F1': f1,
        'Precision': precision,
        'Recall': recall
    }
```

Figure 7.16

9. Perform grid search in order to find the best values for hyperparameters
    a. Define parameter grid

```python
parameter_grid = {
    'learning_rate': [5e-5, 3e-5, 2e-5],
    'per_device_train_batch_size': [16, 32],
    'num_train_epochs': [3, 4, 5],
    'weight_decay': [0.01, 0.1, 0.001]
}

best_score = float('-inf')
best_params = None
best_trainer = None
```

Figure 7.17

b. Perform manual grid search

```
for lr in parameter_grid['learning_rate']:
    for batch_size in parameter_grid['per_device_train_batch_size']:
        for epoch in parameter_grid['num_train_epochs']:
            for wd in parameter_grid['weight_decay']:
                print(f"Training with learning_rate={lr}, batch_size={batch_size}, num_train_epochs={epoch}, weight_decay={wd}")

                training_args = TrainingArguments(
                    output_dir='./TTC4900Model',
                    do_train=True,
                    do_eval=True,
                    num_train_epochs=epoch,
                    per_device_train_batch_size=batch_size,
                    per_device_eval_batch_size=batch_size,
                    warmup_steps=100,
                    weight_decay=wd,
                    logging_strategy='steps',
                    logging_dir='./multi-class-logs',
                    logging_steps=50,
                    evaluation_strategy="steps",
                    eval_steps=50,
                    save_strategy="steps",
                    fp16=True,
                    load_best_model_at_end=True,
                    learning_rate=lr
                )

                trainer = Trainer(
                    model=model,
                    args=training_args,
                    train_dataset=train_dataset,
                    eval_dataset=test_dataset,
                    compute_metrics=compute_metrics
                )

                trainer.train()
                eval_results = trainer.evaluate()
                score = eval_results['eval_Accuracy']

                if score > best_score:
                    best_score = score
                    best_params = {'learning_rate': lr, 'batch_size': batch_size, 'num_train_epochs': epoch, 'weight_decay': wd}
                    best_trainer = trainer
```

Figure 7.18

10. Launch the training
11. Evaluate the model

```
# Evaluate the model
train_pred = best_trainer.predict(train_dataset)
test_pred = best_trainer.predict(test_dataset)

# Extract predicted labels
train_preds = train_pred.predictions.argmax(-1)
test_preds = test_pred.predictions.argmax(-1)
```

Figure 7.19

12. Generate the classification report

```
print("Classification Report - Test Data:")
print(classification_report(test_labels, test_preds))

# Evaluate the model
train_pred = best_trainer.predict(train_dataset)
test_pred = best_trainer.predict(test_dataset)

# Extract predicted labels
train_preds = train_pred.predictions.argmax(-1)
test_preds = test_pred.predictions.argmax(-1)
```

Figure 7.20

AraBERT-v2 Model for Named Entity Recognition:

Model Implementation:

1. We start by importing libraries

```
from sklearn.model_selection import train_test_split
import datasets
import pandas as pd
import torch
from transformers import pipeline
from sklearn.preprocessing import LabelEncoder
from datasets import DatasetDict, Dataset
import numpy as np
from transformers import BertTokenizerFast
from transformers import DataCollatorForTokenClassification
from transformers import AutoModelForTokenClassification
from transformers import Trainer, TrainingArguments
from sklearn.metrics import accuracy_score
```

Figure 7.21

2. Reading and Preprocessing Data
3. Define a function to process the Data Frame into a format suitable for Named Entity Recognition (NER).

This is a sample of our customized dataset

{"اسم العنصر":"الجهاز","فتره":"ساعتين" } إنتظر ساعتين ثم أعد تشغيل الجهاز

Figure 7.22

Not every word has a label, as it should be in normal NER task so the following function loops over all dataset and set label for each word, if the word doesn't have label in dataset so we set it to 'O' stands for others as it doesn't matter for us.

```python
def process_data(df):
    processed_data = []
    for sentence_index, row in df.iterrows():
      sentence = row[0].strip()
      words = sentence.split()
      words_list = []
      labels_list = []
      label_dict = eval(row[1]) if pd.notna(row[1]) else {}
      right_dic = {}
      for label_word, label_value in label_dict.items():
        if label_word in sentence and label_value not in sentence:
          right_dic[label_value] = label_word

        elif label_value in sentence:
          right_dic[label_word] = label_value

      for word_index, word in enumerate(words):
          label = 'O'

          for label_word, label_value in right_dic.items():
            if word in label_value:
                label = label_word
                break
          words_list.append(word)
          labels_list.append(label.strip())
      processed_data.append({
          "sentence_id": sentence_index,
          "words": words_list,
          "labels": labels_list
      })
    return pd.DataFrame(processed_data)
```

Figure 7.23

4. Apply encoding for labels like the following using label Encoder:

```python
from sklearn.preprocessing import LabelEncoder

all_labels = [label for sublist in processed_df['labels'] for label in sublist]

label_encoder = LabelEncoder()

label_encoder.fit(all_labels)

ner_tags_str = processed_df['labels']

processed_df['labels'] = processed_df['labels'].apply(lambda x: label_encoder.transform(x))

unique_labels = label_encoder.classes_

print("Label Encoding Mapping:")
for label, encoded_label in zip(unique_labels, range(len(unique_labels))):
    print(f"{label} -> {encoded_label}")

print("Encoded Labels:")
print(processed_df['labels'])
```

```
Label Encoding Mapping:
O -> 0
1 <- اسم العنصر
2 <- الملاحظة
3 <- فترة
4 <- نوع العنصر
5 <- وقت
```

Figure 7.24

5. Define a function to tokenize and align labels with respect to the tokens. This function is specifically designed for Named Entity Recognition (NER) tasks where alignment of the labels is necessary after tokenization.
    a. Parameters:
        i. examples (dict): A dictionary containing the tokens and the corresponding NER tags.
        -"tokens": list of words in a sentence.

        -"ner_tags": list of corresponding entity tags for each.

        ii. label_all_tokens (bool): A flag to indicate whether all tokens should have labels.

        -If False, only the first token of a word will have a label, the other tokens (sub words) corresponding to the same word will be assigned -100.

    b. Returns:
        i. tokenized_inputs (dict): A dictionary containing the tokenized inputs and the corresponding labels aligned with the tokens.

```python
def tokenize_and_align_labels(examples, label_all_tokens=True):
    tokenized_inputs = tokenizer(examples["tokens"], truncation=True, is_split_into_words=True)
    labels = []
    for i, label in enumerate(examples["ner_tags"]):
        word_ids = tokenized_inputs.word_ids(batch_index=i)
        previous_word_idx = None
        label_ids = []
        for word_idx in word_ids:
            if word_idx is None:
                label_ids.append(-100)
            elif word_idx != previous_word_idx:
                label_ids.append(label[word_idx])
            else:
                label_ids.append(label[word_idx] if label_all_tokens else -100)

            previous_word_idx = word_idx
        labels.append(label_ids)
    tokenized_inputs["labels"] = labels
    return tokenized_inputs
```

Figure 7.25

6. Define a function to compute the evaluation metrics for Named Entity Recognition (NER) tasks.

   The function computes precision, recall, F1 score and accuracy.

   a. Parameters: eval_preds (tuple): A tuple containing the predicted logits and the true labels.
   b. Returns:

   A dictionary containing the precision, recall, F1 score and accuracy.

```
def compute_metrics(eval_preds):
    pred_logits, labels = eval_preds

    pred_logits = np.argmax(pred_logits, axis=2)
    predictions = [
        [unique_labels[eval_preds] for (eval_preds, l) in zip(prediction, label) if l != -100]
        for prediction, label in zip(pred_logits, labels)
    ]

    true_labels = [
      [unique_labels[l] for (eval_preds, l) in zip(prediction, label) if l != -100]
       for prediction, label in zip(pred_logits, labels)
   ]

    results = metric.compute(predictions=predictions, references=true_labels)
    return {
   "precision": results["overall_precision"],
   "recall": results["overall_recall"],
   "f1": results["overall_f1"],
  "accuracy": results["overall_accuracy"],
  }
```

Figure 7.26

7. Perform gird search in order to find the best values for hyperparameters
   a. Define parameter grid

```
parameter_grid = {
    'learning_rate': [5e-5, 3e-5, 2e-5],
    'per_device_train_batch_size': [16, 32],
    'num_train_epochs': [3, 4, 5],
    'weight_decay': [0.01, 0.1, 0.001]
}

best_score = float('-inf')
best_params = None
best_trainer = None
```

Figure 7.27

   b. Perform manual grid search

```python
for lr in parameter_grid['learning_rate']:
    for batch_size in parameter_grid['per_device_train_batch_size']:
        for epoch in parameter_grid['num_train_epochs']:
            for wd in parameter_grid['weight_decay']:
                print(f"Training with learning_rate={lr}, batch_size={batch_size}, num_train_epochs={epoch}, weight_decay={wd}")

                training_args = TrainingArguments(
                    output_dir='./TTC4900Model',
                    do_train=True,
                    do_eval=True,
                    num_train_epochs=epoch,
                    per_device_train_batch_size=batch_size,
                    per_device_eval_batch_size=batch_size,
                    warmup_steps=100,
                    weight_decay=wd,
                    logging_strategy='steps',
                    logging_dir='./multi-class-logs',
                    logging_steps=50,
                    evaluation_strategy="steps",
                    eval_steps=50,
                    save_strategy="steps",
                    fp16=True,
                    load_best_model_at_end=True,
                    learning_rate=lr
                )

                trainer = Trainer(
                    model=model,
                    args=training_args,
                    train_dataset=tokenized_datasets["train"],
                    eval_dataset=tokenized_datasets["test"],
                    data_collator=data_collator,
                    tokenizer=tokenizer,
                    compute_metrics=compute_metrics
                )

                trainer.train()
                eval_results = trainer.evaluate()
                score = eval_results['eval_f1']

                if score > best_score:
                    best_score = score
                    best_params = {'learning_rate': lr, 'batch_size': batch_size, 'num_train_epochs': epoch, 'weight_decay': wd}
                    best_trainer = trainer
```

Figure 7.28

8. Launch the training.
9. Evaluate the model.

Now we will illustrate the implementation of our functionalities across the mobile, desktop, and server which work together to successfully do the required task/function.

1. Add desktop to server:

Get desktop details (public Ip, private Ip, mac address) and send it to server to save this data on database.

```
export const add = asyncHandler(async (req, res, next) => {
    var device = await desktopModel.findOne({mac_address: req.body.mac_address});
    if(device){
        if(device.public_ip != req.body.public_ip || device.private_ip != req.body.private_ip){
            device.public_ip = req.body.public_ip;
            device.private_ip = req.body.private_ip;
            await device.save();
        }
        return device? res.status(200).json({ message: "Done", device }) : next(new Error(`Found Error`, { case: 400 }));
    }
    else {
        device = await desktopModel.create({
            public_ip: req.body.public_ip,
            private_ip: req.body.private_ip,
            mac_address: req.body.mac_address,
        });
        return device? res.status(201).json({ message: "Done", device }) : next(new Error(`Found Error`, { case: 400 }));
    }
})
```

Figure 7.29 - Desktop Code

```
1 usage
def addDeviceToServer(self):
    url = f"{self.url}/desktop"
    data = {
        "public_ip": self.publicIP,
        "private_ip": self.privateIP,
        "mac_address": self.macAddress
    }
    response = requests.post(url, json=data)
    if response.status_code == 200 or response.status_code == 201:
        print(f"Device added successfully!, {response.status_code} \n")
    else:
        print(f"request failed with status code: {response.status_code}\n")
```

Figure 7.30 - Server Code

2. Add Mobile to server:

get mobile details (device name, model, and version Number) then send it to server to save this data in database.

```
export const add = asyncHandler(async (req, res, next) => {
    var mobile = await mobileModel.create({
        name: req.body.name,
        model: req.body.model,
        version_number: req.body.version_number,
    });
    return mobile? res.status(201).json({ message: "Done", mobile }) : next(new Error(`Found Error`, { case: 400 }));
})
```

Figure 7.3.1 - Mobile Code

```
Future<void> addDeviceToServer() async {
  final headers = {'Content-Type': 'application/json'};
  String jsonData = jsonEncode({
    "name": deviceName,
    "model": deviceModel,
    "version_number": version_number,
  });
  final response = await http.post(
    Uri.parse('$serverUrl/mobile'),
    headers: headers,
    body: jsonData
  );

  if (response.statusCode == 200 || response.statusCode == 201) {
    deviceId = json.decode(response.body)["mobile"]["_id"];
    savedValues.setString("deviceId", deviceId);
    // print("device Id: ${deviceId} \n\n\n\n");
  } else {
    delay(sec: 1);
    addDeviceToServer();
  }
}
```

Figure 7.3.2 - Server Code

3. Create connection between Mobile and Desktop:

Mobile send Ip of desktop that want to connect to server to check if this desktop exists or not, if desktop exist then mobile send password to desktop to check if password is correct or not and if correct then mobile send connection data to server save it and mobile save desktop data on local database.

```
socket.on("checkIPResult", (data) {
  sendR = true;
  emit(updateSocketEmitState());
  if (data["found"] == true) {
    if (data["note"] == "add device") {
      socket.emit('addDevice', {"ip": new_ip, "type": "mobile"});
    }
    emitSocketEvent(ip: new_ip, event: "checkPassword", msg: {
      "password": new_password,
    });
  } else {
    showToast(flag: data["found"], message: data["message"]);
  }
});

socket.on("password", (data) async {
  if (data["valid"]) {
    String connectionId = await createConnectionOnServer(data["mac"]);
    insertToDatabase(
      name: new_name,
      password: new_password,
      ip: new_ip,
      mac: data["mac"],
      connectionId: connectionId,
    );
    togglePairLoading(state: false);
    Navigator.of(context).pop();
  }
  showToast(flag: data["valid"], message: data["message"]);
  togglePairLoading(state: false);
  sendR = true;
  emit(updateSocketEmitState());
});
```

Figure 7.33 - Mobile Code (Check IP and Password)

```dart
Future<String> createConnectionOnServer(desktop_mac) async {
  final headers = {'Content-Type': 'application/json'};
  String jsonData = jsonEncode({
    "desktop_mac": desktop_mac,
    "mobile_id": deviceId,
  });
  final response = await http.post(
      Uri.parse('$serverUrl/connection'),
      headers: headers,
      body: jsonData
  );
  if (response.statusCode == 200 || response.statusCode == 201) {
    var res = json.decode(response.body);
    return res["connection"]["_id"];
  } else {
    delay(sec: 1);
    return createConnectionOnServer(desktop_mac);
  }
}
```

Figure 7.34 - Mobile Code (Send connection data to server and save it local)

```python
@sio.event
def checkPassword(data):
    print("checkPassword: ", data["password"])
    result = ""
    valid = False
    # valid = True
    msg = {}
    if self.password == data["password"]:
        result = "You have Successfully paired to this device."
        valid = True
        msg["mac"] = self.macAddress
    else:
        result = "In-valid password"

    msg["message"] = result
    msg["valid"] = valid

    sio.emit( event: "event",  data: {
        "ip": ip,
        "type": "desktop",
        "target_type": "mobile",
        "event": "password",
        "message": msg,
        "eventError": "error",
        "messageError": "",
    })
```

Figure 7.35 - Desktop Code

```
socket.on("event", (data) => {
    const { event, message } = data;
    console.log(`event => ${event}`);
    var found = false

    devices.forEach(receiver => {
        if (receiver['ip'] == data['ip'] && receiver['type'] == data['target_type']) {
            found = true;
            io.to(receiver.id).emit(event, message);
        }
    });

    if (found == false)
        socket.emit(data["eventError"], { ip: data['ip'], messageError: data["messageError"], lostData: message })
})

socket.on("checkAvailableDevice", (data) => {
    console.log("checkAvailableDevice");
    var found = checkAvailability(data['ip'], data['target_type']);
    socket.emit("checkIPResult", {
        "message": found ? "found" : "Please check the IP",
        "found": found,
        "note": data["note"],
    });
});
```

Figure 7.36 - Server Code (Check IP exist and send data between mobile and desktop)

```
export const addConnection = asyncHandler(async (req, res, next) => {
    const { desktop_mac, mobile_id } = req.body;
    var connection = await connectionModel.findOne({ 'desktop_mac': desktop_mac, 'mobile_id': mobile_id, });
    if(connection){
        return res.status(200).json({ message: "Done", connection });
    }else {
        connection = await connectionModel.create({
            'desktop_mac': desktop_mac,
            'mobile_id': mobile_id,
        });

        const {public_ip} = await desktopModel.findOne({"mac_address": connection.desktop_mac});
        refreshDevicesConnection(public_ip);
    }
    return connection? res.status(200).json({ message: "Done", connection }) : next(new Error(`Found Error`, { case: 404 }));
})
```

Figure 7.37 - Server Code (Save Connection Data)

4. Auto Connection:

When desktop is active send to server his data to be active on server, when open mobile send mac addresses which saved on local database to server to get latest connection data between it and other desktops the update saved data on local database,

```
@sio.event
def connect():
    print('Connected to server')
    sio.emit( event: 'addDevice',  data: {
        # "id": sio.sid,
        "ip": ip,
        "type": "desktop"
    })
```

Figure 7.38 - Desktop Code (Send Data to Server)

```
Future<List<dynamic>> getNewIps(macList) async {
  final headers = {'Content-Type': 'application/json'};
  String jsonData = jsonEncode({
    "mac": macList,
  });
  final response = await http.post(
    Uri.parse('$serverUrl/desktop/information/$deviceId'),
    headers: headers,
    body: jsonData
  );

  if (response.statusCode == 200) {
    var ready_devices = json.decode(response.body)["ready_devices"];
    return ready_devices;
  } else {
    delay(sec: 1);
    return getNewIps(macList);
  }
}
```

Figure 7.39 - Mobile Code (get latest data of connected desktops)

```
Future<void> autoConnection() async {
  if(desktops.length==0){
    laoding=false;
    return;
  }
  var macList = [];
  desktops.forEach((device) => macList.add(device["mac"]));
  final ready_devices = await getNewIps(macList);
  for(int i=0; i<ready_devices.length; i++){
    if(ready_devices[i]["authorized"]==false){
      deleteActiveDevice(ready_devices[i]["public_ip"]);
      deleteData(mac: ready_devices[i]["mac_address"]);
      continue;
    }
    if(desktops[i]["ip"]!=ready_devices[i]["public_ip"]
        || desktops[i]["online"]!=ready_devices[i]["availabile"]){
      updateIpData(
        ip: ready_devices[i]["public_ip"],
        mac: ready_devices[i]["mac_address"],
        online: ready_devices[i]["available"],
        getDataFlag: false,
      );
    }

    if(ready_devices[i]["available"]==0) {
      deleteActiveDevice(ready_devices[i]["public_ip"]);
    }
  }
  getDataFromDatabase(database);
  laoding = false;
}
```

Figure 7.40 - Mobile Code (update local data server with latest data get from server)

```
export const getConnectedDevicesInformation = asyncHandler(async (req, res, next) => {
    const devices = await desktopModel.find({mac_address: req.body.mac});
    const connections = await connectionModel.find({mobile_id: req.params.mobile_id});
    var ready_devices = [];
    devices.forEach(element => {
        var authorized = false;
        for(var i=0; i<connections.length; i++){
            if(element.mac_address == connections[i].desktop_mac){
                authorized = true;
                break;
            }
        }
        if (authorized){
            ready_devices.push({
                authorized,
                public_ip: element.public_ip,
                private_ip: element.private_ip,
                mac_address: element.mac_address,
                available: checkAvailability(element.public_ip, "desktop")? 1 : 0,
                updatedAt: element.updatedAt,
            })
        }else{
            ready_devices.push({
                authorized,
                public_ip: element.public_ip,
                mac_address: element.mac_address,
            })
        }

    });
    return res.status(200).json({ message: "Done", ready_devices });
})
```

Figure 7.41 - Server Code

```
Future<void> getConnections() async {
  setState(() {...});
  final response = await http.get(
    Uri.parse('$serverLink/connection/desktop/$macAddress'),
  );

  if (response.statusCode == 200) {
    connections = json.decode(response.body)["mobiles"];
    print("\n\n\n ${connections} \n\n\n\n");
  }
  setState(() {...});
}
```

Figure 7.42 - Desktop Code (get all connected mobiles on desktop from server to show them)

5. Multi-device Synchronization:

Create lest for active desktops on mobile to swap between them

```
List<Map> active_desktops = [
  {
    "name": "Desktop Anywhere",
    "ip": "0.0.0.0",
    "listofpartitions": [],
    "path": "",
    "view": "Partitions",
    "roomId": "",
    "rotate": false,
    "sizeFactor": 1,
    "loading": false,
    "recordLoading": false,
    "isRecording": false,
  },
];
int addActiveDevice(ip, name, listofpartitions) {
  var index;
  bool found = false;
  for (var i = 1; i < active_desktops.length; i++) {
    if (active_desktops[i]['ip'] == ip) {
      found = true;
      index = i;
      break;
    }
  }
  if (!found) {
    active_desktops.add({...});
    index = active_desktops.length - 1;
    updateAccessTimeData(
      ip: ip,
      accessTime: getCurrentTime(),
    );
    emit(AddActiveDesktopList());
  }
  return index;
}
```

Figure 7.43 - Mobile Code

6. Get Partitions, Files and Folders:

Mobile sent request to get data of files and folders to desktop then desktop get all files and folders name of partition which sent in request then sent this data to mobile again throw the server.

```
IconButton(
  icon: Icon(
    Icons.arrow_circle_right_outlined,
    size: min(25*scaleFactor, 35),
    color: Colors.white,
  ),  // Icon
  padding: EdgeInsets.zero,
  onPressed: () async {
    cubit.toggleConnectDeviceLoading(index: index ,state: true);

    cubit.socket.emit('addDevice', {
      "ip": desktopData["ip"],
      "type": "mobile"
    });
    cubit.emitSocketEvent(
      ip: desktopData["ip"],
      event: "getPartition",
      msg: {
        "ip": desktopData["ip"],
        "target": "all",
        "deviceName": desktopData["name"],
        "part": "main",
        "index": index,
      },
      event_error: "notFoundDevice",
      msg_error: "This device not active now",
    );
  },
```

Figure 7.44 - Mobile Code (Send request to desktop)

```
socket.on('partition', (data) async {
  if (data["part"] == "main") {
    toggleConnectDeviceLoading(index: data["index"], state: false);
    addActiveDevice(data["ip"], data["deviceName"], data["partitions"]);
  }
  else {
    if(data["update"] == null) {
      data["update"] = 1;
    }
    setpath(
        foldername: data["componentName"],
        flag: data["path_flag"],
        update: data["update"],
        index: data["index"]
    );

    if(data["popScreen"] != null){
      Navigator.pop(context);
    }

    if(data["update"]==1){
      updateActiveDeviceLoading(index: data["index"]);
    }

    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => FilesAndFolder(...),  // FilesAndFolder
      ),  // MaterialPageRoute
    );
  }

  sendR = true;
  emit(updateSocketEmitState());
});
```

Figure 7.45 - Mobile Code (receive data of partitions from desktop)

```python
@sio.event
def getPartition(data):
    print(f"getPartition: ", data["target"])
    partitions = self.support.transfer_partition(data["target"])
    data["partitions"] = partitions
    sio.emit( event: "event",  data: {
        "ip": ip,  # IP
        "type": "desktop",  # mobile or desktop or web
        "target_type": "mobile",  # mobile or desktop or web
        "event": "partition",  # target event
        "message": data,
        "eventError": "error",  # error event if target not found
        "messageError": "",
    })
```

Figure 7.46 - Desktop Code (get request and get partition data then send to mobile)

```python
def transfer_partition(self, target):
    colon_index = target.find(':')

    if colon_index != -1: # Check if the colon is found
        # Create a new string with a backslash inserted after the colon
        target = target[:colon_index + 1] + '\\' + target[colon_index + 1:]

    if target == 'all':
        partitions = []
        for disk in psutil.disk_partitions():
            drive = disk.device
            partitions.append(drive)
        return partitions
    else:
        directories = []
        files = []
        content = {}
        if os.path.isdir(target):
            contents = os.listdir(target)
            for item in contents:
                item_path = os.path.join(target, item)
                if os.path.isdir(item_path):
                    directories.append(item)
                else:
                    files.append(item)
        content['dir'] = directories
        content['files'] = files

        # return jsonify(content)
        return content
```

Figure 7.47 - Desktop Code (get data of specific partition or folder)

7. Delete File or Folder:

Mobile send location of file or Folder to desktop to delete it

```dart
if (val.title == "Delete") {
  showDialog(
      context: context,
      builder: (BuildContext context) => ConfirmationPopup(
        title: "Alert",
        message: "Are you sure that you want to remove this Folder (${dataList["dir"][index]})?
        functionOfAcceptanceButton: (){
          cubit.emitSocketEvent(
            ip: ip,
            event: "deleteFileOrFolder",
            msg: {
              "ip": ip,
              "path": cubit.active_desktops[this.index]['path'].toString(),
              "deleted_name": dataList["dir"][index],
              "index": this.index,
            },
            skipWaiting: true,
          );
          cubit.showToast(
              waitingMsg: true,
              message: "Waiting for deleting"
          );
          Navigator.pop(context);
        },
      ),  // ConfirmationPopup
  );
}
```

Figure 7.48 - Mobile Code

```python
@sio.event
def deleteFileOrFolder(data):
    print("delete File Or Folder")
    print(data)
    delete_path = os.path.join(data["path"], data["deleted_name"])
    type = "folder" if os.path.isdir(delete_path) else "file"
    msg = f"Error, couldn't delete this {type}"
    isDeleted = self.support.deleteFilesAndFolders(delete_path)
    if isDeleted:
        msg = "F" + type[1:] + " deleted successfully"

    print(f"{msg} \n\n")
    sio.emit( event: "event",  data: {...})
    print(f"finish delete {type} \n")

    if not isDeleted: return
    self.updatePartioningView(socket=sio, ip=ip, path=data["path"], desktopIndexInFlutter=data["index"])
    print(f"Updated View after delete")
```

Figure 7.49 - Desktop Code

```python
def deleteFilesAndFolders(self, path):
    if os.path.exists(path):
        if os.path.isdir(path):
            os.rmdir(path)
            print(f"Folder '{path}' deleted successfully.")
        else:
            os.remove(path)
            print(f"File '{path}' deleted successfully.")
        return True
    else:
        print(f"File or Folder '{path}' does not exist.")
        return False
```

Figure 7.50 - Desktop Code (delete function)

8. Copy and Paste Files:

Mobile send message to desktop with path of file want to copy then desktop get this message and upload this file to server with details of file and mobile Id to store it in server database, then when paste mobile send to desktop the path of folder he want to past in and sent his id then desktop send to server to get file details of this id then download and save it in desktop after that desktop send to server to delete this file from server storage.

```dart
if (val.title == "Copy") {
  cubit.showToast(
      waitingMsg: true,
      message: "Waiting for ${val.title}"
  );
  cubit.emitSocketEvent(
    ip: ip,
    event: "uploadFile",
    msg: {
      "ip": ip,
      "path": path,
      "deviceId": cubit.deviceId,
    },
    skipWaiting: true,
  );
}
```

Figure 7.51 - Mobile Code (send message to desktop to copy file)

```
void pasteFile({ip, index, path=""}){
  showToast(
      waitingMsg: true,
      message: "Waiting for paste"
  );
  emitSocketEvent(
    ip: ip,
    event: "downloadFile",
    msg: {
      "ip": ip,
      "path": path.isNotEmpty? path : active_desktops[index]['path'].toString(),
      "deviceId": deviceId,
      "index": index,
    },
    skipWaiting: true,
  );
}
```

Figure 7.52 - Mobile Code (send message to desktop to paste file)

```
@sio.event
def uploadFile(data):
    print("upload File")
    print(data)
    path = data["path"]
    file_size = os.path.getsize(path)
    isUploaded = 0
    msg = "Error, file size more than 50MB"
    if file_size <= 52428800:
        isUploaded = self.uploadFileToServer(data["deviceId"], path)
        msg = "File copy successfully"

    sio.emit( event: "event",  data: {
        "ip": ip,  # IP
        "type": "desktop",  # mobile or desktop or web
        "target_type": "mobile",  # mobile or desktop or web
        "event": "uploadResults",  # target event
        "message": {
            "isUploaded": isUploaded,
            "message": msg,
            "forMobile": data["forMobile"] if data.__contains__("forMobile") else False,
            "directoryPathInMobile": data["directoryPathInMobile"] if data.__contains__(
                "directoryPathInMobile") else "",
        },
        "eventError": "error",  # error event if target not found
        "messageError": "",
    })
```

Figure 7.53 - Desktop Code (receive copy message from mobile and call uploading function)

```python
def uploadFileToServer(self, mobile_Id, path):
    print("Start Uploaded")
    url = f"{self.url}/media"
    data = {"mobile_Id": mobile_Id, }
    file = {"file": open(path, 'rb'), }
    response = requests.post(url, files=file, data=data)
    if response.status_code == 200 or response.status_code == 201:
        print(f"File uploaded successfully!, {response.status_code} \n")
        return True
    else:
        print(f"request failed with status code: {response.status_code}\n")
        return False
```

Figure 7.54- Desktop Code (upload file function)

```python
@sio.event
def downloadFile(data):
    print("download File")
    print(data)
    mobileId = data["deviceId"]
    save_path = data["path"]
    isDownloaded = self.getFileDataFromServer(mobileId, save_path)
    msg = "Error, file couldn't paste"
    if isDownloaded:
        msg = "File paste successfully"

    print(f" {msg} \n\n")

    sio.emit( event: "event", data: {
        "ip": ip,  # IP
        "type": "desktop",  # mobile or desktop or web
        "target_type": "mobile",  # mobile or desktop or web
        "event": "downloadResults",  # target event
        "message": {
            "isDownloaded": isDownloaded,
            "message": msg
        },
        "eventError": "error",  # error event if target not found
        "messageError": "",
    })
    print("finish download File")

    if not isDownloaded: return
    self.updatePartioningView(socket=sio, ip=ip, path=data["path"], desktopIndexInFlutter=data["index"])
```

Figure 7.55 - Desktop Code (receive paste message from mobile and call downloading function)

```python
def getFileDataFromServer(self, mobile_Id, save_path):
    url = f"{self.url}/media/view/{mobile_Id}"
    response = requests.get(url)
    if response.status_code == 200:
        body = response.json()["file"]
        file_name = body["file_name"]
        path = os.path.join(save_path, file_name)
        self.downloadFileFromServer(path, body["secure_url"])
        print(f"File downloaded successfully!, {response.status_code} \n")
        self.deleteFileFromServer(mobile_Id)
        return True
    else:
        print("Error: ", response.status_code)
        return False
```

Figure 7.56 - Desktop Code (download function)

```python
def downloadFileFromServer(self, filePath, serverPath):
    downloadUrl = f'{self.url}/{serverPath}'
    req = requests.get(downloadUrl)
    with open(filePath, 'wb') as f:
        for chunk in req.iter_content(chunk_size=8192):
            if chunk:
                f.write(chunk)

1 usage
def deleteFileFromServer(self, mobile_Id):
    url = f"{self.url}/media/{mobile_Id}"
    response = requests.delete(url)
    if response.status_code == 200:
        print(f"File deleted from server successfully!, {response.status_code} \n")
    else:
        print("Error: ", response.status_code)
```

Figure 7.57 - Desktop Code (functions download file then delete it from server)

```
export const add = asyncHandler(async (req, res, next) => {
    const mobile_Id = req.body.mobile_Id;
    var media = await mediaModel.find({ createdBy: mobile_Id });
    if (media.length > 0) {
        media.forEach(element => {
            deleteFile(element.secure_url);
        });
        await mediaModel.deleteMany({ createdBy: mobile_Id });
    }

    media = await mediaModel.create({
        createdBy: mobile_Id,
        secure_url: req.file.dest,
        file_name: req.file.originalname,
        mimetype: req.file.mimetype,
        size: req.file.size,
    });

    return res.status(201).json({ message: "Done", media });
})
```

Figure 7.58 - Server Code (upload file and store its details in database)

```
export const getOne = asyncHandler(async (req, res, next) => {
    const mobile_Id = req.params.mobile_Id;
    const file = await mediaModel.findOne({ createdBy: mobile_Id });
    if (file) {
        // Check if the file exists
        if(! await fileExists(file.secure_url)){
            console.log('File does not exist');
            await mediaModel.deleteMany({ createdBy: mobile_Id });
            return next(new Error(`No files found for this Id: ${mobile_Id}`, { cause: 404 }));
        }
    }
    return file? res.status(200).json({ message: "Done", file })
        : next(new Error(`In-valid or Not Found Id or No files for this Id: ${mobile_Id}`, { cause: 404 }));
})
```

Figure 7.59 - Server Code (get file uploaded details)

```
export const deleteOne = asyncHandler(async (req, res, next) => {
    const mobile_Id = req.params.mobile_Id
    var media = await mediaModel.find({ createdBy: mobile_Id });
    media.forEach(async (element) => {
        if(await fileExists(element.secure_url)){
            deleteFile(element.secure_url);
        }
    });
    media = await mediaModel.deleteMany({ createdBy: mobile_Id });
    return media.deletedCount > 0 ? res.status(200).json({ message: "Done" })
        : next(new Error(`In-valid or Not Found Id or No files found for this Id: ${req.params.mobile_Id} `, { cause: 404 }));
})
export function deleteFile(url) {
    const fullPath = path.join(__dirname, `./../../../../${url}`)
    try {
        fs.unlinkSync(fullPath);
    }
    catch (error) {
        console.log(error);
    }
}
export function fileExists(url) {
    const filePath = path.join(__dirname, `./../../../../${url}`);
    return new Promise((resolve, reject) => {
        fs.access(filePath, fs.constants.F_OK, (err) => {
            if (err) {  // File does not exist or cannot be accessed
                resolve(false);
            } else { // File exists
                resolve(true);
            }
        });
    });
}
```

Figure 7.60 - Server Code (delete file form server storage)

9. Save file in mobile storage from desktop:

Mobile makes the user pick directory which save file on it then send message to desktop to upload this file on server to download it in mobile storage.

```
else if (val.title == "Save") {
  cubit.showToast(
      waitingMsg: true,
      message: "Choose location for saving ..."
  );
  String directoryPath = await cubit.pickDirectory();
  cubit.showToast(
      waitingMsg: true,
      message: "Waiting for downloading file"
  );
  cubit.emitSocketEvent(
    ip: ip,
    event: "uploadFile",
    msg: {
      "ip": ip,
      "path": path,
      "deviceId": cubit.deviceId,
      "forMobile": true,
      "directoryPathInMobile": directoryPath,
    },
    skipWaiting: true,
  );
 }
},
```

Figure 7.61 - Mobile Code (call pick function and send upload message to desktop)

```dart
Future<String> pickDirectory() async {
  String? selectedDirectory = await FilePicker.platform.getDirectoryPath();
  selectedDirectory ??= "";
  return selectedDirectory;
}
```

Figure 7.62 - Mobile Code (pick directory function)

```dart
Future<void> downloadFile(String directoryPath) async {
  //...
  var time = DateTime.now().microsecondsSinceEpoch;
  final response = await http.get(
    Uri.parse('$serverUrl/media/view/$deviceId'),
  );
  if (response.statusCode == 200) {
    var res = json.decode(response.body);
    print("\n\n\n ${res} \n\n\n\n");
    res = res["file"];
    directoryPath = "$directoryPath/${res["file_name"]}__$time";
    var file = File(directoryPath);
    var fileResponse = await http.get(
      Uri.parse('$serverUrl/${res["secure_url"]}'),
    );
    if(fileResponse.statusCode == 200){
      await file.writeAsBytes(fileResponse.bodyBytes);
      showToast(flag: true, message: "File downloaded successfully");
    } else{
      showToast(flag: false, message: "Failed download file");
    }
  } else {
    showToast(flag: false, message: "Failed download file");
  }
}
```

Figure 7.63 - Mobile Code (download file from server and save it in mobile storage)

```python
@sio.event
def uploadFile(data):
    print("upload File")
    print(data)
    path = data["path"]
    file_size = os.path.getsize(path)
    isUploaded = 0
    msg = "Error, file size more than 50MB"
    if file_size <= 52428800:
        isUploaded = self.uploadFileToServer(data["deviceId"], path)
        msg = "File copy successfully"

    sio.emit( event: "event",  data: {
        "ip": ip,  # IP
        "type": "desktop",  # mobile or desktop or web
        "target_type": "mobile",  # mobile or desktop or web
        "event": "uploadResults",  # target event
        "message": {
            "isUploaded": isUploaded,
            "message": msg,
            "forMobile": data["forMobile"] if data.__contains__("forMobile") else False,
            "directoryPathInMobile": data["directoryPathInMobile"] if data.__contains__(
                "directoryPathInMobile") else "",
        },
        "eventError": "error",  # error event if target not found
        "messageError": "",
    })
```

Figure 7.64 - Desktop Code

10. Create Foder on Desktop:

Take folder name from user then send it with path want to save to desktop to create this folder

```
showDialog(
  context: context,
  builder: (BuildContext context) => ConfirmationPopup(
    title: "Create Folder",
    titleColor: Colors.black,
    messageWidget: const SizedBox(),
    textOfAcceptanceButton: "Ok",
    textOfRejectionButton: "Cancel",
    content: TextField(
      controller: _textFieldController,
      decoration: const InputDecoration(
        hintText: "Enter folder name ..",
        hintStyle: TextStyle(
          color: Colors.grey,
          fontWeight: FontWeight.normal,
          fontSize: 20,
        ),  // TextStyle
      ),  // InputDecoration
    ),  // TextField
    functionOfAcceptanceButton: (){
      if(_textFieldController.text.isEmpty)
      {
        return;
      }
      Navigator.pop(context);
      //...

      cubit.showToast(
          waitingMsg: true,
          message: "Waiting for creating"
      );
      cubit.emitSocketEvent(
        ip: ip,
        event: "createFolder",
        msg: {...},
        skipWaiting: true,
      );
    },
  ),  // ConfirmationPopup
);
```

Figure 7.65 - Mobile Code

```python
@sio.event
def createFolder(data):
    print(data)
    folder_name = data["folder_name"]
    path = data["path"]
    new_folder_path = os.path.join(path, folder_name)

    msg = ""
    isCreated = False
    try:
        os.mkdir(new_folder_path)
        isCreated = True
        msg = f"Folder '{folder_name}' created at '{path}' successfully"
    except FileExistsError:
        msg = f"Folder '{folder_name}' already exists at '{path}'"
    except OSError as e:
        msg = f"Error: {e}"

    print(f"msg: {msg} \n\n")
    sio.emit( event: "event",  data: {...})
    print(f"finished create {type} \n")
    if not isCreated: return
    self.updatePartioningView(socket=sio, ip=ip, path=data["path"], desktopIndexInFlutter=data["index"])
    print(f"Updated View after created")
```

Figure 7.66 - Desktop Code

11. Virtual Control:

Detect the position of touch on mobile screen or key pressed then send it to desktop to perform this action on desktop.

```dart
child: GestureDetector(
  onPanStart: (details) {
    previousOffset = details.globalPosition;
  },
  onPanUpdate: (x) {
    Size? size = _key.currentContext
        ?.findRenderObject()
        ?.paintBounds
        .size;

    // Calculate the difference between current and previous positions
    double deltaX = x.globalPosition.dx - previousOffset.dx;
    double deltaY = x.globalPosition.dy - previousOffset.dy;

    // Update the previous position
    previousOffset = x.globalPosition;
    coordinate["X"] = deltaX;
    coordinate["Y"] = deltaY;
    coordinate["height"] = size!.width;
    coordinate["width"] = size!.height;

    // cubit.sendMessageTouchpad(coordinate, ip, 8888);
    cubit.emitSocketEvent(
      ip: ip,
      event: "mouse_move",
      msg: coordinate,
      event_error: "errorMouseMove",
      msg_error: "device not found",
      skipWaiting: true,
    );
```

Figure 7.67 - Mobile Code (detect touch position and send it to desktop)

```
child: Row(
  children: [
    Expanded(
      child: InkWell(
        onDoubleTap: (){
          cubit.emitSocketEvent(
            ip: ip,
            event: "mouse_click",
            msg: "Double Click",
            skipWaiting: true,
          );
        },
        onTap: (){
          cubit.emitSocketEvent(
            ip: ip,
            event: "mouse_click",
            msg: "Left Click",
            skipWaiting: true,
          );
        },
        child: Container(...),  // Container
      ),  // InkWell
    ),  // Expanded
    const SizedBox(...),  // SizedBox
    Expanded(
      child: InkWell(
        onTap: (){
          cubit.emitSocketEvent(
            ip: ip,
            event: "mouse_click",
            msg: "Right Click",
            skipWaiting: true,
          );
        },
        child: Container(...),  // Container
      ),  // InkWell
    ),  // Expanded
  ],
),  // Row
```

Figure 7.68 - Mobile Code (detect clicking and send it to desktop)

```dart
return Expanded(
  child: Container(
    decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(60.0),
      color: Colors.grey[200],
    ),  // BoxDecoration
    child: TextButton(
      onPressed: onPressed ?? () {
        if (!cubit.isArabic) {
          cubit.emitSocketEvent(
              ip: ip,
              event: "keyboard",
              msg: text,
              skipWaiting: true,
          );
          print(text);
        }
        else {
          cubit.emitSocketEvent(
            ip: ip,
            event: "keyboard",
            msg: (cubit.isUppercase && cubit.isArabic)?text.toUpperCase():text.toLowerCase(),
            skipWaiting: true,
          );
        }
      },
      child: Text(
        cubit.isUppercase ? text.toUpperCase() : text.toLowerCase(),
        style: const TextStyle(
          color: Colors.black,
          fontWeight: FontWeight.bold,
        ),  // TextStyle
      ),  // Text
    ),  // TextButton
  ),  // Container
);  // Expanded
```

Figure 7.69 - Mobile Code (detect key pressed and send it to desktop)

```python
@sio.event
def mouse_click(data):
    print('mouse_click: ', data)
    mouse = Controller()
    if data == "Left Click":
        mouse.click(Button.left)
    elif data == "Right Click":
        mouse.click(Button.right)
    elif data == "Double Click":
        mouse.click(Button.left)
        time.sleep(0.1)  # Optional: Add a short delay between clicks
        mouse.click(Button.left)

@sio.event
def mouse_move(data):
    print(f"mouse_move, X:{int(data['X'])}, Y:{int(data['Y'])}")
    # Extract the x and y coordinates from the dictionary
    mobile_x = data["X"]
    mobile_y = data["Y"]
    mobile_width = data["width"]
    mobile_height = data["height"]

    desktop_width = win32api.GetSystemMetrics(0)
    desktop_height = win32api.GetSystemMetrics(1)

    desktop_x, desktop_y = self.support.map_coordinates(mobile_x, mobile_y, mobile_width, mobile_height,
                                                        desktop_width,
                                                        desktop_height)

    mouse = Controller()
    if data["touchpad"] == 0:...
    else:...
```

Figure 7.70 - Desktop Code (perform mouse move and click that coming from mobile)

```python
@sio.event
def keyboard(data):
    print("keyboard: ", data)
    message = data
    if len(message) > 1:
        pyautogui.press(f'{message}')
    elif message == '\n':
        # Copy the newline character to the clipboard
        pyperclip.copy('\n')
        # Use PyAutoGUI to paste the newline character
        pyautogui.hotkey( *args: 'ctrl', 'v')
    else:
        pyperclip.copy(message)
        # Use PyAutoGUI to paste the text from the clipboard
        pyautogui.hotkey( *args: 'ctrl', 'v')
```

Figure 7.71 - Desktop Code (perform key pressed that coming from mobile)

12. Share Desktop Screen:

Mobile create peer to peer connection room throw WebRTC then send room id to desktop to join to room

```dart
Future<void> shareScreen() async {
  widget.roomId =
      await signaling.createRoom(_remoteRenderer, widget.configuration);

  widget.socket.emit("event", {
    "ip": widget.ip,
    "type": "mobile",
    "target_type": "web",
    "event": "roomId",
    "message": {
      "roomId": widget.roomId,
    },
    "eventError": "errorRoomId", // error event if target not found
    "messageError": "device not found",
  });
}
```

Figure 7.72 - Mobile Code (create room and send to desktop)

```dart
Future<String> createRoom(RTCVideoRenderer remoteRenderer,
    Map<String, dynamic> configuration) async {
  FirebaseFirestore db = FirebaseFirestore.instance;
  DocumentReference roomRef = db.collection('rooms').doc();
  print('Create PeerConnection with configuration: $configuration');
  peerConnection = await createPeerConnection(configuration);
  registerPeerConnectionListeners();
  localStream?.getTracks().forEach((track) {
    peerConnection?.addTrack(track, localStream!);
  });

  // Code for collecting ICE candidates below
  var callerCandidatesCollection = roomRef.collection('callerCandidates');
  peerConnection?.onIceCandidate = (RTCIceCandidate candidate) {
    print('Got candidate: ${candidate.toMap()}');
    callerCandidatesCollection.add(candidate.toMap());
  }; // Finish Code for collecting ICE candidate

  // Add code for creating a room
  RTCSessionDescription offer = await peerConnection!.createOffer();
  await peerConnection!.setLocalDescription(offer);
  print('Created offer: $offer');
  Map<String, dynamic> roomWithOffer = {'offer': offer.toMap()};
  await roomRef.set(roomWithOffer);
  var roomId = roomRef.id;
  print('New room created with SDK offer. Room ID: $roomId');
  currentRoomText = 'Current room is $roomId - You are the caller!';

  // Created a Room
  peerConnection?.onTrack = (RTCTrackEvent event) {
    print('Got remote track: ${event.streams[0]}');
    event.streams[0].getTracks().forEach((track) {
      print('Add a track to the remoteStream $track');
      remoteStream?.addTrack(track);
    });
  };

  // Listening for remote session description below
  roomRef.snapshots().listen((snapshot) async {
    print('Got updated room: ${snapshot.data()}');
    Map<String, dynamic> data = snapshot.data() as Map<String, dynamic>;
    if (peerConnection?.getRemoteDescription() != null && data['answer'] != null) {
      var answer = RTCSessionDescription(
        data['answer']['sdp'],
        data['answer']['type'],
      );
      print("Someone tried to connect");
      await peerConnection?.setRemoteDescription(answer);
    }
  }); // Listening for remote session description above

  // Listen for remote Ice candidates below
  roomRef.collection('calleeCandidates').snapshots().listen((snapshot) {
    snapshot.docChanges.forEach((change) {
      if (change.type == DocumentChangeType.added) {
        Map<String, dynamic> data = change.doc.data() as Map<String, dynamic>;
        print('Got new remote ICE candidate: ${jsonEncode(data)}');
        peerConnection!.addCandidate(
          RTCIceCandidate(
            data['candidate'],
            data['sdpMid'],
            data['sdpMLineIndex'],
          ),
        );
      }
    });
  }); // Listen for remote ICE candidates above
  return roomId;
}
```

Figure 7.73 - Mobile Code (function create peer to peer connection room)

```dart
Future<void> joinRoom(String roomId, RTCVideoRenderer remoteVideo,
    Map<String, dynamic> configuration,) async {
  this.roomId = roomId;
  FirebaseFirestore db = FirebaseFirestore.instance;
  print(roomId);
  DocumentReference roomRef = db.collection('rooms').doc('$roomId');
  var roomSnapshot = await roomRef.get();
  print('Got room ${roomSnapshot.exists}');
  if (roomSnapshot.exists) {
    print('Create PeerConnection with configuration: $configuration');
    peerConnection = await createPeerConnection(configuration);
    registerPeerConnectionListeners();
    localStream?.getTracks().forEach((track) {
      peerConnection?.addTrack(track, localStream!);
    });

    // Code for collecting ICE candidates below
    var calleeCandidatesCollection = roomRef.collection('calleeCandidates');
    peerConnection!.onIceCandidate = (RTCIceCandidate? candidate) {
      if (candidate == null) {...}
      print('onIceCandidate: ${candidate.toMap()}');
      calleeCandidatesCollection.add(candidate.toMap());
    }; // Code for collecting ICE candidate above

    peerConnection?.onTrack = (RTCTrackEvent event) {
      print('Got remote track: ${event.streams[0]}');
      event.streams[0].getTracks().forEach((track) {
        print('Add a track to the remoteStream: $track');
        remoteStream?.addTrack(track);
      });
    };

    // Code for creating SDP answer below
    var data = roomSnapshot.data() as Map<String, dynamic>;
    print('Got offer $data');
    var offer = data['offer'];
    await peerConnection?.setRemoteDescription(
      RTCSessionDescription(offer['sdp'], offer['type']),
    );
    var answer = await peerConnection!.createAnswer();
    print('Created Answer $answer');
    await peerConnection!.setLocalDescription(answer);
    Map<String, dynamic> roomWithAnswer = {
      'answer': {'type': answer.type, 'sdp': answer.sdp}
    };
    await roomRef.update(roomWithAnswer);
    // Finished creating SDP answer

    // Listening for remote ICE candidates below
    roomRef.collection('callerCandidates').snapshots().listen((snapshot) {
      snapshot.docChanges.forEach((document) {
        var data = document.doc.data() as Map<String, dynamic>;
        print(data);
        print('Got new remote ICE candidate: $data');
        peerConnection!.addCandidate(
          RTCIceCandidate(
            data['candidate'],
            data['sdpMid'],
            data['sdpMLineIndex'],
          ),
        );
      });
    });
  }
}
```

Figure 7.74 - Desktop Code (function join to room)

```
socket.on("roomId", (data) {
  print(data);
  signaling.hangUp(_localRenderer);
  signaling.joinRoom(data["roomId"], _remoteRenderer, configuration);
});
```

Figure 7.75 - Desktop Code (receive room id from mobile and join it)

13. Voice Command:

Record voice command then store it in mobile then send to desktop and save it to perform the order on it.

```
FloatingActionButton(
onPressed: () async {
  if(cubit.active_desktops[index]["recordLoading"]) return;

  if(cubit.active_desktops[index]["isRecording"]==false){
    cubit.toggleRecordState(index: index, state: true);
    cubit.recordVoice();
  }
  else {
    await cubit.stopRecording();
    File audioFile = File(cubit.filePath);
    List<int> audioBytes = audioFile.readAsBytesSync();
    // print("Audio Finished \n\n\n\n\n\n\n");
    cubit.toggleRecordLoadingState(index: index, state: true);
    cubit.toggleRecordState(index: index, state: false);

    cubit.emitSocketEvent(
      ip: ip,
      event: "voice",
      msg: audioBytes,
      event_error: "notFoundDevice",
      msg_error: "Voice not sent",
      skipWaiting: true,
    );
  }
},
```

Figure 7.76 - Mobile Code (create voice and send it to desktop)

```dart
Future<String> makeDirectory({required String dirName}) async {
  final directory = await getExternalStorageDirectory();
  final formattedDirectory = '/$dirName/';
  final Directory newDir =
      await Directory(directory!.path + formattedDirectory).create();
  return newDir.path;
}

get getAudioDir async => await makeDirectory(dirName: 'recordings');
Future<String> createRecordAudioPath({
  required String dirPath,
  required String fileName,
}) async {
  return """$dirPath${fileName.substring(0, min(fileName.length, 100))}.wav""";
}

recordVoice() async {
  final voiceDirPath = await getAudioDir;
  final voiceFilePath = await createRecordAudioPath(
      dirPath: voiceDirPath, fileName: "audio_message");
  await record.start(const RecordConfig(encoder: AudioEncoder.wav),
      path: voiceFilePath);
  emit(StartRecordState());
}

stopRecording() async {
  String? audioFilePath;
  audioFilePath = await record.stop();
  print('file path : $audioFilePath');
  filePath = audioFilePath ?? "";
  emit(EndRecordState());
}
```

Figure 7.77 - Mobile Code (voice control functions)

```python
@sio.event
def voice(data):
    print("voice")
    voice_file = data

    # Save the file to a specified directory
    save_dir = 'datasets'
    # save_dir = 'saved_voices_socket'
    if not os.path.exists(save_dir):
        os.makedirs(save_dir)

    save_path = os.path.join(save_dir, "received_audio.wav")
    with open(save_path, 'wb') as f:
        f.write(bytearray(voice_file))
    print(f"Audio saved to: {save_path}")

    msg = "Command Done"
    flag = True
    sio.emit( event: "event",  data: {
        "ip": ip,  # IP
        "type": "desktop",  # mobile or desktop or web
        "target_type": "mobile",  # mobile or desktop or web
        "event": "voiceArrived",  # target event
        "message": {...},
        "eventError": "error",  # error event if target not found
        "messageError": "",
    })
    self.model.run(save_path)
```

Figure 7.78 - Desktop Code (receive voice and save it to perform it order)

Chapter 8

< System Testing and Deployment >

The purpose of our quality assurance process is to Identifies the SQA, software quality assurance, responsibilities of the project developers or team members. And defines how to review and audit the process. Also Lists the activities, processes, and work products that we will review and audit.

1. Management:

- Team members consists of:
  - Team leader: Andrew Samir.
  - Team members: Sara Mazhar, Rehab Youssef, Bassant Hossam, Hossam Hatem.
- Tasks:
  - Develop the requirement specification.
  - Develop the project plan and test plan.
  - Implement and test, Deliver the final version along with the documentation.
- Responsibilities:
  - The developer or team member will perform all software development tasks.
  - The team leader will review the work performed by the team members and provide feedback and advice.

2. Documentation:

The following documents will be provided at the end of each phase.

Phase 1: Software Requirement

- Project Overview
- System Diagram
- Project Gantt Chart Timeline
- System Requirements Specification (SRS)

Phase 2: Software Design

- Test Plan

Phase 3: Software Implementation

- Software test plan
- Project Evaluation

Appendix:

- Source Code

3. Software test plan

A Software Test Plan (STP) will be written to satisfy the software requirements. The plan will provide functional testing and non-functional testing.

- **Functional testing**

1- Unit testing

- Each individual feature will be tested (e.g., Remote Desktop Access, Voice Commands, Multi-Device Sync, Virtual Touchpad/Keyboard, Autoconnection, and File/Folder Search) to verify that the feature performs the required functions and outputs the correct results.

2- Integration testing

- Test the interactions between major components (e.g., mobile application, desktop application, and API server) to ensure seamless communication and data transfer between these elements.

3- System testing

- Once the development is completed, the system can be evaluated as a whole. End-to-end testing of the entire system will be performed to verify that all user requirements are met.

- **Non-Functional testing**

1. Performance Testing

- Our performance testing involves evaluating the swift response times of our remote desktop access application for tasks such as accessing the desktop remotely and managing files. We assess how efficiently the system handles these tasks under various network conditions and device setups, ensuring a seamless user experience. For example, we measure the response time for accessing the desktop remotely and transferring files, ensuring consistency across different scenarios.

2. Availability Testing

- ensure that our remote desktop access application is designed for continuous operation, allowing users to access their desktop computers at any time. We assess the system's ability to remain accessible and functional without interruptions, even during peak usage periods. For example, we test the system's response to high user loads and ensure that it maintains availability without degradation in performance.

3. Usability Testing

- Usability testing focuses on evaluating the intuitiveness of our remote desktop access application's user interface. We ensure that users can navigate the application proficiently with minimal training, enhancing their overall experience. For example, we assess the clarity of navigation paths, the intuitiveness of menu options, and the ease of performing common tasks such as connecting to a desktop remotely or files access.

4. Reliability Testing**.**

- Reliability testing ensures that our remote desktop access application maintains its performance over time, providing consistent and stable functionality. We assess the system's ability to withstand continuous usage without experiencing frequent failures or disruptions. For example, we test for potential issues such as crashes, or connectivity problems, ensuring that users can rely on the system to maintain a stable connection to their desktop computers.

We provide last version of test cases and testing scenarios according to the software requirement specifications:

Test Scenario	Test Scenario Description	Test Case ID	Test Case Description	Test Steps	Precondition	Test Data	Post Condition	Expected Result	Actual	Status
TS_01	Remote Desktop Access	01_1	Verify successful connection to a desktop with valid credential (IP Adress And Password)	1. Launch a desktop application. 2. Enter Device Name. 3. Enter **a valid** IP Address through mobile app. 4. Enter **a valid** Password through mobile app. 5. Click on Pair button.	**valid** IP, **valid** Password, and **valid** name	Device name: MyDesktop IP: 192.168.1.100 Password: 1e2b3#6$	-	Connection established successfully And user to go homepage	-	**Pass**
		01_2	Verify connection failure with invalid **invalid** IP and **valid** Password	1. Launch remote desktop application. 2. Enter Device Name. 3. Enter **an invalid** IP Address. 4. Enter **valid** Password 5. Click on Pair button.	**invalid** IP, **valid** Password, and **valid** name	Device name: MyDesktop IP: 192.168.1 Password: 1e2b3#6$	Connection error message displayed	Connection failed due to invalid IP	-	**Pass**
		01_3	Verify connection failure with invalid Password and valid IP	1. Launch remote desktop application. 2. Enter Device Name. 3. Enter **a valid** IP Address. 4. Enter **invalid** Password. 5. Click on Pair button	**valid** IP, **invalid** Password, and **valid** name	Device name: MyDesktop IP: 192.168.1.100 Password: 1e2b	Connection error message displayed	Connection failed due to invalid Password	-	**Pass**
		01_4	Verify connection failure with click on pair button by leaving device name field	1. Launch remote desktop application. 2. leave Device Name field empty 3. Enter **valid** IP Address 4. Enter **valid** Password 5. Click on Pair button	**valid** IP, **valid** Password, and **invalid** name	Device name: IP: 192.168.1.100 Password: 1e2b3#6$	Connection error message displayed	Connection failed due to missing device name	-	**Pass**

		01_5	Verify connection failure with invalid Password and invalid IP	1. Launch remote desktop application. 2. Enter Device Name 3. Enter **invalid** IP Address 4. Enter **invalid** Password 5. Click on Pair button	**invalid** IP, **invalid** Password, and **invalid** name	Device name: IP: 192.168 Password: 1e2b	Connection error message displayed	Connection failed due to invalid credentials	-	**Pass**
		01_6	Verify Touch movement.	1. After successful connection, the user should navigate to virtual control page or share screen. 2. Use the designed area for mouse.	Successful connection to desktop	-	-	Accurate and live mouse movement according to the finger position on the mobile screen.	-	**Pass**
		01_7	Verify that users have a control over the mouse buttons	1. After successful connection, the user should navigate to virtual control page or share screen. 2. Use the designed buttons for mouse.	Successful connection to desktop	-	-	Action is taken according to the finger position on the mobile screen.	-	**Pass**
		01_8	Verify share screen of the desktop	1. Click on the device you want to connect. 2.Choose Share Screen from list on Navigation	Successful connection to desktop	-	device's screen is displayed	The connected device's screen should appear on app mobile	-	**Pass**
		01_9	Verify Letters insertion using Mobile keyboard at the cursor position on desktop	1. After successful connection, the user should navigate to virtual control page or share screen. 2. Use the designed area for Keyboard.	Successful connection to desktop	-	-	Letters successfully inserted	-	**Pass**

		01_10	Verify Access of partitions internal data	1. After successful connection, choose your device. 2. Choose any partition.	Successful connection to desktop	-	-	Successful access	-	**Pass**
		01_11	Verify the copy, delete, and save files are working correctly	1. After successful connection, choose your device. 2. Choose any partition. 3. Perform one of the available operations on any file	Successful connection to desktop	-	-	Successful operation	-	**Pass**
TS-02	Voice Command	02-1	Verify correctness of command execution considering voice variations (e.g. speed, noise)	1. After successful connection, Activate voice command feature. 2. Execute any command with different voice variations.	Successful connection to desktop and Voice command enabled	-	results displayed	commands executed accurately with different voice variations	-	**Pass**
		02-2	Verify having the ability to cancel sending the command after recording it.	1. After successful connection, Activate voice command feature. 2. record voice command. 3.cancel the record.	Successful connection to desktop and Voice command enabled	-	-	Voice Command Canceled	-	**Pass**
TS_03	Multidevice synchronization	03-1	Verify multi-device synchronization	1. Launch a desktop application. 2. Enter Device Name. 3. Enter **a valid** IP Address through mobile app. 4. Enter **a valid** Password through mobile app. 5. Click on the Pair button. Repeat the steps for the other desktop to be connected.	Successful connection to more than one desktop	-	-	multi-device synchronization is done successfully	-	**Pass**

		03-1	Verify data is accurately exchanged across multi-devices	1. Launch a desktop application. 2. Enter Device Name. 3. Enter **a valid** IP Address through mobile app. 4. Enter **a valid** Password through mobile app. 5. Click on the Pair button. Repeat the steps for the other desktop to be connected. Copy a file from one desktop to the other.	Successful connection to more than one desktop	-	-	Data is exchanged successfully	-	**Pass**
TS_04	Verify the autoconnection functionality	04-1	Verify autoconnection reliability after app restart.	1. Pair device and restart app. 2. Check for reconnection.	Device paired and connected	Paired device details	Device reconnects automatically	Device reconnects successfully after app restart	-	**Pass**

Chapter 9

<User Manual>

1. Installation Guide:
   a. Go to download project from here.
   b. You will find Desktop application and mobile application as shown in figure 8.1:

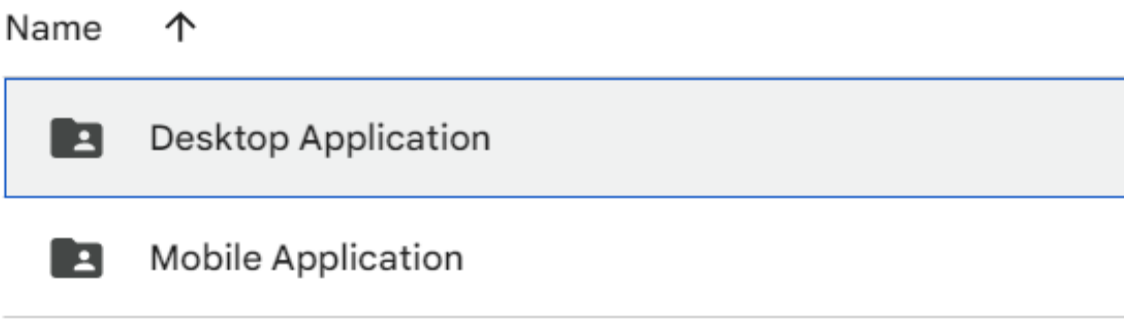


Figure 8.1

   c. Download Desktop Application folder to your pc as shown in figure 8.2 :

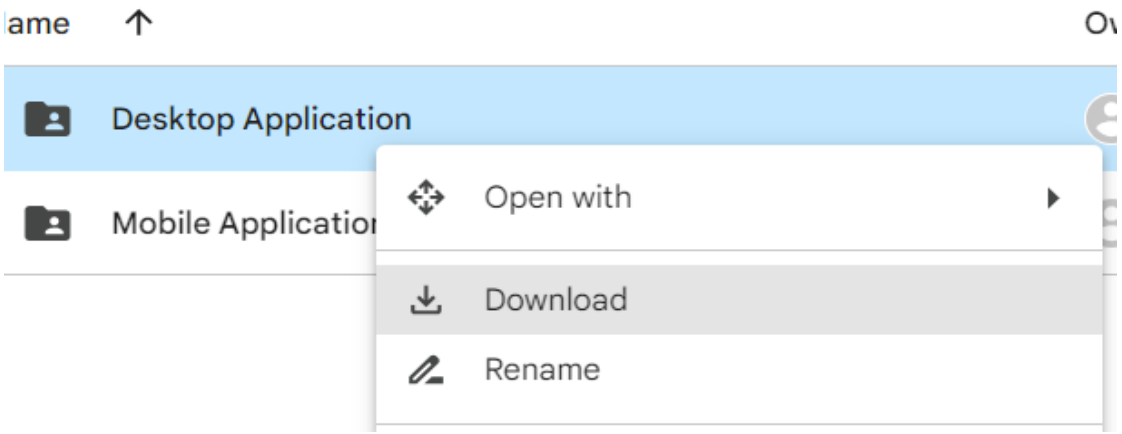


Figure 8.2

   d. Now after it's downloaded you will find this .zip file as shown in figure 8.3:

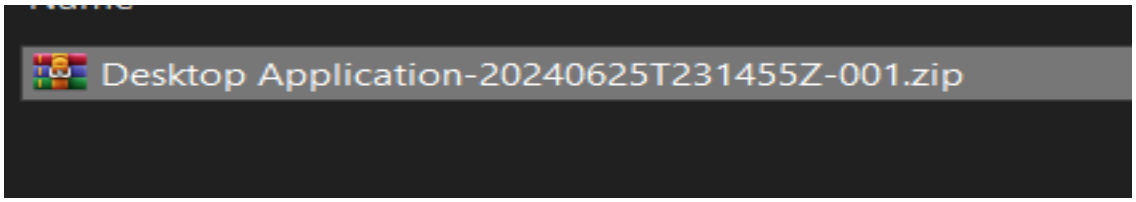


Figure 8.3

e. Extract it as shown in figure 8.4:

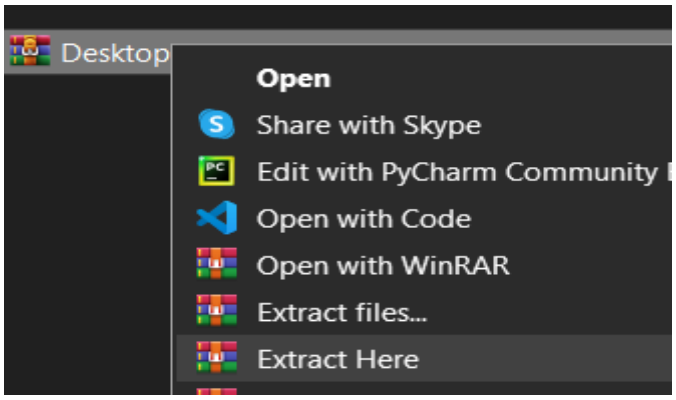


Figure 8.4

f. Then you will find inside the folder the following files as shown in figure 8.5:

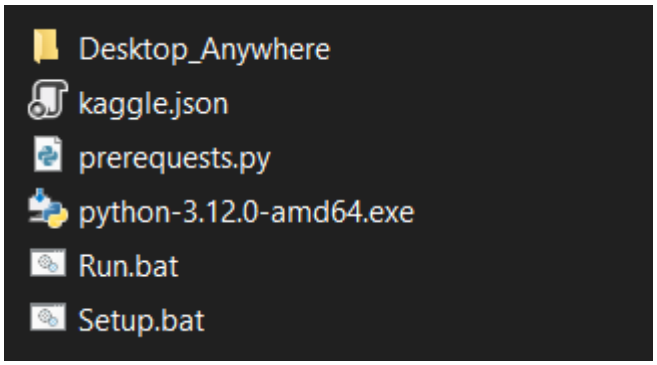


Figure 8.5

g. First run python-3.12.0-amd64.exe and check add python.exe to PATH as shown in figure 8.6:

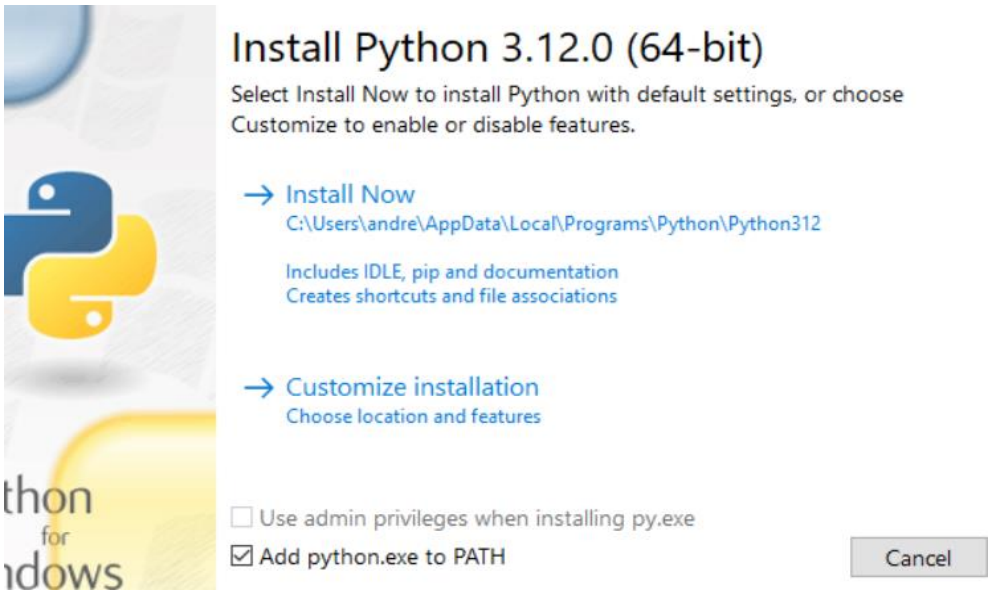


Figure 8.6

h. Then click install now and after it ends close.
i. Second you will find Kaggle.json file cut it and put it in the following path C:\Users\andre\.kaggle if you don't find folder called .kaggle create it the steps is shown here in figure 8.7:

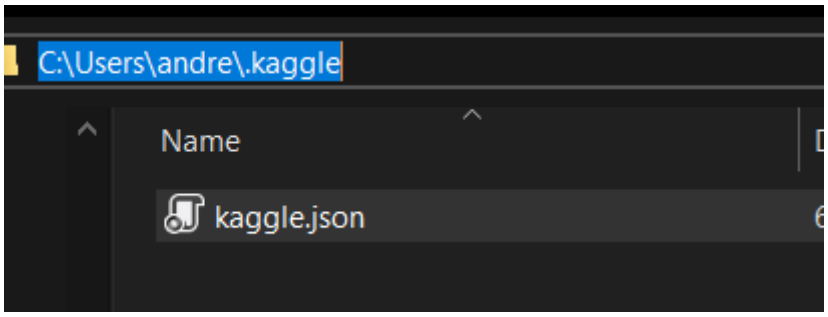


Figure 8.7

j. Third run Setup.bat and wait until it finishes the run as it's shown here in figure 8.8:

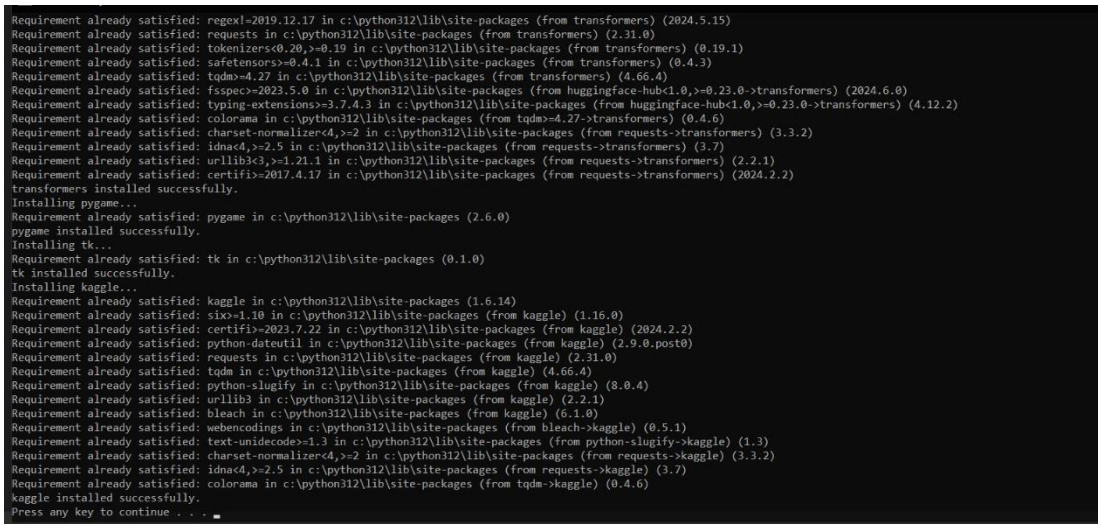


Figure 8.8

k. Last step run the desktop application using Run.bat every time you want to use it.
l. Now let's download mobile application in your mobile phone you will find desktop.apk file inside Mobile Application folder as shown here in figure 8.9:

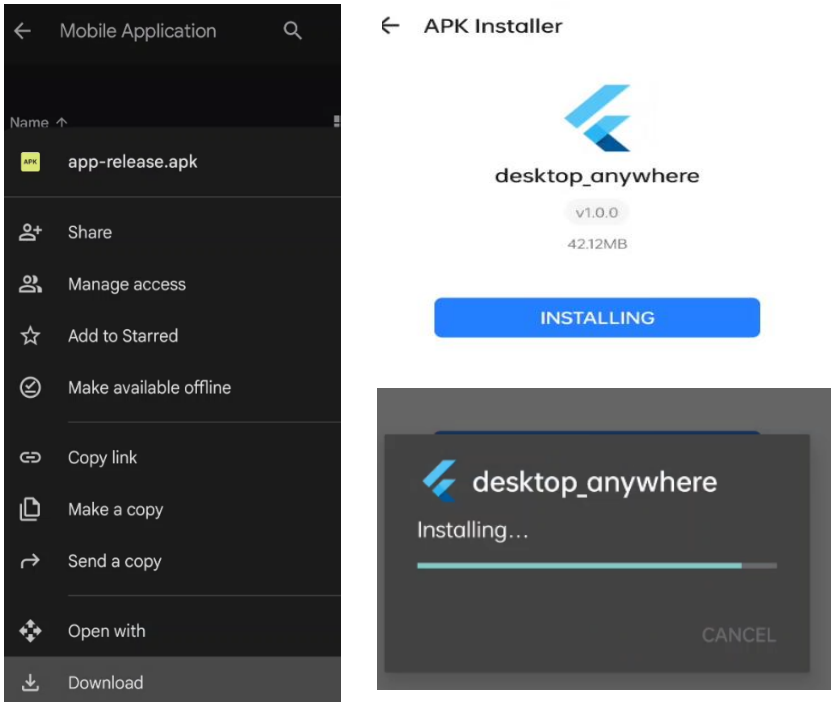


Figure 8.9

How each functionality works in the project

- Run the Run.bat file and wait until program starts and select which screen you want to share as shown in figure 8.10:

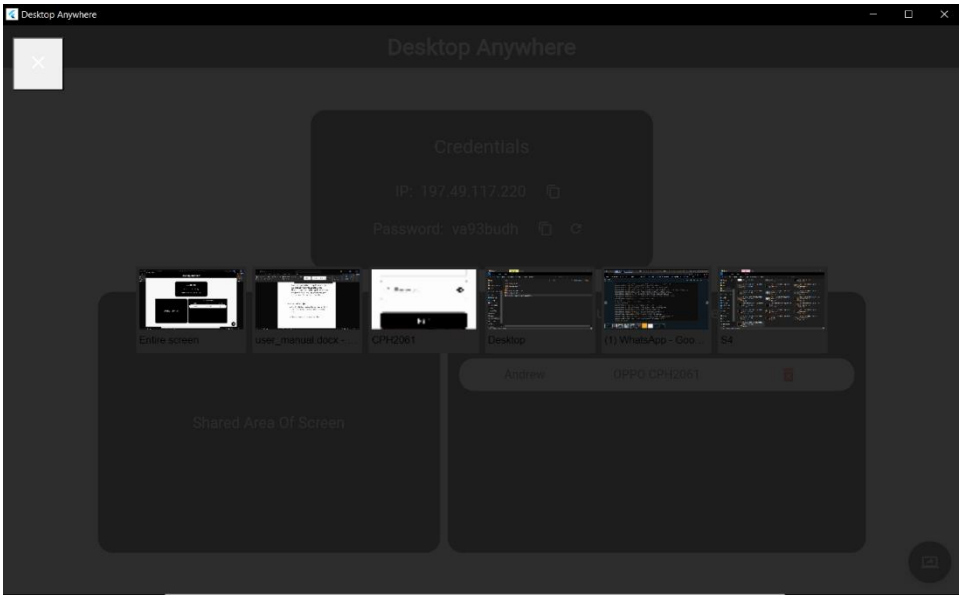


Figure 8.10

- Then open mobile application and press open then enter your Name as shown in figure 8.11:

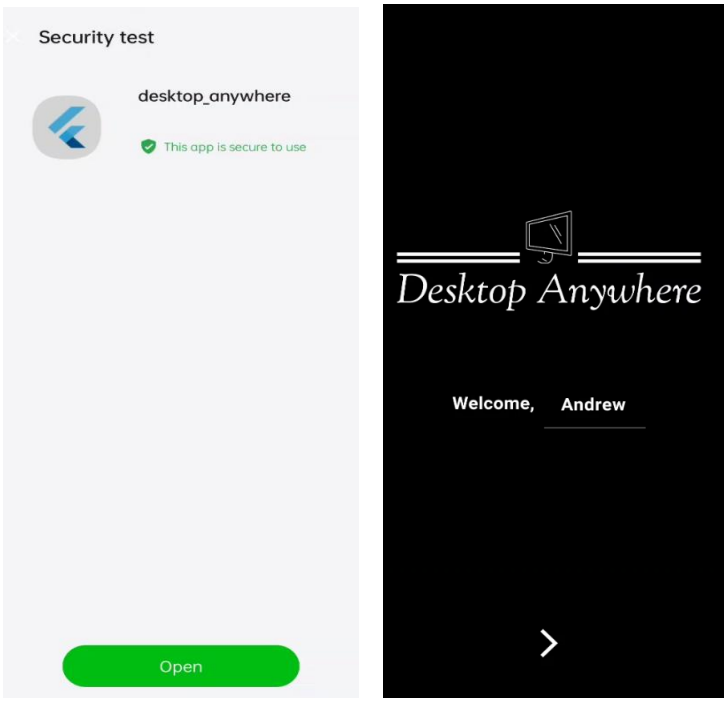


Figure8.11

- Now we in home page as shown in figure 8.12:
- To add device to manage it click the plus sign (+).

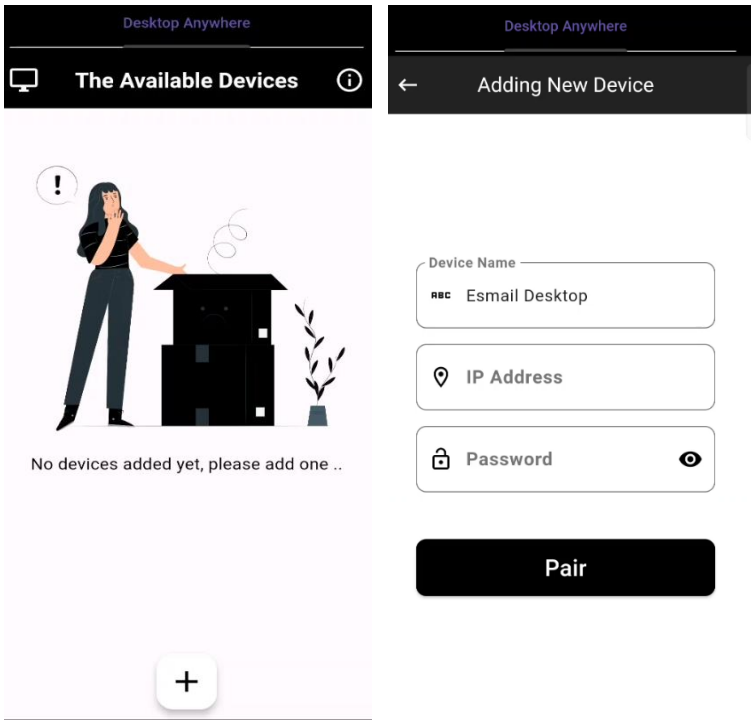


Figure8.12

- You can get IP Address and Password from desktop application:

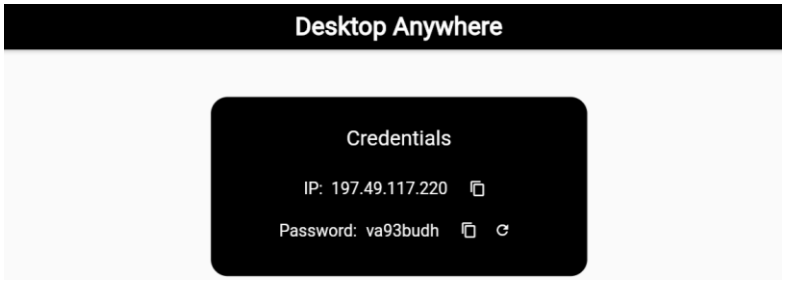


Figure 8.13

- After adding device we find it in mobile application as shown, note that you can add more than one in the same way.
- By pressing on the arrow in the right side of the device detailes card you can enter this device.

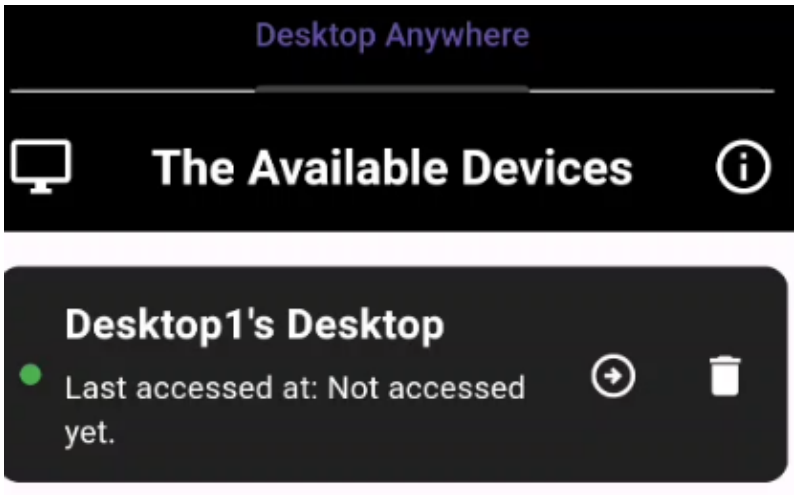


Figure 8.14

- that's what will be shown after this arrow is clicked

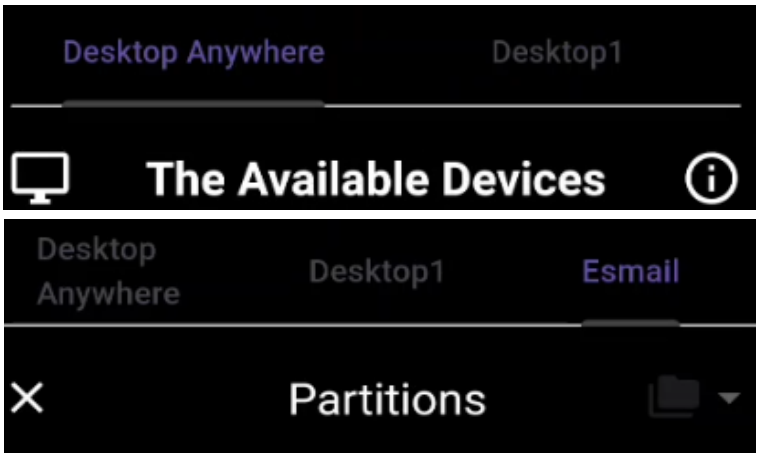


Figure 8.15

- Let’s press on our desktop named here as Desktop1:
- Now we see the partitions on the desktop and button to record voice commands and icon on the top right that make us access share screen or virtual control:
- Let’s try each one first try to access partition D for example let’s press on the arrow:

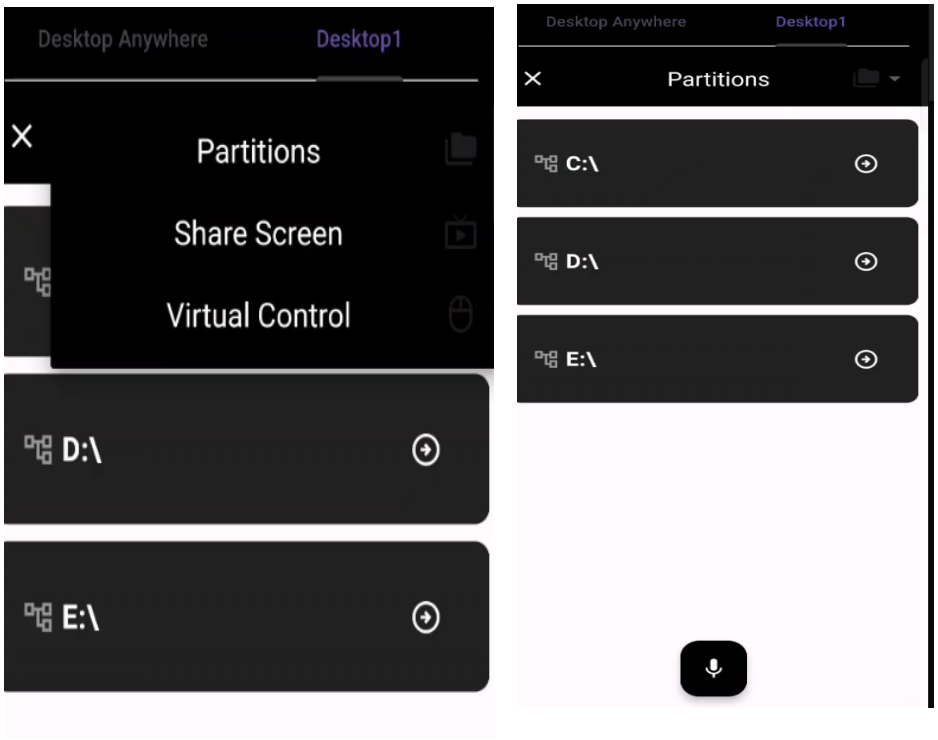


Figure 8.16

  - Now we see all folders and files inside it.

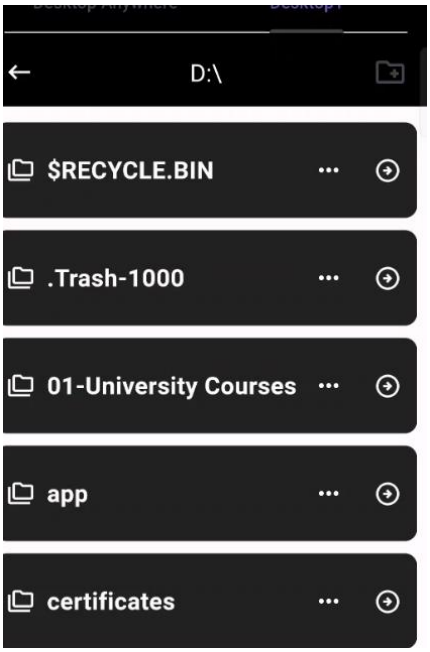


Figure 8.16

- Now we see (…) before arrow it provide us 3 functionality:
  - Save on mobile, Delete file from desktop or copy it to another folder in the same device or another device.
  - For folder it provides only Delete it.

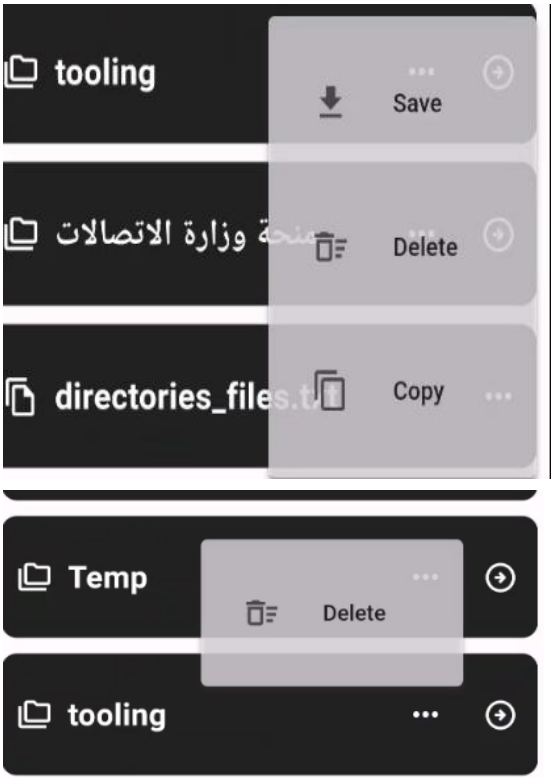


Figure 8.17

- Now let's try Share Screen when I press on Share Screen option from previous icon we show now we see live streaming of the window we selected to share from desktop application :
  - We can move our finger on shared screen to control mouse of desktop.
  - We have some buttons down share let's discuss them from left to right:
    - First one to refresh the streaming.
    - second one to send voice commands like that in partitions screen.
    - Third one to get virtual keyboard that any key we press it typed on desktop where cursor focused

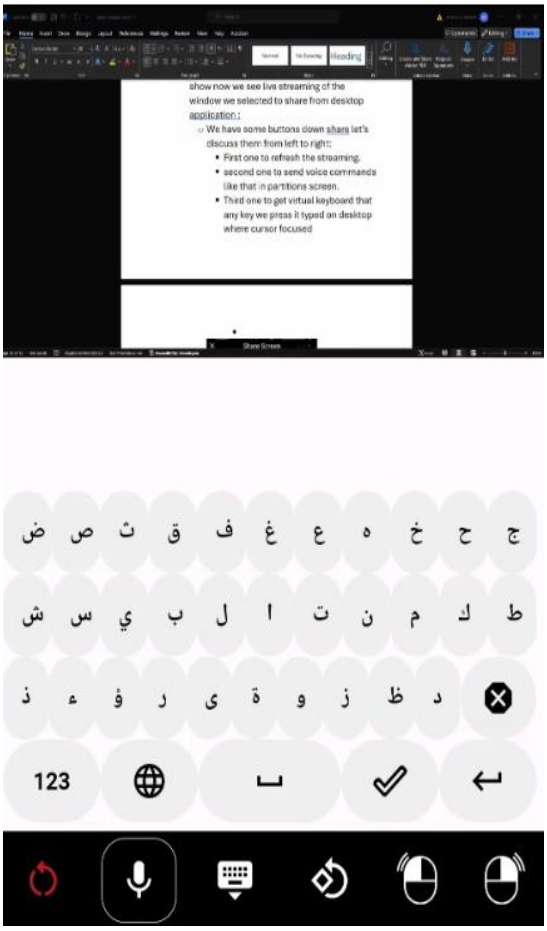


Figure 8.18

- Fourth one to rotate the screen
    - That give us two more buttons for zoom in and zoom out

Figure 8.19

- Fifth one for left click.
- Sixth one for right click.

- Now let’s try Virtual Control:
  - We find first half from screen is touchpad as I can control mouse of desktop from it.
  - Second half from screen has right, left click and virtual keyboard

Figure 8.19

- Now let’s try voice command button I press on it and it starts recording and I got two buttons one to cancel voice and another to send it to desktop to implement command

Figure 8.20

Chapter 10 <Results and Discussion>

1. **Results:**

Expected Results:

1.1. Collecting Voice Dataset for voice command to be executed on deskop that consists of Arabic and English command.

1.2. Collecting Dataset for Multi class classification in Arabic.

1.3. Collecting Dataset for NER Arabic.

1.4. Chose suitable models and Finetune them for the following tasks speech to text, classification and Name Entity Recognition to reach our goal which is getting accuracy 95% or higher.

1.5. Building Mobile app, Desktop app and server.

1.6. Deploy our server on free online server.

1.7. Building Command execution system.

Actual Results:

1.1. Voice Dataset collected.
You can find all used voice records through the following link: https://drive.google.com/drive/folders/1uuMxrbdnhS-MyFNL2u8Vopd7lHz8MIkV

1.2. Dataset for Multi class classification collected.

1.3. Dataset for NER Arabic collected.
You can find all generated text data in the following link: https://docs.google.com/spreadsheets/d/1l1NMW-FEDAkgBqRQfjB3KDp6vuhzJbLK/edit#gid=317260803

1.4. models for speech to text, classification and Name Entity Recognition has been chosen and finetuned depends on the following trials and previouslymentioned features of each chosen model.

1.4.1. Trails and final results:

1.4.1.1. Speech to Text Task: we chose Whisper model for this task we finetuned it with 800 records at first but, it didn't achieve our WER target so we increased it slightly with tests each time till we reached 1932 record which met our targeted accuracy, and the results we got from the first and final trail is displayed in the following table.

	Customized Dataset	Customized Dataset 1
SIZE	800	1932
SPLITS	Train: 640 Test: 160	Train: 1544 Test: 387
WER Before Fine-Tuning	19.8 %	28 %
WER After Fine-Tuning	7.69 %	5.3 %

1.4.1.2. Testing whisper on Arabic sentences only and on a sentences that is a mix of Arabic and English to ensure that the blending of languages doesn't cause an issue.

	Arabic	Mixed
Tested On	400	400
WER	20.61%	19.8%

1.4.1.3. Adding a period at the end of each sentence proved to achieve a better Word Error Rate (WER) than omitting it.

	With Period	Without Period
Size	1932	
Splits	Train: 1544 Test: 387	
WER	4.6%	5.3%

1.4.2. For classification and NER tasks we tried arabert and DistilBert, DistilBert was chosen as a competitor to Arabert which is the most used model in arabic tasks today, because this is a general model based on Bert but it’s not for specific language like arabert, so we can determine which is better for our data as we

mentioned before we use multiple languages in each command. In the following tables we will show the trials and the final results of both models.

1.4.2.1. Classification Task:

Trail 1 results:

	DistilBERT	AraBERT
Dataset	• Our customized dataset of 1405 row: • Train -> 1123 80% • Test -> 281 20%	
Accuracy Before Fine-Tuning	20 %	28 %
Accuracy After Fine-Tuning	95 %	97 %

Trail 2:

	Dataset-V1	Dataset-V2
Dataset Size	1405 row	1872 row
Splits	• Train -> 1123 • Test -> 281	• Train ->1497 • Test -> 375
Accuracy Before Fine-Tuning	40 %	25 %
Accuracy After Fine-Tuning	99 %	100 %

1.4.2.2. NER Task:

	DistilBERT	AraBERT
Dataset	• Our customized dataset of 1405 row: • Train -> 1123 80% • Test -> 281 20%	
Accuracy Before Fine-Tuning	20 %	28 %
Accuracy After Fine-Tuning	95 %	97 %

	Dataset-V1	Dataset-V2
Dataset Size	1405 row	1872 row
Splits	• Train -> 1123 • Test -> 281	• Train ->1497 • Test -> 375
Accuracy Before Fine-Tuning	28 %	43 %
Accuracy After Fine-Tuning	97 %	98 %

As is clearly shown in the tables Arabert proved that it's the perfect choice for this type of commands. Thus, we worked with Arabert.

1.5. Mobile app, Desktop app and server built.
1.6. Server deployed.
1.7. Command execution system built.

2. Discussion:
2.1. We managed to get the expected results but, Regarding the part of preparing the training dataset, we faced some challenges. We searched

extensively for an open-source dataset but couldn't find what we were looking for. Consequently, we built the dataset from scratch. It took a lot of time and effort to collect it, as we gathered the data from our colleagues.

2.2. In conclusion for the ai based part in our project and based on the results of the models which are illustrated in previous tables, we decided to use Whisper for speech to text which give us a WER: 4.6% after finetune, arabert-v2 for classification which give us an accuracy: 100% and arabert-v2 for NER task which give us an accuracy : 97.6%.

Chapter 11

<Conclusion and Recommendations>

Conclusion

Eventually, in this project we tried to bridge the gap between physical presence and desktop accessibility. Through Desktop Anywhere, a comprehensive system designed for users to be user-friendly/ the mobile application empowers users to access and control their personal desktops from anywhere through the server. This remote access capability simplifies file management by allowing users to effortlessly manage and navigate through files and resources across multiple desktops concurrently.

Desktop Anywhere embraces inclusivity by providing accessibility features like voice commands (currently in Arabic) that cater to a wider range of users. For individuals who prefer hands-free interaction or those engaged in multitasking activities, Desktop Anywhere provides voice control and virtual input methods for executing tasks and controlling applications. Repetitive tasks like shutdowns or application launches can be automated, streamlining daily routines and boosting efficiency.

The ability to manage tasks effectively is crucial in today's fast-paced world. Desktop Anywhere addresses this need by offering task scheduling, allowing users to organize activities seamlessly across both desktop and mobile devices.

Recommendation for Future Work

Our project is a comprehensive software system which gives the user a seamless experience in accessing his desktop devices through a handy portable device, his Personal Mobile, at any place. Working on Improving this system to give users better experience is a must. Thus, we will outline all the recommendations we have for improving what we have so far.

- **New System Versions:** As our system now can work between devices having Android and windows as their operating systems only. We recommend creating a Version of this system to be compatible with Mac, Linux, and IOS.
- **Advanced Voice Commands:** Instead of enabling LIMITED voice commands to be performed through the system, giving the user a full access through the voice commands is going to be an Important and effective improvement.
- **Context-Aware Recommendations:** Leverage machine learning (ML) to personalize the user experience. The mobile app could recommend frequently used desktop applications based on time, location, or user activity. This would streamline workflow and save time spent searching for specific programs.
- **Offline Functionality:** Enable basic offline functionality within the mobile app. Users could access and edit local copies of previously downloaded documents on the mobile device when an internet connection is unavailable. Synchronization with the desktop PC would occur when online again.

These recommendations leverage advancements in System Compatibility, System Features, and ML to further elevate Desktop Anywhere's user experience and functionality but we all have to continuously researching emerging technologies as it can open doors for even more innovative features in the future.

Tools

Tools used in the project: -

- **Environment:** Widows 10 and 11 .64-bit, anaconda 2023.09, Python 3.10, Android.
- **Tools:**
  - Visual Studio Code.
  - Android studio.
  - Anaconda.
  - Kaggle.
  - Colab.
- **Programming Languages and technologies:**
  - Python.
  - Firebase
  - Express JS.
  - Node js.
  - Socket IO.
  - Flutter.
  - web RTC.
  - MongoDB.
  - **Python Libraries:** transformers, socket, subprocess, torch, os, requests.

References

[1] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision," *arXiv:2212.04356 [cs, eess]*, Dec. 2022, Available: https://arxiv.org/abs/2212.04356

[2] Abdelhalim Hafedh Dahou, Mohamed Amine Cheragui, and A. Abdelali, "Performance Analysis of Arabic Pre-Trained Models on Named Entity Recognition Task," Jan. 2023, doi: https://doi.org/10.26615/978-954-452-092-2_051.

[3] W. Antoun, F. Baly, and H. Hajj, "AraBERT: Transformer-based Model for Arabic Language Understanding," arXiv:2003.00104 [cs], Mar. 2020, Available: https://arxiv.org/abs/2003.00104

[4] WebRTC, "WebRTC Home | WebRTC," Webrtc.org, 2017. https://webrtc.org/

[5] V. Tiwari, M. F. Hashmi, A. Keskar, and N. C. Shivaprakash, "Virtual home assistant for voice based controlling and scheduling with short speech speaker identification," Multimedia Tools and Applications, Jul. 2018, doi: https://doi.org/10.1007/s11042-018-6358-x.

[6] F.-C. Cheng, "Automatic and Secure Wi-Fi Connection Mechanisms for IoT End-Devices and Gateways," Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 98–106, 2018, doi: https://doi.org/10.1007/978-3-319-95450-9_8.

[7] "DistilBERT," huggingface.co. https://huggingface.co/docs/transformers/model_doc/distilbert#usage-tips