MINOR PROJECT

Speech Emotion Recognition

Submitted in partial fulfilment of the requirements
for the award of the degree of

Bachelor of Technology
Computer Science and Engineering

Guide(s):
Mr. Anil Kumar Vats

Submitted by:
Abhishek Sharma - 41113302717
Parth - 41013302717
Kanishk Gupta -41713302717

(CSE-7B)

HMR INSTITUTE OF TECHNOLOGY & MANAGEMENT

HAMIDPUR, DELHI 110036

Affiliated to

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY

Sector - 16C Dwarka, Delhi - 110075, India

2017 - 21

DECLARATION

We, student(s) of B.Tech Computer Science hereby declare that the minor project entitled "**SPEECH EMOTION RECOGNITION**" which is submitted to Department of Computer science Engineering, HMR Institute of Technology & Management, Hamidpur Delhi, affiliated to Guru Gobind Singh Indraprastha University, Dwarka(New Delhi) in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in CSE, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition. The list of members involved in the project is listed below-

S No.	Name	Roll No.	Signature
1	**Abhishek Sharma**	**41113302717**	
2.	**Parth**	**41013302717**	
3.	**Kanishk Gupta**	**41713302717**	

This is to certify that the above statement made by the candidate(s) is correct to the best of my knowledge.

Place: Delhi Date: 15/12/2020

Dr. Mohd. Izhar
(Head of the Department, CSE)

Mr. Anil Kumar Vats
(Assistant Professor, CSE)

HMRITM, New Delhi

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of this project would be incomplete without the mention of the people who made it possible, without whose constant guidance and encouragement would have made efforts go in vain. We consider ourselves privileged to express gratitude and respect towards all those who guided us through the completion of this project. Firstly, it gives us immense pleasure to acknowledge our institute **HMR Institute of Technology and Management** for providing us with an opportunity to develop a project on **Speech Emotion Recognition.**

In addition, we wish to express our deep sense of gratitude to **Dr. VC Pandey, Director**, **HMRITM** for permitting us to carry out this project work and for his guidance and support. We would also like to thank our guide **Mr. Anil Kumar Vats** for guiding us for the project and motivating us throughout its making. Last but not the least, we extend our whole-hearted gratitude for the invaluable contribution of our parents for their blessings and earnest affection and all those people 'behind the veil' for their necessary support, which enabled us to complete this project.

ABSTRACT

Human-Machine interaction through audio or speech is always challenging because of the difficulties in emotion detection from audio data. Emotion is a natural way to express an individual's mental state. When a person is in an unusual mental state, the voice changes accordingly in his or her subconsciousness. Speech Emotion Recognition is a technology that extracts emotional features from speech signals by computer and contrasts and analyses the characteristic parameters and the emotional change acquired. Many systems have been developed to identify the emotions from the speech signal. Emotion recognition is based on the technologies which use different classifiers for emotion recognition. The classifiers differentiate emotions such as anger, happiness, sadness, surprise, neutral state, etc. The dataset for the speech emotion recognition system is the emotional speech samples and the features extracted from these speech samples. There are many voice products developed like Amazon Alex, Google Home, Apple Home Pod, which functions mainly on voice-based commands. It is evident that voice will be the better medium for communicating with the machines. In this project, we have created a speech emotion recognizer that identifies the speaker's emotions through his voice.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 - INTRODUCTION

1.1 GENERAL

Speech Emotion Recognition (SER) is the task of recognizing the emotional aspects of speech irrespective of the semantic contents. While humans can efficiently perform this task as a natural part of speech communication, the ability to conduct it automatically using programmable devices is still an ongoing subject of research. Studies of automatic emotion recognition systems aim to create efficient, real-time methods of detecting the emotions of mobile phone users, call center operators and customers, car drivers, pilots, and many other human-machine communication users. Adding emotions to machines has been recognized as a critical factor in making machines appear and act in a human-like manner.

The ability to understand people through spoken language is a skill that many human beings take for granted. On the contrary, the same task is not as easy for machines, as consequences of a large number of variables which vary the speaking sound wave while people are talking to each other. A sub-task of speech understanding is about the detection of the emotions elicited by the speaker while talking, and this is the main focus of our contribution.

Robots capable of understanding emotions could provide appropriate emotional responses and exhibit emotional personalities. In some circumstances, humans could be replaced by computer-generated characters having the ability to conduct very natural and convincing conversations by appealing to human emotions. Machines need to understand emotions

conveyed by speech. Only with this capability, an entirely meaningful dialogue based on mutual human-machine trust and understanding can be achieved.

Speech emotion recognition is an important problem receiving increasing interest from researchers due to its numerous applications, such as audio surveillance, E-learning, clinical studies, detection of lies, entertainment, computer games, and call centres. Nevertheless, this problem still remains a significantly challenging task for advanced machine learning techniques.

One of the reasons for such a moderate performance is the uncertainty of choosing the right features.

In addition, the existence of background noise in audio recordings, such as real-world voices, could dramatically affect the effectiveness of a machine learning model. Nevertheless, the advent of decent emotional speech recognition models could significantly improve the user experience in systems involving human-machine interactions. Indeed, the ability to recognize emotions from audio samples and, therefore, the ability to imitate these emotions could have a considerable impact on the field of AI.

One of the main advantages of deep learning techniques lies in the automatic selection of features, which could, for example, be applied to important attributes inherent to sound files having a particular emotion in the task of recognition of speech emotions. In recent years, various models based on deep neural networks for speech emotion recognition have been introduced.

1.2 OVERVIEW

There are two types of features in a speech namely, the lexical features (the vocabulary used) and the acoustic features (sound properties like pitch, tone, jitter, etc.).

The problem of speech emotion recognition can be solved by analyzing one or more of these features. Choosing to follow the lexical features would require a transcript of the speech which would further require an additional step of text extraction from speech if one wants to predict emotions from real-time audio. The analysis of the acoustic features can be done in real-time while the conversation is taking place as we'd just need the audio data for accomplishing our task. Hence, we choose to analyze the acoustic features in this work.

Furthermore, the representation of emotions can be done in two ways:

- Discrete Classification: Classifying emotions in discrete labels like anger, happiness, boredom, etc.
- Dimensional Representation: Representing emotions with dimensions such as Valence (on a negative to positive scale), Activation, Energy (on a low to high scale) and Dominance (on an active to passive scale)

1.3 LITERATURE SURVEY

In this field of research, many classification strategies have been presented in the last years. One of the systems, proposed by Iqbal et al. used Gradient Boosting, KNN and SVM to work on a granular classification on the RAVDESS dataset used in this work to identify differences based on gender with approximately between 40% and 80% overall accuracy, depending on the specific task.

Over the last years, excessive investigation has been completed to recognize emotions by using speech statistics. Cao et al. proposed a ranking SVM method for synthesizing information about emotion recognition to solve the problem of binary classification. This ranking method, instructs SVM algorithms for particular emotions, treating data from every utterer as a distinct query then mixing all predictions from rankers to apply multi-class prediction. Ranking SVM achieves two advantages, first, for training and testing steps in speaker-independent it obtains speaker-specific data. Second, it considers the intuition that each speaker may express mixed emotion to recognize the dominant emotion. Ranking approaches achieve a substantial gain in terms of accuracy compared to conventional SVM in two public datasets of acted emotional speech, Berlin and LDC. In both acted data and the spontaneous data, which comprises neutral intense emotional utterances, ranking-based SVM achieved higher 020105-2 accuracy in recognizing emotional utterances than conventional SVM methods. Unweighted average (UA) or Balance accuracy achieved 44.4%. Chen et al. aimed to improve speech emotion recognition in speakers-independent with three-level speech emotion recognition method. This method classifies different emotions from coarse to fine then select appropriate features by using Fisher rate. The output of Fisher rate is an input parameter for multi-level SVM based classifiers.

Wu et al. proposed a new modulation spectral feature (MSFs) human speech emotion recognition. Appropriate feature extracted from an auditory-inspired long-term spectro-temporal by utilizing a modulation filterbank and an auditory filterbank for speech decomposition. This method obtained acoustic frequency and temporal modulation frequency components to convey important data which is missing from traditional short-

term spectral features. For the classification process, SVM with radial basis function (RBF) is adopted. Berlin and Vera am Mittag (VAM) are employed to evaluate MSFs. In experimental results, the MSFs display capable performance in comparison with MFCC and perceptual linear prediction coefficients (PLPC). When MSFs utilize augment prosodic features, there is a considerable improvement in the performance of recognitions.

1.4 PROBLEM STATEMENT

Speech Emotion Recognition (SER) is the task of recognizing the emotional aspects of speech irrespective of the semantic contents. The ability to understand people through spoken language is a skill that many human beings take for granted. On the contrary, the same task is not as easy for machines, as consequences of a large number of variables which vary the speaking sound wave while people are talking to each other. The task is to build an end-to-end hardware-software speech emotion recognition that recognizes the emotions of the speaker through voice and provides a suitable response.

1.5 SCOPE OF STUDY

Studies of automatic emotion recognition systems aim to create efficient, real-time methods of detecting the emotions of mobile phone users, call centre operators and customers, car drivers, pilots, and many other human-machine communication users.

Robots capable of understanding emotions could provide appropriate emotional responses and exhibit emotional personalities. In some circumstances, humans could be replaced by computer-generated characters having the ability to conduct very natural and convincing conversations by appealing to human emotions. Various virtual assistants in the field of health after using such models can significantly improve their performance.

Methods based on the Fourier transform such as MFCC and MS are the most used in speech emotion recognition. However, their popularity and effectiveness have a downside. It has led to a very specific and limited view of frequency in the context of signal processing. Simply put, frequencies, in the context of Fourier methods, are just a collection of the individual frequencies of periodic signals that a given signal is composed of. To use methods that provide an alternative interpretation of frequency and an alternative view of non-linear and non-stationary phenomena is our future work. More work is needed to improve the system so that it can be better used in real-time speech emotion recognition.

CHAPTER 2 - SYSTEM ANALYSIS

2.1 GENERAL

Software Engineers have been trying various tools, methods and procedures to control the process of software development in order to build high-quality software with high productivity. This method provides “how it is” for building the software while the tools provide automated or semi-automated support for the methods. They are used in all stages of the software development process, namely, planning, analysis, design, development and maintenance. The software development procedure integrates the methods and tools together and enables rational and timely development of the software system.

Systems development is a systematic process which includes phases such as planning, analysis, design, deployment, and maintenance, we will primarily focus on −

1. System Analysis:

System Analysis is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components. System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem-solving technique that improves the system and ensures that 7 all the components of the system work efficiently to accomplish its purpose. They provide the guidelines as to how to apply these methods and tools, how to produce the deliverable at each stage, what controls to apply, and what milestones to use to access the performance of the program. There exist a number of software development paradigms each using a different set of methods and tools. The selection of a particular paradigm depends on the nature of the application of the programming language used for the controls and the deliverables

required. The development of such successful systems depends not only on the use of appropriate methods and techniques but also the developers' commitment to the objective of the system. A successful system must:

1. Satisfy the user requirement
2. Be easy to understand by user and operator
3. Be easy to operate
4. Have a good user interface
5. Be easy to modify
6. Be expandable
7. Have adequate security control against the misuse of data
8. Handle the errors and exceptions satisfactorily
9. Be delivered on schedule within the budget

Within this analysis phase, the analyst is discovering and fact-finding. Along with meeting with stakeholders, the analyst must meet with end-users to understand what the user's needs are and to learn about problems that affect the current system in order to assist with designing a new and more efficient system. There are several activities that must occur within the analysis phase:

1. Gather Information
2. Define the new system's requirements
3. Build prototypes for the new system
4. Prioritize requirements
5. Evaluate alternatives

6. Meet with management to discuss new options

2. System Design:

It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, you need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently.

System Design focuses on how to accomplish the objective of the system. System Analysis and Design (SAD) mainly focuses on −

1. Systems
2. Processes
3. Technology

Systems Analysis and Design is an active field in which analysts repetitively learn new approaches and different techniques for building the system more effectively and efficiently. The primary objective of systems analysis and design is to improve organizational systems. This tutorial provides a basic understanding of system characteristics, system design, and development processes. It is a good introductory guide that provides an overview of all the concepts necessary to build a system. System design is the most creative phase of system development. The term describes a final system and the process by which it is developed. The question in system design is: How the problem is to be solved?

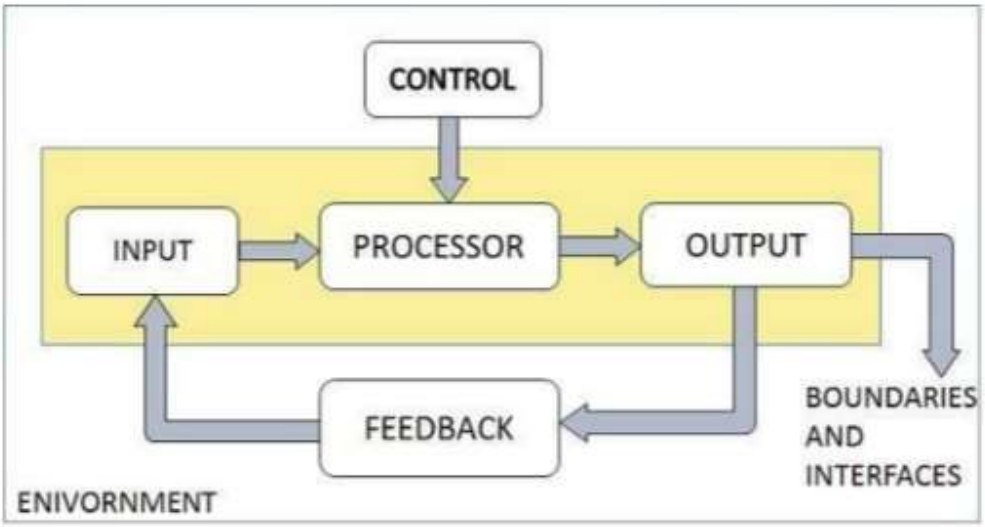

Fig 2. 1:Systems Analysis and Design

A systematic method has to achieve beneficial results in the end. It involves starting with a vague idea and developing it into a series of steps. The series of steps for successful system design are:

1. The first step is to study the problem completely because we should know about the goal. We should see what kind of output we require and what kind of input we give so that we can get the desired result. We should see what kind of program should be developed to reach the final goal.
2. Then we write individual programs, which later on joining solve the specified problem.
3. Then we test these programs and make necessary corrections to achieve the target of the programs.

While designing we had to consider all the requirements of the users to make such an interface which makes the communication easy and accurate. The interface is designed to provide efficient and clear information of each and every detail of the application.

2.2 PRELIMINARY INVESTIGATION

The main aim of preliminary analysis is to identify the problem. First, the need for the new or the enhanced system is established. Only after the recognition of need, for the proposed system is done then further analysis is possible. In this investigation, we researched about the working of Django application and integrating it with the ML model to generate the result onto the Arduino. We investigated how the implementation and server will be built to control the full application and to predict.

2.3 FEASIBILITY STUDY

A feasibility study is undertaken to determine the possibility or probability of either improving the existing system or developing a completely new system. It helps to obtain an overview of the problem and to get a rough assessment of whether a feasible solution exists or not. There are three aspects of the feasibility study portion of the preliminary investigation.

As the name implies, a feasibility analysis is used to determine the viability of an idea, such as ensuring a project is legally and technically feasible as well as economically justifiable. It tells us whether a project is worth the investment—in some cases, a project may not be doable. There can be many reasons for this, including requiring too many

resources, which not only prevents those resources from performing other tasks but also may cost more than an organization would earn back by taking on a project that isn't profitable.

A well-designed study should offer a historical background of the business or project, such as a description of the product or service, accounting statements, details of operations and management, legal requirements. Generally, such studies precede technical development and project implementation, such are given below:

1. Is there any new and better way to do the job that solves the problem?
2. What are the costs and savings of the alternatives?
3. Are there any legal restrictions imposed by the government or any other regulatory body?
4. Is the proposed solution economically feasible?
5. Does there exist any bottleneck that may turn the process of development futile exercise?
6. What is recommended?

Although this project has very good feasibility due to required fewer resources to complete this project, where the requirement features are given in the software and hardware specification unit.

Five Areas of Project Feasibility

A feasibility analysis evaluates the project's potential for success; therefore, perceiving objectivity is an essential factor in the credibility of the study for potential investors and lending institutions.

1. Economical feasibility
2. Legal feasibility
3. Technical feasibility
4. Operational feasibility
5. Behavioural feasibility

There are three types of feasibility study separated areas that a feasibility study examines.

1. Technical Feasibility
2. Economic Feasibility
3. Operational Feasibility

2.3.1 TECHNICAL FEASIBILITY

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems.

Technical feasibility also involves evaluation of the hardware, software, and other technical requirements of the proposed system. Organization objective is to serve the plan to create a standard application which addresses the dual problems like availability of information and format conversions. The proposed system will fulfil both objectives because the use of web sites provides security.

Speech Emotion Recognition is the best application in the manner of technical feasibility because the technical resources required for this project is very less. This project involves Python language & Arduino Hardware just to be implemented, which is performed on

PyCharm. PyCharm is the type of open-source application where we can implement python programming language easily. For the user interface, we have used the Django Framework which is a python library. Technical feasibility centres on an existing computer system (hardware and software) etc. and to what extent it can support the proposed edition. It is technically feasible since the whole system is designed into technologies like Python which are the most recent technologies to develop windows-based systems.

2.3.2 ECONOMICAL FEASIBILITY

The tools that will be used for the system are the latest one and thus the cost involved in tools, designing and developing the system will be a good investment for the organization. The benefits of using the system are not in monetary terms, but its increased interaction between Users & administrators. The modules designed can be easily navigated. The hardware available with the company is already the best available and hence no new purchase is required. This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility — helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide.

All of the above are free of cost and Eco-friendly in nature as well as all of the above are open source so that anyone can use it. For the user interface, we have used Django which is a python framework and it is free of cost just need to download. And for Arduino

connectivity, we have used PyFirmata library which is also free of cost. Rest of these we have used the platform named as PyCharm which is also free of cost and open-source environment.

2.3.3 OPERATIONAL FEASIBILITY

The systems analyst must still consider the operational feasibility of the requested project. Operational feasibility is dependent on human resources available for the project and involves projecting whether the system will operate and be used once it is installed. The proposed system provides functionality based on user characteristics to End-user or authorized user. This assessment involves undertaking a study to analyse and determine whether—and how well—the organization's needs can be met by completing the project. Operational feasibility studies also examine how a project plan satisfies the requirements identified in the requirements analysis phase of system development. Once it is determined that the system is both technically and economically feasible then it has to be seen if it is operationally feasible. Operational feasibility refers to projecting whether the system will operate and be used once it is installed. The proposed system is operationally feasible, as there is no need for the faculty to keep a check on the use of the internet at the time of execution. There is no need to keep a manual eye on the behaviour of the client on the internet. Site blocking, service blocking (FTP/HTTP) and content filtering features of the proposed system cater to this very task. The end users can easily understand and expand it in the future. As the faculty members expressed the need for an improved system, they put in all efforts to see that it becomes feasible.

2.4 SOFTWARE & HARDWARE SPECIFICATIONS

Software (Applications & Libraries):

- Python
- Django
- Flask
- Keras
- NumPy
- Pandas
- Scilab
- Tensor flow
- Arduino Cloud
- PyFirmata
- LiquidCrystal
- Google Cloud

Hardwares

- System (Min 2 GB RAM, Min 4 GB Memory Space)
- Arduino UNO
- Potentiometer
- Wires
- Breadboard
- LCD Display(16x2 Hitachi HD44780)
- USB Cable Type B
- Resistors 2k Ohm

2.5 DATA FLOW DIAGRAM

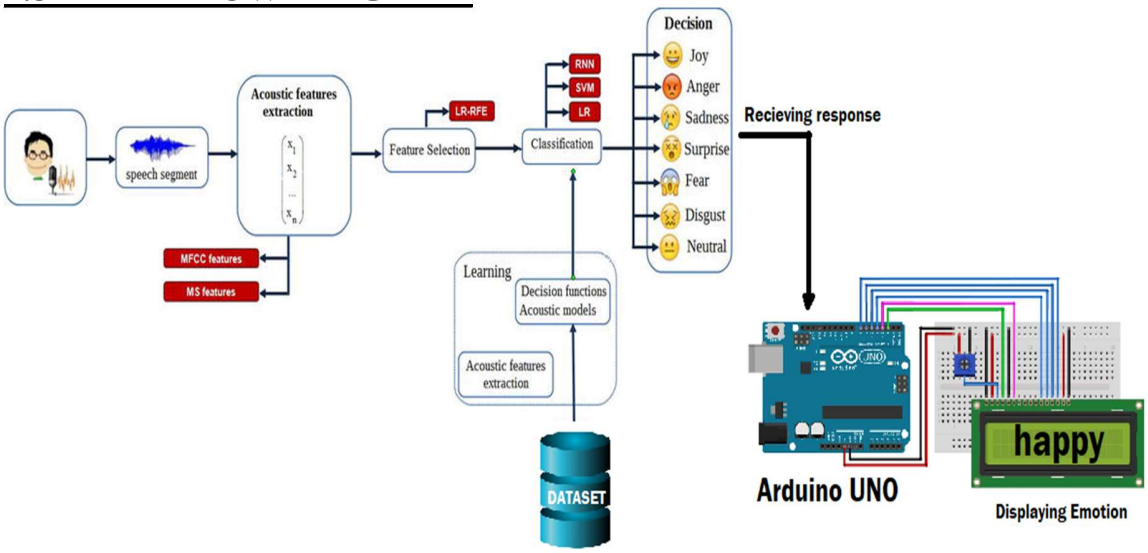

Fig 2. 2:Data Flow Diagram SER

CHAPTER - 3 SYSTEM DESIGN

3.1 DESIGN METHODOLOGY

A design methodology encapsulates various phases each containing some stages, which guide the designer in the techniques suitable at each stage of the project. A design methodology also helps the designer to plan, manage, control, and evaluate database development and managing projects. Furthermore, it is a planned approach for analysing and modelling a group of requirements for a database in a standardized and ordered manner.

i) Conceptual Database Design:

In this design methodology, the process of constructing a model of the data is used in an enterprise, independent of all physical considerations. The conceptual database design phase starts with the formation of a conceptual data model of the enterprise that is entirely independent of implementation details such as the target DBMS, use of application programs, programming languages used, hardware platform, performance issues or any other physical deliberations.

ii) Logical Database Design:

A local logical data model is used to characterize the data requirements of one or more but not all user views of a database, and a universal logical data model represents the data requirements for all user views. The final step of the logical database design phase is to reflect on how well the model can support possible future developments for the database system.

iii) Physical Database Design:

This physical methodology is the third and final phase of the database design methodology. Here, the designer must decide how to translate the logical database design (i.e., the entities, attributes, relationships, and constraints) into a physical database design which can ultimately be implemented using the target DBMS. As the various parts of physical database design are highly reliant on the target DBMS, there may be more than one method of implementing any given portion of the database.

Consequently, to do this work appropriately, the designers must be fully aware of the functionality of the target DBMS and must recognize the advantages and disadvantages of each alternative approach for a particular accomplishment. For some systems, the designer may also need to select a suitable storage space/strategy that can take account of intended database usage.

Physical database design is the process of making a description of the execution of the database on secondary storage which describes the base relations, file organizations as well as indexes used to gain efficient access to the data and any associated integrity constraints and security measures.

The emotion recognition system can be divided into two parts: the 1st part is emotional classier used for classification of emotion from a sound recording and the second part of the system is the emotion database intended for learning and training of emotional classier. Emotional classier is a very important component of the emotion recognition system, and it is also the core of the system. Three parts of neural classier are shortly described. The 1st part describes the process of sound sample preparation, the second part discusses feature extraction from a sample, and the last part of emotional classier describes a specific

type of neural networks and the way how these networks work. In the second part, we describe a tool for subjective evaluation. It describes certain parts of the tool and also the process of evaluation altogether with the processing of evaluation results. The last subchapter is dedicated to the future vision of a complex automated emotion recognition system and its use.

Fig 3. 1:Feature extraction Process

3.2 USER INTERFACE DESIGN

What is user interface design?

User interface design or UI design generally refers to the visual layout of the elements that a user might interact with in a website, or technological product. This could be the control buttons of a radio, or the visual layout of a webpage. User interface designs must not only be attractive to potential users, but must also be functional and created with users in mind.

Why is user interface design important for usability?

User interface design can dramatically affect the usability and user experience of an application. If a user interface design is too complex or not adapted to targeted users, the user may not be able to find the information or service they are looking for. In 31website design, this can affect conversion rates. The layout of a user interface design should also be clearly set out for users so that elements can be found in a logical position by the user.

How to optimize user interface design?

User interface designs should be optimized so that the user can operate an application as quickly and easily as possible. Many experts believe that UI design should be simple and intuitive, often using metaphors from non-computer systems. With a more intuitive user interface design, users will be able to navigate around a website easily, finding the product or service they want quickly. One way to check the intuitiveness of a user interface design is through usability testing. The feedback from usability testing can then be used to optimize the user interface design of a prototype or final product.

Project User Interface

Our whole User Interface (UI) / UX is made with pure HTML5, CSS, JavaScript, jQuery. For the backend we are using the Django framework to render content as the HTML. This framework helps us in structuring the whole UI which makes it easier for the programmers to code and handle any false & broken request to the server. UI is developed in keeping in mind the user intractability and user friendly. Our UI is able to handle any error and broken request from the Arduino or from the ML model. Every request made from the UI is asynchronous which makes the user experience more convenient and quicker. Our User Interface also displays proper errors to the user to make them clear about its functioning. Discussing the overall functioning of UI, it sends 5 AJAX calls when triggered on the recognize emotion button which makes the request to store uploaded or recorded voice in the storage (drive), then it makes a request to the ML model for fetching emotion passing the audio file path, after loading and running the model we get the emotion label (JSON obj) which further make another request to

Arduino UNO to display the emotion label on LCD and waits for the success message from it and at last after confirmation from arduino it displays the result to the UI. Also a request is made to visualize recorded or uploaded audio files to the frontend. Below are the technologies explained used in UI.

DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

It is primarily used to create web applications framework and control the whole application with its help.

Flask

Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Flask's framework is more explicit than Django's framework and is also easier to learn because it has less base code to implement a simple web-Application. A Web-Application Framework or Web Framework is the collection of modules and libraries that helps the developer to write applications without writing the low-level codes such as protocols, thread management, etc. Flask is based on WSGI (Web Server Gateway Interface) toolkit and Jinja2 template engine. Python 2.6 or higher is required for the installation of the Flask. You can start by importing Flask from the flask package on any python IDE. When the homepage of the web server is opened in the browser, the output of this function will be rendered accordingly.

However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Applications that use the Flask framework include Pinterest and LinkedIn. The Flask application is started by calling the `run()` function. The method should be restarted manually for any change in the code. To overcome this, the debug support is enabled so as to track any error.

JavaScript

JavaScript often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it for client-side page behavior, and all major web browsers have a dedicated JavaScript engine to execute it.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM). However, the language itself does not include any input/output (I/O), such as networking, storage, or graphics facilities, as the host environment (usually a web browser) provides those APIs.

AJAX

Ajax is a set of web development techniques using many web technologies on the client side to create asynchronous web applications. With Ajax, web applications can send and retrieve data from a server asynchronously (in the background) without interfering with the display and behaviour of the existing page. By decoupling the data interchange layer from the presentation layer, Ajax allows web pages and, by extension, web applications, to change content dynamically without the need to reload the entire page. In practice, modern implementations commonly utilize JSON instead of XML.

Ajax is not a single technology, but rather a group of technologies. HTML and CSS can be used in combination to mark up and style information. The webpage can then be modified by JavaScript to dynamically display—and allow the user to interact with—the new information. The built-in XMLHttpRequest object, or since 2017 the new "fetch()" function within JavaScript, is commonly used to execute Ajax on webpages, allowing websites to load content onto the screen without refreshing the page. Ajax is not a new technology, or different language, just existing technologies used in new ways.

REST APIs

A RESTful API is an architectural style for an application program interface (API) that uses HTTP requests to access and use data. That data can be used to GET, PUT, POST and DELETE data types, which refers to the reading, updating, creating and deleting of operations concerning resources.

An API for a website is code that allows two software programs to communicate with each other. The API spells out the proper way for a developer to write a program requesting services from an operating system or other application.

A RESTful API -- also referred to as a RESTful web service or REST API -- is based on representational state transfer (REST), which is an architectural style and approach to communications often used in web services development.

REST technology is generally preferred over other similar technologies. This tends to be the case because REST uses less bandwidth, making it more suitable for efficient internet usage. RESTful APIs can also be built with programming languages such as JavaScript or Python.

Pyfirmata

PyFirmata is basically a prebuilt library package of a python program which can be installed in Arduino to allow serial communication between a python script on any computer and an Arduino. This python package can give access to read and write any pin on the Arduino. PyFirmata is an intermediate protocol that connects an embedded system to a host computer, and the protocol channel uses a serial port by default. The Arduino platform is the standard reference implementation for Firmata. The Arduino IDE comes with the support for Firmata.

Burning the Firmata Firmware via Arduino IDE

1. The Arduino IDE >= 1.6.X version comes with the Firmata firmware so we can use Arduino IDE to burn the firmata firmware into Arduino core.
2. Open Arduino IDE and navigate to Examples -> Firmata -> StandardFirmata
3. The StandardFirmata.ino will appear.
4. Select the correct board and port and click.
5. To test the Firmata firmware, you can navigate to the Microsoft Store, search and download the Windows Remote Arduino Experience.

UI Screenshots

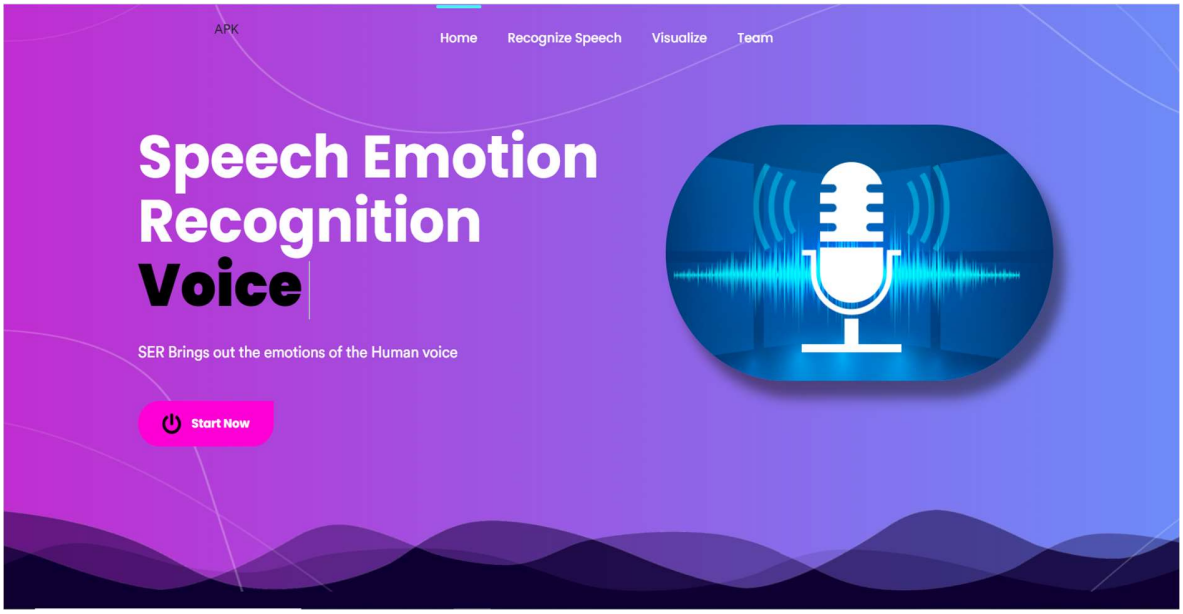

Fig 3. 2:UI Screenshot

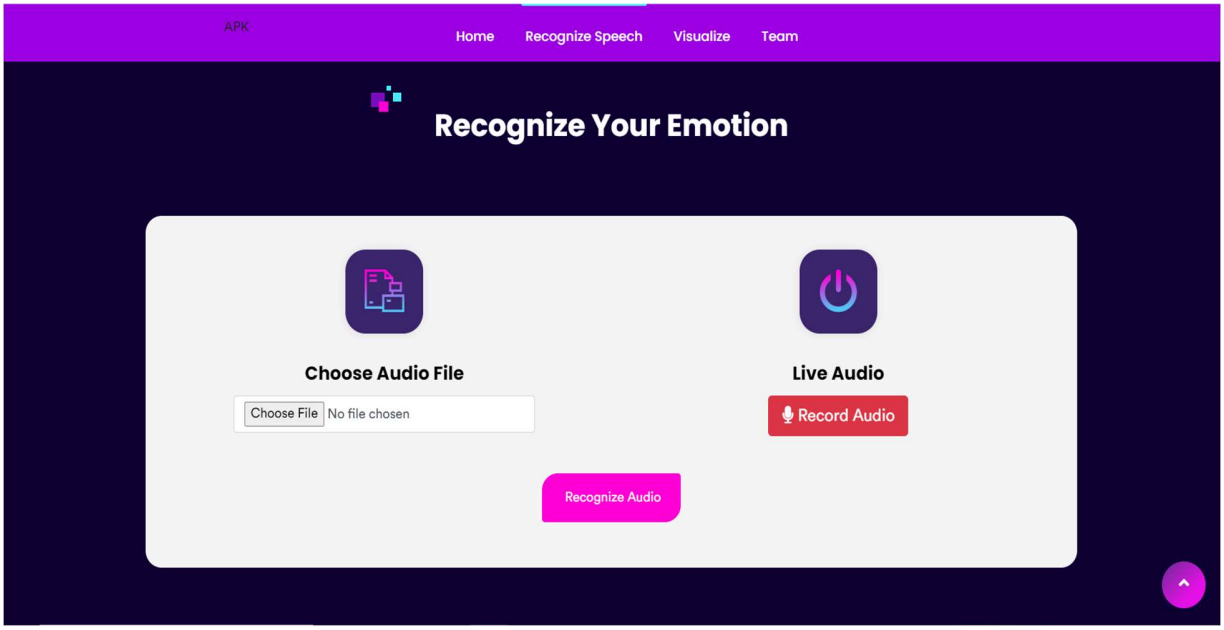

Fig 3. 3:UI Upload Audio Screenshot

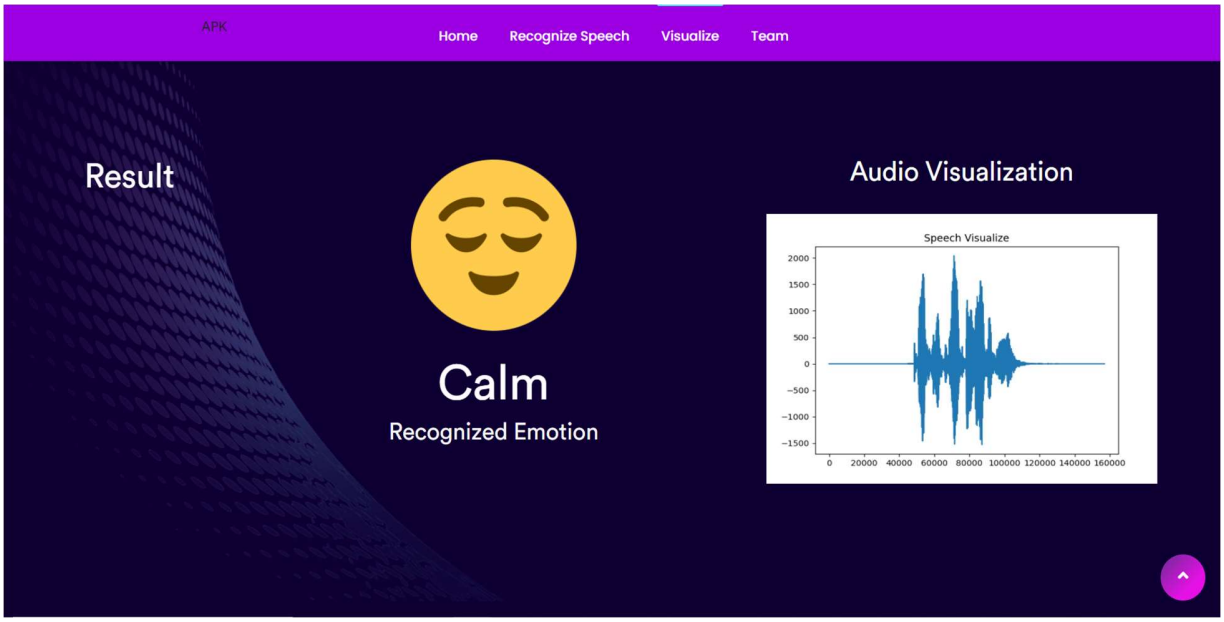

Fig 3. 4: UI Emotion Result Screenshot

3.3 HARDWARE DESIGN

Hardware design, of course, is more constrained than software by the physical world. It is instructive to examine the abstractions that have worked for hardware, such as synchronous design. The synchronous abstraction is widely used in hardware to build large, complex, and modular designs, and has recently been applied to software [6], particularly for designing embedded software.

Conceptually, the distinction between hardware and software, from the perspective of computation, has only to do with the degree of concurrency and the role of time. An application with a large amount of concurrency and a heavy temporal content might as well be thought of using hardware abstractions, regardless of how it is implemented. An application that is sequential and has no temporal behavior might as well be thought of

using software abstractions, regardless of how it is implemented. The key problem becomes one of identifying the appropriate abstractions for representing the design.

Hardware design aims to interface LCD to Arduino. A Liquid Crystal Display commonly abbreviated as LCD is basically a display unit built using Liquid Crystal technology. When we build real-life/real world electronics-based projects, we need a medium/device to display output values and messages. The most basic form of electronic display available is 7 Segment display – which has its own limitations. The next best available option is Liquid Crystal Displays which comes in different size specifications. Out of all available LCD modules in the market, the most commonly used one is the 16×2 LCD Module which can display 32 ASCII characters in 2 lines (16 characters in 1 line). Other commonly used LCD displays are 20×4 Character LCD, Nokia 5110 LCD module, 128×64 Graphical LCD Display and 2.4-inch TFT Touch screen LCD display.

3.3.1 INTERFACING 16×2 LCD TO ARDUINO UNO

LCD modules form a very important part in many arduino based embedded system designs. So the knowledge on interfacing an LCD module to arduino is very essential in designing embedded systems. This section of the article is about interfacing an Arduino to 16×2 LCD. JHD162A is the LCD module used here. JHD162A is a 16×2 LCD module based on the HD44780 driver from Hitachi. The JHD162A has 16 pins and can be operated in 4-bit mode (using only 4 data lines) or 8-bit mode (using all 8 data lines). Here we are using the LCD module in 4-bit mode. First, I will show you how to display a plain text message on the LCD module using an arduino and then I have designed a useful project using LCD and arduino – a digital thermometer.

3.3.2 16×2 LCD MODULE PIN OUT DIAGRAM

The lcd module has 16 pins and can be operated in 4-bit mode or 8-bit mode. Here we are using the LCD module in 4-bit mode. Before going into the details of the project, let's have a look at the LCD module.The schematic of a LCD pin diagram is given below.

Fig 3. 5:Auduino LCD Result Display

The name and functions of each pin of the 16×2 LCD module is given below.

- Pin1(Vss):Ground pin of the LCD module.
- Pin2(Vcc): Power to LCD module (+5V supply is given to this pin)
- Pin3(VEE):Contrast adjustment pin. This is done by connecting the ends of a 10K potentiometer to +5V and ground and then connecting the slider pin to the VEE pin. The voltage at the VEE pin defines the contrast. The normal setting is between 0.4 and 0.9V.
- Pin4(RS):Register select pin.The JHD162A has two registers namely command register and data register. Logic HIGH at RS pin selects data register and logic LOW at RS pin selects command register. If we make the RS pin HIGH and feed an input to the data lines (DB0 to DB7), this input will be treated as data to display on the LCD screen. If we make the RS pin LOW and feed an input to the data

lines, then this will be treated as a command (a command to be written to LCD controller – like positioning cursor or clear screen or scroll).

- Pin5(R/W): Read/Write modes. This pin is used for selecting between read and write modes. Logic HIGH at this pin activates read mode and logic LOW at this pin activates write mode.
- Pin6(E): This pin is meant for enabling the LCD module. A HIGH to LOW signal at this pin will enable the module.
- Pin7(DB0) to Pin14(DB7): These are data pins. The commands and data are fed to the LCD module though these pins.
- Pin15(LED+): Anode of the back light LED. When operated on 5V, a 560 ohm resistor should be connected in series to this pin. In arduino based projects the back light LED can be powered from the 3.3V source on the arduino board.
- Pin16(LED-): Cathode of the back light LED

CHAPTER - 4 TESTING

4.1 GENERAL

During systems testing, the system is used experimentally to ensure that the software does not fail. In other words, we can say that it will run according to its specifications and in the way users expect. Special test data are input for processing, and the results examined. A limited number of users may be allowed to use the system so that analysts can see whether they try to use it in unforeseen ways. It is desirable to discover any surprises before the organization implements the system and depends on it.

Software modules are tested for their functionality as per the requirements identified during the requirements analysis phase. To test the functionality of the file transfer and chat application, a Quality Assurance (QA) team is formed. The requirements identified during the requirements analysis phase are submitted to the QA team. In this phase the QA team tests the application for these requirements. The development team submits a test case report to the QA team so that the application can be tested in the various possible scenarios.

The software development project errors can be injected at any stage during development i.e. if there is an error injected in the design phase then it can be detected in the coding phase because there is the product to be executed ultimately on the machine, so we employ a testing process. During the testing the program to be tested is executed with certain test cases and output of these test cases is evaluated to check the correctness of the program. It is the testing that performs the first step in determining the errors in the program.

Test Cases and Test Criteria

During Test Cases that are good at revealing the presence of faults is central to successful testing. The reason for this is that if there is a fault in the program, the program can still provide the expected behaviour on the certain inputs. Only for the set of inputs the faults that exercise the fault in the program will the output of the program devise from the expected behaviour. Hence, it is fair to say that testing is as good as its test case.

The number of test cases used to determine errors in the program should be minimum. There are two fundamental goals of a practical testing activity: -

1. maximize the number of errors detected and.
2. minimize the number of test cases.

As these two goals are contradictory so the problem of selecting test cases is a complex one. While selecting the test cases the primary objective is to ensure that if there is an error or fault in the program, it is exercised by one of its test cases. An ideal test case is one which succeeds (meaning that there are no errors, revealed in its execution) only if there are no errors in the program one possible set of ideal test cases is one which includes all the possible inputs to the program. This is often called "exhaustive testing", however it is impractical and infeasible as even a small program can have an infinite input domain.

So, to avoid this problem we use "test criteria" in selection of the test cases. There are two aspects of the test case selection: -

1. specifying a criterion for evaluating the test cases

2. generating the set of cases that satisfy a given criteria.

The fully automated process of generating test criteria has not been yet found; rather guidelines are only the automated tool available to us previously. The two fundamental properties for a testing criterion are: -

1. **Reliability :** a criterion is reliable if all the sets that satisfy the criteria detect the same error.
2. **Validity :** a criterion is valid for any error in the program if there is some set satisfying the criteria that will reveal the error.

Unit Testing

During the phase of unit testing different constituent modules were testing against the specifications produced during the design for the modules. Unit testing is essentially for the verification of the code produced during the coding phase, and the goal is to test the internal logic of the modules. The modules once tested were then considered for integration and use by others.

Overview of Testing

1. **Testing:** Testing involves executing the program (or part of it) using sample data and inferring from the output whether the software performs correctly or not. This can be done either during module development (unit testing) or when several modules are combined (system testing).
2. **Defect Testing:** Defect testing is testing for situations where the program does not meet its functional specification. Performance testing tests a system's

performance or reliability under realistic loads. This may go some way to ensuring that the program meets its non-functional requirements.

3. **Debugging:** Debugging is a cycle of detection, location, repair and test. Debugging is a hypothesis testing process. When a bug is detected, the tester must form a hypothesis about the cause and location of the bug. Further examination of the execution of the program (possible including many returns of it) will usually take place to confirm the hypothesis. If the hypothesis is demonstrated to be incorrect, a new hypothesis must be formed. Debugging tools that show the state of the program are useful for this, but inserting print statements is often the only approach.

Overview of Testing Strategies:

Large systems are usually tested using a mixture of strategies. Different strategies may be needed for different parts of the system or stages of the process.

Top-down testing: This approach tests high levels of system before detailed components. This is appropriate when developing the system top-down and is likely to show up structural design errors early (and therefore cheaply). But this often has the advantage that a limited, working system is available early on. Validation (as distinct from verification) can begin early. Its disadvantage is that stubs need to be generated (extra effort) and might be impracticable if the component is complex (e.g. converting an array into a linked list; unrealistic to generate random list; therefore, end up implementing unit anyway). Test output may be difficult to observe (needs creation of artificial environment). This is not appropriate for OO systems (except within a class).

Bottom-up testing: This is opposite of top-down testing. This tests low-level units then works up the hierarchy. Its advantages and disadvantages mirror those of top-down testing. In this testing there is a need to write test drivers for each unit. These are as reusable as the unit itself. Combining top-down development with bottom-up testing means that all parts of the system must be implemented before testing can begin, which does not accord with the incremental approach discussed above. Bottom-up testing is less likely to reveal architectural faults early on. However, bottom-up testing of critical low-level components is almost always necessary. Appropriate for OO systems.

Stress testing: Tests system's ability to cope with a specified load (e.g. transactions per second). Tests should be planned to increase load incrementally. This type of testing goes beyond design limits until the system fails (this test is particularly important for distributed systems like checking degradation of performance a s network traffic increases).

4.2 TESTING TECHNIQUE AND TESTING STRATEGIES

There are two fundamental approaches to testing they are:

FUNCTIONAL TESTING/BLACK BOX TESTING:

In functional testing the structure of the program is not considered. Test cases are solely determined on the basis of requirement or specification of the program or module. Internals of the modules and the programs are not considered for selection of test cases. Due to this nature it is also called "black box testing". The most obvious functional testing procedure is "exhaustive testing", which is impractical. The other criteria for generating the test case are to generate them "randomly" this strategy has a little chance of resulting in a test case that is close to optimal.

There is no appropriate criterion for developing the test case so we have certain heuristic approaches -

a. **Equivalence Class Partitioning:** in which the domain of all inputs is divided into a set of equivalence classes, so that if any test in that class succeeds, then every test in that class will succeed i.e. the success of one set element implies the success of the other.
b. **Boundary Value Analysis:** It has been observed that programs work correctly for a set of values in an equivalence class that fails on certain values. These values generally lie on the boundary of equivalence class. So, in the boundary value analysis we chose an input for a test case from an equivalence class, such that input lies on the edge of the equivalence class. Boundary value test cases are also called "extreme cases".
c. **Cause-Effect Graphing:** The problem with the prior approaches is that they consider each input separately i.e. both concentrate on classes and conditions of one input. The combination of inputs is not considered which is used in many cases. One way to exercise the combination of various input conditions is to consider all the valid combinations of the equivalence class of the input conditions. The technique starts with identifying the causes and the effect of the system under testing.

STRUCTURAL TESTING

In the structural approach the test cases are generated the basis of the actual code of the program or the module to be tested, this structural approach is sometimes called "glass

box testing", This testing is concerned with the implementation of the program. The content is not to exercise the various input conditions rather different programming structures and data structures used in the program. There are three different approaches to structural testing they are-

a. **Control flow-based criteria:** Most common structure-based criteria use control flow-based testing in which the control flow graph of the program is considered and coverage of various aspects of the graph are specified as criteria. The various control flow-based strategies are
    i. statement coverage,
    ii. branch coverage and
    iii. all path coverage
b. **Data Flow based testing:** The basic idea behind the data flow-based testing is to make sure that during testing, the definitions of variables and their subsequent use is tested. To implement the data flow-based testing the data flow graph is first made from the control flow graph.
c. **Mutation Testing:** In the above two testing techniques the focus is on which path to be executed, but mutation testing is not a path-based approach. The mutation testing requires that the set of test cases must be such that they can distinguish between the original programs and mutants. In the software world there is no fault model as in hardware so most of the techniques do guess work regarding where the fault must lie and then select the test cases to reveal those faults. In Mutation testing faults of some pre-decided types are introduced in the program being tested. Then those faults are found in the mutants.

The aim of the system testing process was to determine all defects in our project. The program was subjected to a set of test inputs and various observations were made and based on these observations it will be decided whether the program behaves as expected or not.

4.3 DEBUGGING AND CODE IMPROVEMENT

In ideal worlds, all programmers would be so skilled and attentive to detail that they would write bug-free code. Unfortunately, we do not live in an ideal world. As such, debugging, or tracking down the source of errors and erroneous results, is an important task that all developers need to perform before they allow the end-user to use their applications. We will discuss some techniques for reducing the number of bugs in code up front.

There are three categories of bugs

a. **Syntax error:** These errors occur when code breaks the rule of the language, such as visual Basic sub statement without a closing End sub, or a forgotten closing curly brace ({}) in c#. These errors are the easiest to locate. The language compiler or integrated development environment (IDE) will alert you to them and will not allow you to compile your program until you correct them.

b. **Semantic error:** These errors occur in code that is correct according to rules of the compiler, but that causes unexpected problems such as crashes or hanging on execution. A good example is code that executes in a loop but never exists the loop, either because the loop depends on the variable whose values was expected to be something different than it actually was or because the programmer forgets to increment the loop counter.

c. **Logic error:** Logic errors are like semantic errors, logic errors are runtime errors. That is, they occur while the program is running. But unlike semantic errors, logic errors do not cause the application to crash or hang. Logic error results in unexpected values or output. This can be a result of something as simple as a mistyped variable name that happens to match another declared variable in the program. This type of error can be extremely difficult to track down to eliminate.

CHAPTER - 5 IMPLEMENTATION

5.1 GENERAL

Implementation is the process of having systems personnel check out and put new equipment into use, train users, install the new application and construct any files of data needed to use it. This phase is less creative than system design. Depending on the size of the organization that will be involved in using the application and the risk involved in its use, systems developers may choose to test the operation in only one area of the firm with only one or two persons. Sometimes, they will run both old and new systems in parallel to compare the results. In still other situations, system developers stop using the old system one day and start using the new one the next.

Evaluation of the system is performed to identify its strengths and weaknesses. The actual evaluation can occur along any of the following dimensions:

1. **Operational Evaluation:** Assessment of the manner in which the system functions, including case of use, response time, overall reliability and level of utilization.
2. **Organizational Impact :** Identification and measurement of benefits to the organization in such areas as financial concerns, operational efficiency and competitive impact.
3. **User Manager Assessment:** Evaluation of the attitudes of senior and user manager within the organization, as well as end-users.

4. **Development Performance:** Evaluation of the development process in accordance with such yardsticks as overall development time and effort, conformance to budgets and standards and other project management criteria.

Emotions play an important role in our day-to-day life and they are generated by humans naturally. In computer science, a lot of research has been done especially since finding an emotion from the text or voice is not that simple and it depends on the current state of the speaker and also the culture of the speaker. If we take this simple sentence “Why don’t you ever text me!” we can either arbitrate as an angry or sad emotion. Since there are no facial expressions, the detection of emotion from voice should be a challenging problem.

Emotions are basic human traits and have been studied by researchers in the fields of psychology, sociology, medicine, computer science, etc. for the past several years. Some of the prominent working understanding and categorizing emotions include Ekman’s six class categorization and Plutchik’s “Wheel of Emotion” which is suggested as eight primary bipolar emotions.

Given the vast nature of the study in this field, there is naturally no consensus on the granularity of emotion classes. The major use of this application is since emotions are important to determine the state of the sentence (In real world) sometimes judging the emotion from text/semantic meaning analysis from recording could be ambiguous. So, for accurate emotion analysis, we go with frequency analysis. It tells the state of the speaker. And, it can be used alongside text recognition.

Various automatic facial expression recognition systems have been developed to detect human emotions. Effective computing research becomes more reliable, valid, and accessible as computer science develops. In general, we see the increasing support for the idea that automated facial expression analysis is technically achievable.

There are various challenges to detect emotions from the paralinguistic content. Extraction of emotions from speech is always tricky due to overlapping regions of emotions. Further, the interpretation of emotions is highly subjective and may vary from person to person. Interpretation also changes due to the change in pitch or amplitude of the sound. For example, 'Disgust' emotion may sound as 'Anger' if we listen to it at high amplitudes. In the absence of context and language understanding, sometimes humans also find it challenging to recognize the emotions.

The study of emotions in speech is involved, and models designed for recognizing it have low accuracy because of the ambiguity in label assignment. Moreover, emotions change with the change in intensity of speech. Thus, it becomes necessary to develop the classifiers which consider this change.

5.2 MODEL IMPLEMENTATION

5.2.1 DATASET

Training data for Modeling is a key input to algorithms that comprehend from such data and memorize the information for future prediction. Although, various aspects come

during the model development, without which various crucial tasks cannot be accomplished. Amid, training data is a backbone of the entire AI and ML project without that it is not possible to train a machine that learns from humans and predicts for humans.

Data collected from multiple sources are usually available in unorganized format, which is not useful for machines to ingest the useful information. But when such data is labeled or tagged with annotation it becomes a well-organized data that can be used to train the AI or ML model. Training data does not necessarily mean you should have labeled or annotated data sets, instead an organized data set is also very important for machine learning model training.

Another most important role of training data for machine learning is classifying the data sets into various categories which is very much important for supervised machine learning. When your algorithm learns what are the features are important in distinguishing between two classes. It helps them to recognize and classify the similar objects in future, thus training data is very important for such classification. And if it is not accurate it will badly affect the model results, that can become the major reason behind the failure of AI projects. We have used two datasets in this work:

- **Ryerson Audio-Visual Database for Emotional Speech and Song -** RAVDESS dataset[17] includes 7356 files made by 24 trained actors (12 female, 12 male), vocalizing two lexically matched comments in a neutral North American accent. Speech incorporates calm, happy, sad, angry, fearful, surprise and disgusted emotions, and the songs comprise calm, happy, sad, angry, and fearful emotions.

Each expression appears at two degrees of emotional strength (normal, strong) with an additional neutral expression. All parameters are available in three formats: audio-only (16bit, 48kHz.wav), audio-video (720p H.264, AAC 48kHz,.mp4), and video-only (no sound). Ratings were given by 247 individuals who were representative of untrained adult study participants from North America.

Each of the 7356 RAVDESS files has a unique filename. The filename consists of a 7-part numerical identifier (e.g., 02- 01-06-01-02-01-12.mp4). These identifiers define the stimulus characteristics:

- Modality (01 = full-AV, 02 = video-only, 03 = audio only);
- Vocal channel (01 = speech, 02 = song);
- Emotion (01 = neutral, 02 = calm, 03 = happy, 04= sad, 05 = angry, 06 = fearful, 07 = disgust, 08 =surprised);
- Emotional intensity (01 = normal, 02 = strong).
- Statement (01 = ”Kids are talking by the door”, 02 = ”Dogs are sitting by the door”);
- Repetition (01 = 1st repetition, 02 = 2nd repetition).
- Actor (01 to 24. Odd-numbered actors are male, even-numbered actors are female).

NOTE: There is no strong intensity for the ’neutral’ emotion.

Speech signals have a different amplitude as each signal incorporates different emotions. It is obvious that a signal with a higher amplitude is likely to include

anger. In other languages, this might be different. Emotions, happy and sad, have a common amplitude, and it is fascinating to differentiate between the two emotions using machine learning and to know which features are relevant.

- **Toronto Emotional Speech Set (TESS) Collection -** There are a set of 200 target words were spoken in the carrier phrase "Say the word _' by two actresses (aged 26 and 64 years) and recordings were made of the set portraying each of seven emotions (anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral). There are 2800 data points (audio files) in total[18]. The dataset is organised such that each of the two female actors and their emotions is contained within its folder. And within that, all 200 target words audio files can be found. The format of the audio file is a WAV format.

5.2.2 DATA PRE-PROCESSING

For making a generalized model we combined RAVDESS speech, RAVDESS song and TESS dataset. This helped us in creating a model which can perform well in various scenarios. The number of instances in each dataset and the total number of instances in the training dataset is shown below.

Dataset	Instances
RAVDESS Speech	1440
RAVDESS Song	1012
TESS	2800
Total	**5252**

Table 5. 1:Number Of Instances in Databases

After combining the three datasets following is the class distribution of our dataset.

Emotion	Instances	Percentage
Neutral	588	11.272
Calm	376	7.159
Happy	776	14.775
Sad	776	14.775
Angry	776	14.775
Fear	776	14.775
Disgust	592	11.272
Surprised	592	11.272

Table 5. 2:Class distribution of our dataset

Here, it is visible that we have fewer instances for calm emotion than for others. But, at an overall level, we have more data points in our dataset for other emotions. This slight class imbalance won't affect our model's performance much. After this, we loaded all sound files into the memory at a sampling rate of 22050 Hz. High sampling rate provided us with data of higher quality but less in size compared to the original one. This helped us in making our model faster.

5.2.3 FEATURE ENGINEERING

Mel-frequency Cepstrum coefficient is the most used representation of spectral property of voice signals. These are the best for speech recognition as it takes human perception sensitivity with respect to frequencies into consideration. For each frame, the Fourier transform and the energy spectrum were estimated and mapped into the Mel-frequency

scale. Mel-Frequency Cepstral Coefficients (MFCC) represents the spectral property of the speech signals. The vocal cords, the tongue, and the teeth are all the elements that filter the sound and make it unique for each speaker. The voice is determined by the shape of these elements. The shape manifests in the envelope of the short-time power spectrum, and MFCCs represent this envelope. Usually, the process of calculating MFCC is shown in the figure below.

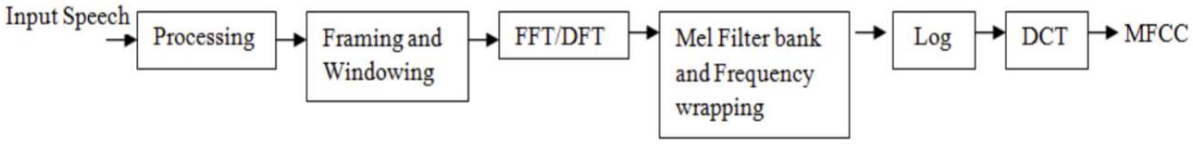

Fig 5.1:Feature Engineering Process

In our project, we have extracted the first 40-order of the MFCC coefficients where the speech signals are sampled at 22050 Hz. The Discrete Cosine Transform (DCT) of the mel log energies were estimated and the first 40 DCT coefficients provided the MFCC values used in the classification process.

The window size used for the feature extraction was 2048 samples combined with a hop length of 512 samples. This helped us in calculating Fast Fourier Transform. In terms of milliseconds, we used a window size of about 93 ms and hop length of 23.22 ms. Hann windowing function was used for avoiding spectral leakage along with overlapping frames. The entire MFCCs extraction pipeline is shown below.

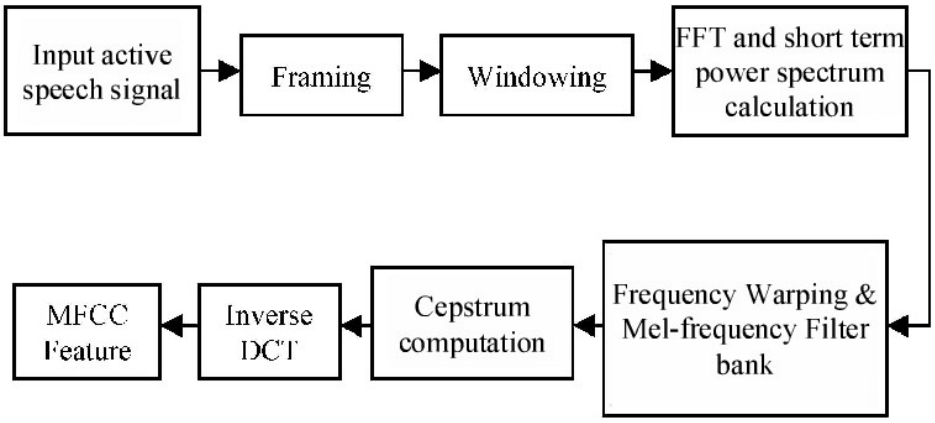

Fig 5. 2:MFCC Feature extraction

The feature extraction at this minute level helped us in extracting finer details present in the speaker's voice through which we were able to characterize each emotion. After this process, we received a 2D - matrix of size number of coefficients X number of frames for each audio file. But this had a problem of its own. All the files were not of same length clipping the files would have resulted in the loss of data.

Therefore, we transposed these matrices and took a mean along each Mel-frequency Cepstral Coefficient. This provides us with a vector of length 40 for each of the instances. After this, we split our dataset into train, validation and test set in a ratio of 70:15:15. The number of instances after the split in each set is shown in the figure below.

Dataset	**Instances**
Train	3676
Validation	788
Test	788

Table 5. 3:Number of instances after the split

5.2.4 MODEL ARCHITECTURE

The model is constructed based on 1D CNN with a fully-connected layer. Each 1D convolutional layer is followed by a ReLU activation layer, a dropout layer and a 1D Max Pooling layer. ReLU activation function is applied to contribute to the sparsity of the network which reduces the interdependence of parameters and alleviates the occurrence of the over-fitting problem. As shown in the table below, the model has three 1D convolutional layers. After the 3rd 1D convolutional layer, the feature maps are flattened into a one-column matrix to fit them into the fully connected layers. The flattened output is then fed to a dense layer then finally softmax function is applied for classification in the last layer. In summary, the whole network is made up of three 1D convolutional layers and three dropout layers and a dense layer. The model architecture is shown in the table below.

Layer	Input Shape	Output Shape
Conv 1D	40 x 1	40 x 64
ReLU	40 x 64	40 x 64
Dropout	40 x 64	40 x 64
Max Pooling 1D	40 x 64	10 x 64
Conv 1D	10 x 64	10 x 128
ReLU	10 x 128	10 x 128
Dropout	10 x 128	10 x 128
Max Pooling 1D	10 x 128	2 x 128
Conv 1D	2 x 128	2 x 256
ReLU	2 x 256	2 x 256
Dropout	2 x 256	2 x 256

Flatten	2 x 256	512
Dense	512	8
Softmax Activation	8	8 (Output)

Table 5. 4:The model architecture

5.2.5 EVALUATION

5.2.5.1 EVALUATION METRICS

A confusion matrix is a performance measurement method for the evaluation of models in classification problems. It's a kind of table that allows you to understand the performance of the models on a test dataset for which true values are known. The word confusion matrix itself is rather plain, but its associated terms can be a little complicated. Here a simple explanation is given for this technique.

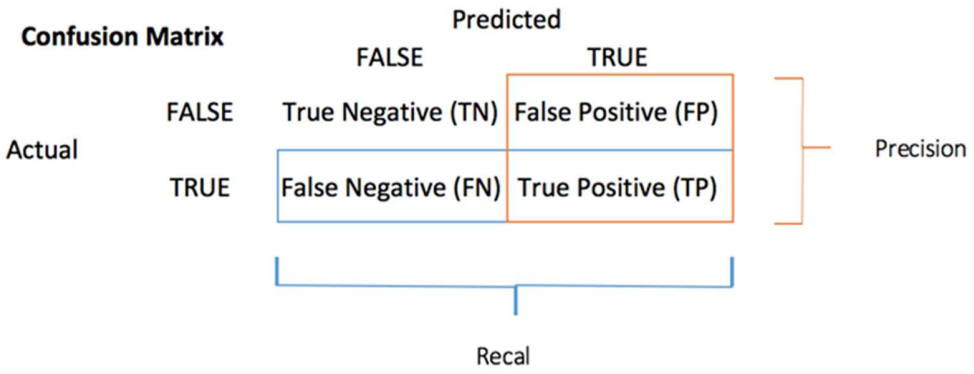

Fig 5. 3:Confusion Matrix

Confusion Matrix is a useful machine learning tool that helps you to calculate Memory, Precision, Accuracy, and F1-Score. It also helps in predicting various mathematical terms such as **Accuracy, True Positive Rate(TPR), True Negative Rate(TNR), False Positive Rate(FPR), False Negative Rate(FNR), and Precision**. All this helps us in understanding machine learning more accurately and to improve it accordingly.

Following are the formulas to calculate some mathematical terms.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where, TP = True Positive

TN = True Negative

FP = False Positive

FN = False Negative

Precision tells us how many of the correctly predicted cases actually turned out to be positive.

$$Precision = \frac{TP}{TP + FP}$$

Recall tells us how many of the actual positive cases we were able to predict correctly with our model.

$$Recall = \frac{TP}{TP + FN}$$

F1-score is a harmonic mean of Precision and Recall, and so it gives a combined idea about these two metrics. It is maximum when Precision is equal to Recall.

$$F1 - Score = \frac{2 * Recall * Precision}{Recall + Precision}$$

5.2.5.2 EVALUATION SCORES

We achieved 86.55% of accuracy and 86.62% of weighted F1-score during the evaluation on the test set. The precision, recall and F1-score for individual classes are shown in the table below.

Emotion	Precision	Recall	F1-Score	Support
Neutral	0.952381	0.909091	0.930233	88
Calm	0.718750	0.807018	0.760331	57
Happy	0.923077	0.820513	0.868778	117
Sad	0.841667	0.870690	0.855932	116
Angry	0.935780	0.879310	0.906667	116
Fearful	0.818966	0.818966	0.818966	116
Disgust	0.845361	0.921348	0.881720	89
Surprise	0.851064	0.898876	0.874317	89
Weighted Average	0.869218	0.865482	0.866238	788

Table 5. 5:Emotion Classes Scores

Following figures show the non-normalized and normalized confusion matrix for the predictions on the test set.

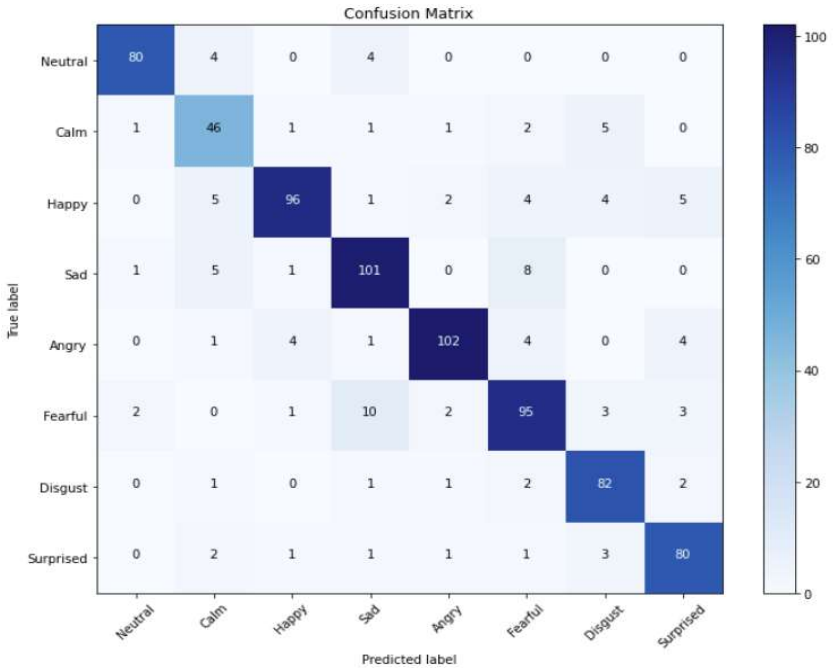

Fig 5. 4:SER Confusion Matrix

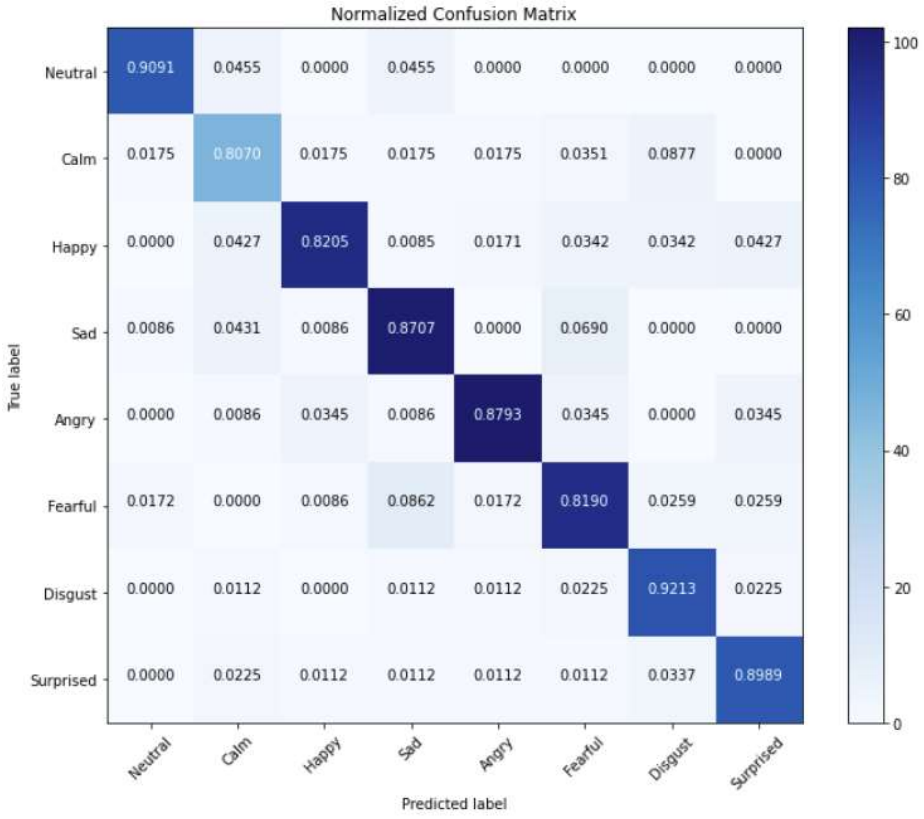

Fig 5. 5:Normalized Confusion Matrix

5.3 SOFTWARE INSTALLATION & IMPLEMENTATION

Software implementation begins with the effort of software fabrication. Fabrication is an act of making something. Software fabrication involves programmatic design, source code editing or programming, and testing of each software unit. This series of technical tasks represents how software procedures, routines, modules, objects, or graphical models are produced. Each software unit is presumed to be suitable for their intended purpose or role in the overall architectural context. The result of software fabrication should be a documented unit of source code that has been tested against its structural unit specification. This source code (software unit) is then available to be assembled, integrated, and compiled with other fabricated software elements to craft larger software

components. These integrated software components are tested against structural component specifications to ensure their correctness. This assembly, integration, and testing series of events continues to generate larger, more complex software components. Software integration progresses until a completely integrated and tested software configuration item is realized and available for acceptance testing.

Application Development Process

1. Install django library into the python environment so that we can create our application and control it.
2. Create the project using `django-admin startproject mysite`

   Django File Structure

   - The outer mysite/ root directory is a container for your project. Its name doesn't matter to Django; you can rename it to anything you like.
   - manage.py: A command-line utility that lets you interact with this Django project in various ways.
   - The inner mysite/ directory is the actual Python package for your project. Its name is the Python package name you'll need to use to import anything inside it (e.g. mysite.urls).
   - mysite/__init__.py: An empty file that tells Python that this directory should be considered a Python package.
   - mysite/settings.py: Settings/configuration for this Django project. Django settings will tell you all about how settings work.

- mysite/urls.py: The URL declarations for this Django project; a “table of contents” of your Django-powered site.mysite/asgi.py: An entry-point for ASGI-compatible web servers to serve your project.
- mysite/wsgi.py: An entry-point for WSGI-compatible web servers to serve your project.

3. Setting up the development server & starting it to the localhost at port 8000. `python manage.py runserve` this command to start the server and host it on the local network at port 8000.
4. Now the Frontend part is made using the django template. A Django template is a text document or a Python string marked-up using the Django template language. Some constructs are recognized and interpreted by the template engine. The main ones are variables and tags. As we used for loop in above example, we used it as a tag. Similarly, we can use various other conditions such as if, else, if-else, empty, etc. Main characteristics of django Template language are Variables, Tags, Filters and Comments.
5. The django template also includes CSS & JavaScript files. Which helps the application in designing and controlling the web application.
    A. Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.
    B. JavaScript, often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often

just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

6. Creating the AJAX requests for fetching, setting & performing various tasks asynchronously.

I. User sends a request from the UI and a javascript call goes to XMLHttpRequest object.
II. HTTP Request is sent to the server by XMLHttpRequest object.
III. Server interacts with the database using Python, PHP, JS, ASP.net etc.
IV. Data is retrieved.
V. Server sends XML data or JSON data to the XMLHttpRequest callback function.
VI. HTML and CSS data is displayed on the browser.

7. On triggering the button event on the UI, the Javascript function is called which makes 5 Ajax calls on the request which makes the request to store uploaded or recorded voice in the storage (drive), then it makes a request to the ML model for fetching emotion passing the audio file path, after loading and running the model we get the emotion label (JSON obj) which further make another request to Arduino UNO to display the emotion label on LCD and waits for the success message from it and at last after confirmation from arduino it displays the result to the UI. Also a request is made to visualize recorded or uploaded audio files to the frontend.

8. The Recognize Emotion button can be triggered more than one time after every successful request without refreshing which makes the UI more user friendly and fast. This creates a fast Single page application and provides a seamless user experience in using our application. It also has world class pictorization which makes it more creative and understandable.

Using Python to Interpret Firmata

There are libraries that implement the Firmata protocol in order to communicate (from a computer, smartphone or tablet for example) with Firmata firmware running on a microcontroller platform. pyFirmata is a Python interface for the Firmata protocol and runs on python3

Ngrok

Ngrok is a reverse proxy that creates a secure tunnel from a public endpoint to a locally running web service. ngrok captures and analyzes all traffic over the tunnel for later inspection and replay.

What can ngrok do?

- Expose any http service behind a NAT or firewall to the internet on a subdomain of ngrok.com
- Expose any tcp service behind a NAT or firewall to the internet on a random port of ngrok.com
- Inspect all http requests/responses that are transmitted over the tunnel
- Replay any request that was transmitted over the tunnel

What is ngrok useful for?

- Temporarily sharing a website that is only running on your development machine
- Demoing an app at a hackathon without deploying
- Developing any services which consume webhooks (HTTP callbacks) by allowing you to replay those requests
- Debugging and understanding any web service by inspecting the HTTP traffic
- Running networked services on machines that are firewalled off from the internet

5.4 HARDWARE INSTALLATION & IMPLEMENTATION

Before we can understand the UNO's hardware, we must have a general overview of the system first.After your code is compiled using Arduino IDE, it should be uploaded to the main microcontroller of the Arduino UNO using a USB connection. Because the main microcontroller doesn't have a USB transceiver, you need a bridge to convert signals between the serial interface (UART interface) of the microcontroller and the host USB signals.The bridge in the latest revision is the ATmega16U2, which has a USB transceiver and also a serial interface (UART interface).To power your Arduino board, you can use the USB as a power source. Another option is to use a DC jack. You may ask, "if I connect both a DC adapter and the USB, which will be the power source?" The answer will be discussed in the "Power Part" section from this article.To reset your board, you should use a push button in the board. Another source of reset should be every time you open the serial monitor from Arduino IDE.

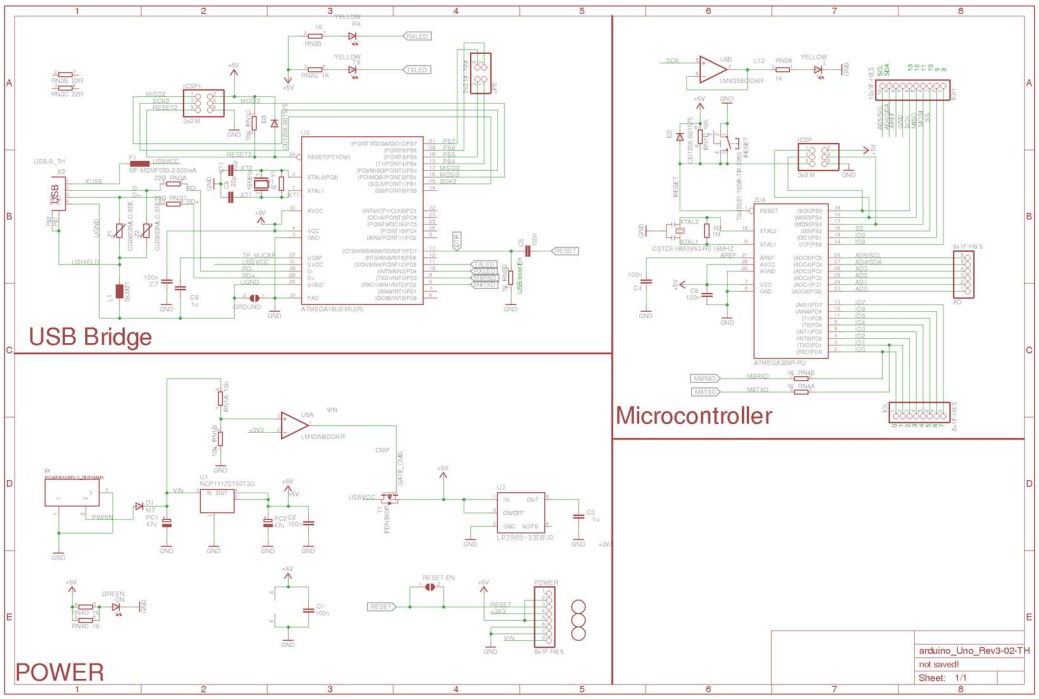

Fig 5. 6:Microcontroller Layout

5.4.1 THE MICROCONTROLLER

It is important to understand that the Arduino board includes a microcontroller, and this microcontroller is what executes the instructions in your program. If you know this, you won't use the common nonsense phrase "Arduino is a microcontroller" ever again.

The ATmega328 microcontroller is the MCU used in Arduino UNO R3 as a main controller. ATmega328 is an MCU from the AVR family; it is an 8-bit device, which means that its data-bus architecture and internal registers are designed to handle 8 parallel data signals.

ATmega328 has three types of memory:

- Flash memory: 32KB nonvolatile memory. This is used for storing applications, which explains why you don't need to upload your application every time you unplug the arduino from its power source.
- SRAM memory: 2KB volatile memory. This is used for storing variables used by the application while it's running.
- EEPROM memory: 1KB nonvolatile memory. This can be used to store data that must be available even after the board is powered down and then powered up again.

5.4.2 DIGITAL I/O:

This MCU has three ports: PORTC, PORTB, and PORTD. All pins of these ports can be used for general-purpose digital I/O or for the alternate functions indicated in the pinout below. For example, PORTC pin0 to pin5 can be ADC inputs instead of digital I/O. There are also some pins that can be configured as PWM output. These pins are marked with "~" on the Arduino board.

Note: The ATmega168 is almost identical to the ATmega328 and they are pin compatible. The difference is that the ATmega328 has more memory—32KB flash, 1KB EEPROM, and 2KB RAM compared to the ATmega168's 16KB flash, 512 bytes EEPROM, and 1KB RAM.

5.4.3 THE USB-TO-UART BRIDGE

As we discussed in the "Arduino UNO System Overview" section, the role of the USB-to-UART bridge part is to convert the signals of USB interface to the UART interface,

which the ATmega328 understands, using an ATmega16U2 with an internal USB transceiver. This is done using special firmware uploaded to the ATmega16U2.

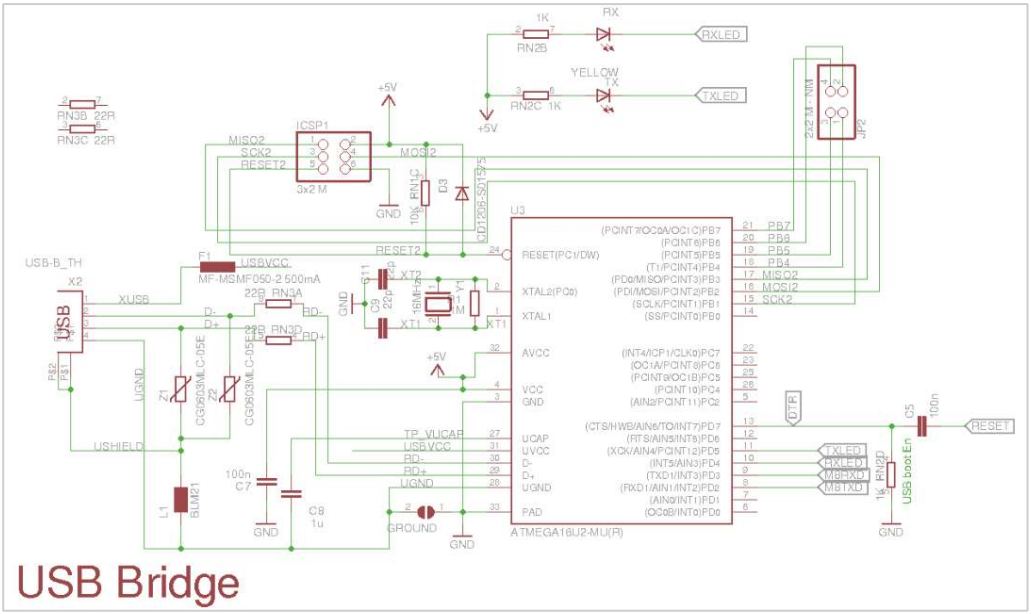

Fig 5. 7:Arduino USB Bridge Part

From an electronic design perspective, this section is similar to the microcontroller section. This MCU has an ICSP header, an external crystal with load capacitors (CL), and a Vcc filter capacitor.

Notice that there are series resistors in the D+ and D- USB lines. These provide the proper termination impedance for the USB signals. Here is some further reading about these resistors:

1. Why USB data series resistors
2. USB Developers FAQ

Z1 and Z2 are voltage-dependent resistors (VDRs), also called varistors. They are used to protect the USB lines against ESD transients.

The 100nF capacitor connected in series with the reset line allows the Atmega16U2 to send a reset pulse to the Atmega328. You can read more about this capacitor here.

5.4.4 THE POWER

For a power source, you have the option of using the USB or a DC jack. Now it's time to answer the following question: "If I connect both a DC adapter and the USB, which will be the power source?

5.4.5 CIRCUIT DIAGRAM AND EXPLANATION

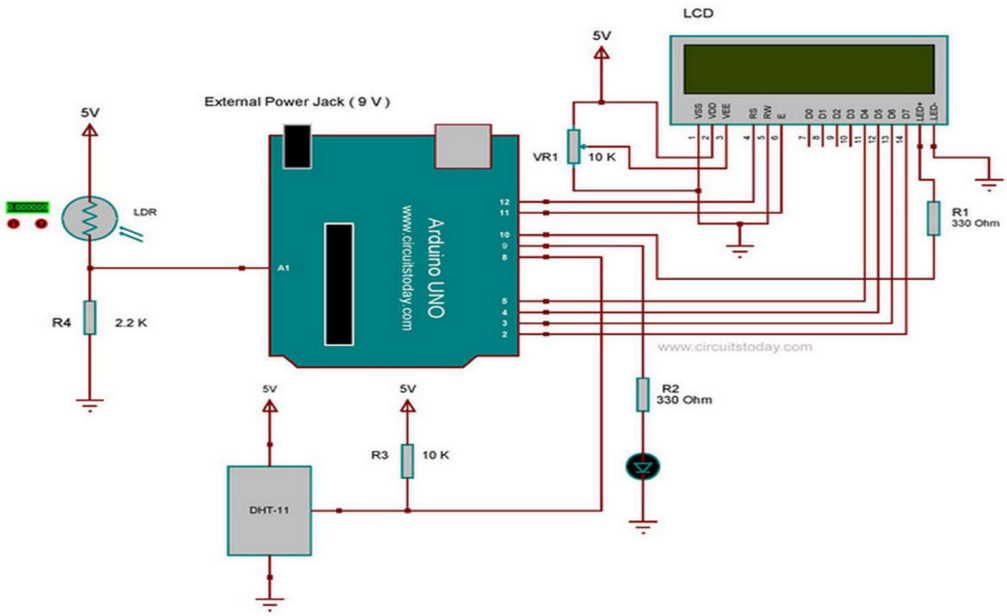

Fig 5. 8:Arduino Circuit Diagram

RS pin of the LCD module is connected to digital pin 12 of the arduino. The R/W pin of the LCD is grounded. Enable pin of the LCD module is connected to digital pin 11 of the arduino. In this project, the LCD module and arduino are interfaced in the 4-bit mode.

This means only four of the digital input lines(DB4 to DB7) of the LCD are used. This method is very simple, requires less connections and you can almost utilize the full potential of the LCD module. Digital lines DB4, DB5, DB6 and DB7 are interfaced to digital pins 5, 4, 3 and 2 of the Arduino. The 10K potentiometer is used for adjusting the contrast of the display. 560 ohm resistor R1 limits the current through the back light LED.

First we need to enable the header file ('#include <LiquidCrystal.h>'), this header file has instructions written in it, which enables the user to interface an LCD to UNO in 4 bit mode without any fuzz. With this header file we need not have to send data to LCD bit by bit, this will all be taken care of and we don't have to write a program for sending data or a command to LCD bit by bit.

Second we need to tell the board which type of LCD we are using here. Since we have so many different types of LCD (like 20x4, 16x2, 16x1 etc.). Here we are going to interface a 16x2 LCD to the UNO so we get 'lcd.begin(16, 2);'. For 16x1 we get 'lcd.begin(16, 1);'.

In this instruction we are going to tell the board where we connected the pins. The pins which are connected need to be represented in order as "RS, En, D4, D5, D6, D7". These pins are to be represented correctly. Since we have connected RS to PIN0 and so on as shown in the circuit diagram, we represent the pin number to board as "LiquidCrystal lcd(0, 1, 8, 9, 10, 11);"..

As you can see we need not to worry about anything else, we just have to initialize and the UNO will be ready to display data. We don't have to write a program loop to send the data BYTE by BYTE here.

5.4.6 CONTROLLING WITH PYTHON

You cannot program Arduino with python. You can however upload an open source Firmata protocol implementation onto the Arduino and then write python code on your PC that will send commands to the Arduino by USB cable to tell it what to do. This way you actually execute your code on PC, but control Arduino pins.

In order to display the emotion on the arduino display we have developed a API using flask and pyfirmata. Firstly the model recognises the emotion and emotion detected is sent to the arduino display using the serial port communication. Here the use of API comes into the picture which sends a request to the controller to display the result. To make the server public and send the request via remote control we have used ngrok. The local server running at port 7070 is made public using ngrok. The type of emotion recognised is sent by the remote server to the local server and the arduino lcd shows the result by communicating through the serial port.

CONCLUSION & FUTURE SCOPE

Through this project, we showed how we can leverage Deep learning to obtain the underlying emotion from speech audio data and some insights on the human expression of emotion through voice. This system can be employed in a variety of setups like Call Centre for complaints or marketing, in voice-based virtual assistants or chatbots, in linguistic research, etc.

A few possible steps that can be implemented to make the models more robust and accurate are the following

- An accurate implementation of the pace of the speaking can be explored to check if it can resolve some of the deficiencies of the model.
- Figuring out a way to clear random silence from the audio clip.
- Exploring other acoustic features of sound data to check their applicability in the domain of speech emotion recognition. These features could simply be some proposed extensions of MFCC like RAS-MFCC or they could be other features entirely like LPCC, PLP or Harmonic cepstrum.
- Following lexical features based approach towards SER and using an ensemble of the lexical and acoustic models. This will improve the accuracy of the system because in some cases the expression of emotion is contextual rather than vocal.
- Adding more data volume either by other augmentation techniques like time-shifting or speeding up/slowing down the audio or simply finding more annotated audio clips.

APPENDIX

TERMS	FULL FORMS
SER	Speech Emotion Recognition
AI	Artificial Intelligence
IDE	Integrated Development Environment
DL	Deep Learning
KNN	K-Nearest Neighbors
RAVDESS	Ryerson Audio-Visual Database of Emotional Speech and Song
SVM	Support Vector Machine
UA	Unweighted average
RBF	radial basis function
MSFs	modulation spectral feature
VAM	Vera am Mittag
PLPC	perceptual linear prediction coefficients
MFCC	Mel Frequency Cepstral Coefficients
SAD	System Analysis and Design
HTTP	Hypertext Transfer Protocol
DBMS	Database Management System

UI	User Interface
HTML	Hypertext Mark-up Language
AJAX	Asynchronous JavaScript
LCD	Liquid Crystal Display
JSON	JavaScript Object Notation
API	Application programming interface
WSGI	Web Server Gateway Interface
CSS	Cascading Style Sheets
I/O	Input/Output
DOM	Document Object Model
XML	Extensible mark-up language
VEE	Voltage Emitter Emitter
ASCII	American Standard Code for Information Interchange
TFT	Thin Film Transistor
QA	Quality Assurance
TESS	Toronto Emotional Speech Set
DCT	Discrete Cosine Transform

REFERENCES

1. I.T. Kun Han, D. Yu, Speech emotion recognition using deep neural networks and extreme learning machine, Interspeech (2014) 223–227.
2. A.M. Badshah, J. Ahmad, N. Rahim, S.W. Baik, Speech emotion recognition from spectrograms with deep convolutional neural network, in: 2017 International Conference on Platform Technology and Service (PlatCon), IEEE, 2017, pp. 1–5.
3. H.-S. Bae, H.-J. Lee, S.-G. Lee, Voice recognition based on adaptive mfcc and deep learning, in: 2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA), IEEE, 2016, pp. 1542–1546.
4. S. Tokuno, G. Tsumatori, S. Shono, E. Takei, T. Yamamoto, G. Suzuki, S. Mituyoshi and M. Shimura, "Usage of emotion recognition in military health care," in 2011 Defense Science Research Conference and Expo (DSR), Singapore, 2011, pp. 1-5.
5. Neiberg, D.; Elenius, K.; Karlsson, I.; Laskowski, K.: Emotion recognition in spontaneous speech. In: Proceedings of Fonetik, pp. 101–104 (2006).
6. Srinivas Parthasarathy and Ivan Tashev (2018), Convolutional Neural Network Techniques For Speech Emotion Recognition, Microsoft Research
7. C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J.N. Chang, S. Lee, S.S. Narayanan, Iemocap: interactive emotional dyadic motion capture database, Lang. Resour. Eval. 42 (2008) 335.
8. LIVINGSTONE, S. R., AND RUSSO, F. A. The Ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in North American English. PloS one 13, 5 (2018), e0196391.
9. Pichora-Fuller, M. Kathleen, and Kate Dupuis. Toronto Emotional Speech Set (TESS). Scholars Portal Dataverse, 2020. DOI.org (Datacite), doi:10.5683/SP2/E8H2MF.
10. J. Lyons, “Mel frequency cepstral coefficient (MFCC) tutorial,” Practical Cryptography, 2015.