

**Rishiaharan
Thirukumaran
(001090097)**

Supervisor: **Prasad
Perera**

November 2020

**Developing a Sign-Based Educational system for Deaf/Mute kindergarten children learn to
read by themselves**

Word count: **11050**

Product and Development files:

1. Uploaded as a zip file

ABSTRACT

Deaf/Dumb kids are difficult to learn lessons when then attempt special kindergarden school and teachers also feel difficult to handle many studets at a time. Deaf/Dumb student started to learn sign they need someone to teach all sign gestures and after the learning process someone have to they can able to use correct sign gestures. In a kindergarten school teachers have to check every student can able to use correct sign gestures after the learning. If the students aren't able to use correct symbols teachers have to repeat lesson so this could make issues in class like teachers might getting bored while they check students again and again. Students losing their confidence in learning new things. The proposed system is this research made for deaf/dumb kids who needs to learn sign language selfly. This system has feature like after the learning deaf/dumb kids can able to make self evaluations so student don't need anyone to check while practice signs because proposed system has trained machine learning model and model predict sign gesture while students try new gestures. This proposed system is designed to motivate deaf/dumb kids to learn english sign language letters selfly.

ACKNOWLEDGEMENT

I have to thank many people because they supported me to complete this project successfully. First I have to thank my supervisor Mr. Prasad Perera who helps me to deliver proposed systems in an effective way. His guidance can able to implement interesting features so finally it became a value system.

I have to thank Mr. Asith Gamge who is the course coordinator of PIBT for computing programs. He scheduled supervisor meetings and arranged proposal discussion meetings on time.

Finally I have to thank my friends who supported me while I stated to doing this project

Table of Contents

Introduction	2
Literature Review	5
Review of other products	13
Requirement Analysis	16
Design	23
Development	47
Evaluation	65
Conclusion	69
Reference	70
Appendix A	72
Appendix B	77

INTRODUCTION

Background

Education is more important for everyone and education helps to manage life in a successful way. There are a lot of education systems currently available in this world and it differs from country to country. Commonly each country has Kindergarten, nursery school, primary school, lower secondary school, general upper secondary school, high schools, vocational schools, and universities are types of school for each level of students but many countries have specific education systems. There are more specific education system for blind, deaf and dumb students. Normal students are able to read and write easily and from their beginning they learnt how to pronounce words so they can easily learn to write or read words with less complexities.

Compared with normal students blind students have different methods to learn and mostly they start to learn the Braille system from the beginning and it helps them to learn further. The dumb and deaf students have a specific method to learn which is known as sign language and each country has own specific sign language. This sign language has more specific structure and also has unique characteristics. There are lot of schools and many institutions are currently available for deaf and dumb students. There are many difficulties to handle small kids because they are more aggressive and sometimes they need their parents to calm down themselves. In multicultural countries they have different people from various origins so kids from different communities or races might be treated in a bad way and sometimes small kids aren't able to report their problems anywhere. Some students take more time to learn and their teachers need to repeat lessons and some teachers might lose patience so they might misbehave with small kids.

Sign language is differ from each country because each country has its own unique sign language and each sign language has its own structure which means grammar, character and specific language structure. In general american english and british english differs from the way of speaking which are different accents same as American sign language differs from British Sign Language. Sign languages are mostly expressed by actions of face and hands but this research mainly focuses on learning hand sign gestures. This research is mainly focus on creating a solution for american deaf/dumb kids who need to learn american sign language. American sign language has unique hand gestures for all 24 english letters and numbers so this research helps to provide solution for learning letters for kindergarten deaf/dumb kids.

Objectives

The main objective of this research project is to develop a online based learning system for deaf/dumb kids with needed functionalities.

User registration.

The main users of this online based learning system are deaf/dumb kids which they need to learn and teachers to support learning process so each and every user can able to register the system before they use the system.

Lessons for Deaf/Dumb kids.

Deaf/Dumb students needs to learn how to read english letters and corresponding sign language for this letter so to archeive that goal have to include lessons which support learning their learning process.

Create a sign language to english interpreter and creating evaluation for deaf/dumb kids.

Once deaf/dumb students learnt every lessons they can able to try to evealute themselves so through this deaf/dumb able to attempt more times to learn lessons.

Maintain a record for deaf/dumb kids and their archeivements.

Every deaf/dumb student's results and attempes needs to stored and can able to monitor their progress.

Creating messaging services in this system.

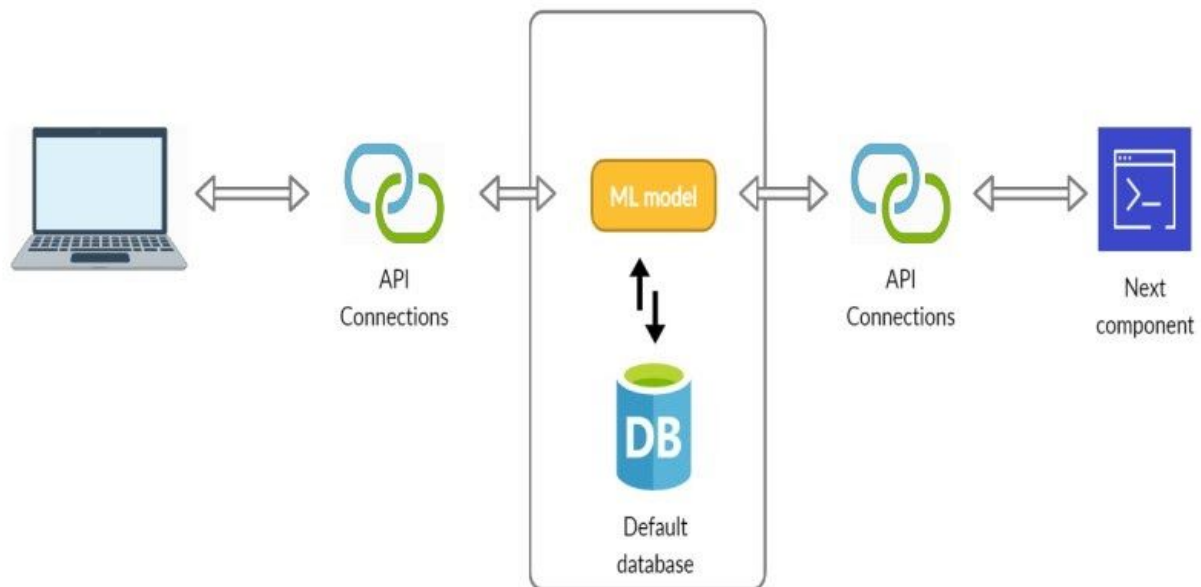
Dumb/Deaf students need to repot if they have some issues with lessons so can use messaging feature to report their issues.

Aim of the project

Developing a system to self learn processes for deaf/mute pre-school children. This process motivates students to be able to learn by themselves. Students are able to learn english words and corresponding american sign language while starting to use this online system.

An overview of methodology

This solution is a web based solution so this solution builds with flask(python micro framework), react(javascript frontend library) and SQLite database is used to store data in the database. Solution is build for deaf/dumb students to learn english sign language selfly so this system has image recognition model to predict sign language and students can able identity correct sign gesture while they got results when they try.



Literature Review

Introduction

In the United States of America there are many kids born with hearing loss according to the Centers for Disease Control and Prevention (CDC) report 2 to 3 out of every 1,000 kids are identified born with hearing loss in one or both ears(CDC, [1999-2007](#)).[1]

This research focuses more on capturing sign languages and detecting corresponding english letters for that specific sign. Sign language has many elements such as Articulation Point, Hands Configuration, Movements Type, Hands Orientation and Facial Expressions. Articulation Point includes fingers joint, wrist joint, elbow joint and shoulder joint. Hands configuration has includes types of finger bends, types of finger movements and types of palm bends. Movement types are also considered as an important factor for sign detection so following movement types are forward, backward, left and right useful in sign detection. Hands orientation is another part of sign detection which has movements like palm up, palm down, palm inward and palm sloping finally facial expressions also used as a major part in sign detection which includes happy face, angry face, lips movement and shaking the head. [3]

There are various methods used in sign language recognition which are sensor based and vision based. Sensor based technology is used to capture sign language with sensors so for this process there are some gloves used. Those gloves have some sensors which are accelerometers, proximity sensors, bend capturing sensors and abduction sensors. Bend capturing sensors are called flexion and they help to measure bend angles in fingers. Abduction helps to analyze abduction between fingers. Accelerometers help to measure orientation of the wrist which means it helps to identify roll, pitch and yaw. The main advantage of using sensor based gloves is being able to get corresponding and required data with voltage values to computing devices so this series of actions helps to process raw data data into meaningful data. There are different types of gloves with different types of sensors used for sign language recognition.[3]

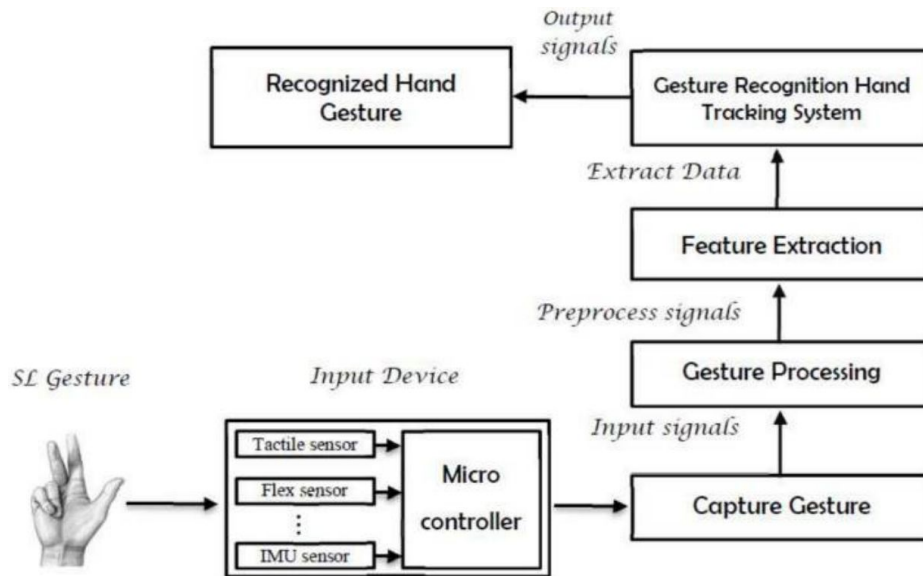


Fig 1:- Process of Sensor based Sign Language Gesture Recognition [3]

Vision-based systems is another important method commonly used in sign language recognition. Cameras help to get inputs from users in vision-based systems and these are more affordable because currently most laptops have sufficient quality web cameras. Vision-based systems have a set of procedures when they recognize sign languages using machine learning and machine learning helps to predict objects from past data analysis. Vision-based sign language recognition using computer vision to identify sign language. Computer vision is a subset of artificial intelligence and this process is more describing object recognition and object tracking. Computer vision has various steps like image processing, acquisition and object focus.[5]. In computer vision there are various deep learning neural networks currently used for object detection such as CNN, RNN and ANN. CNN and RNN are commonly used in many researches for sign language recognition.

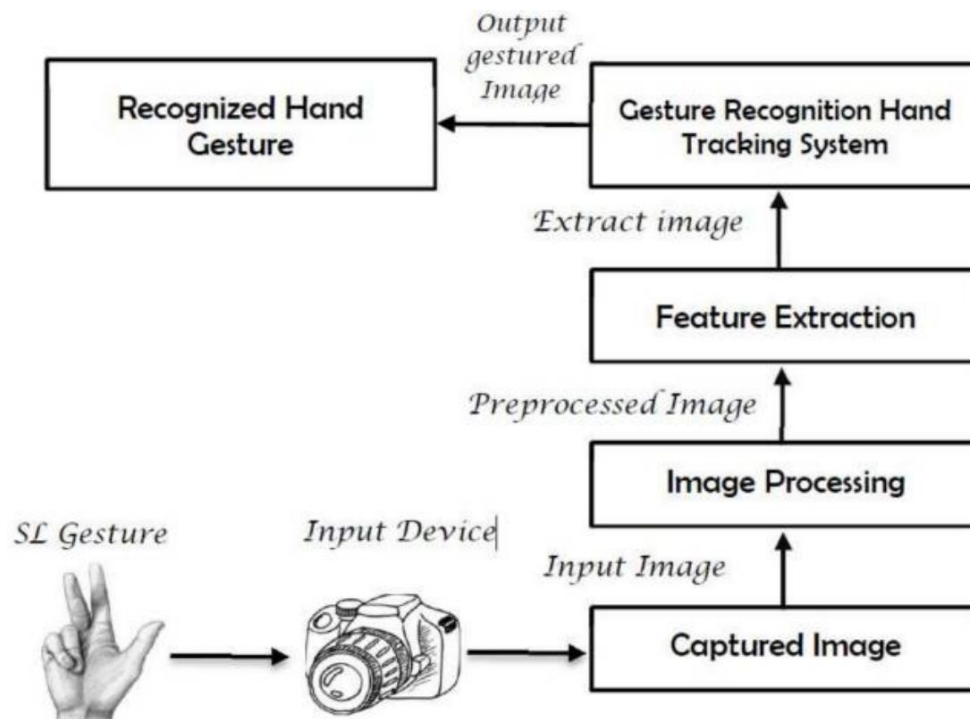


Fig 2:-Process of Vision based Sign Language Gesture Recognition. [3]

Sensor Based Detection

Loss of the ability to talk or hear exerts psychological and social effects on the affected persons due to the lack of appropriate interactions. Basically this detection system is suitable for people, who are not being able to talk or who cannot understand others language. So, these kinds of individuals utilize sign based communicational language for communication but they discover trouble in speaking with other people who don't get sign language. This undertaking expects to bring down this obstruction in communication [7]. And this depends on implementing a device, which can convert the sign language to normal speech language. This will order to make the communication take place between the quiet networks with the general public conceivable.

For this kind of purpose, the researcher is using a wireless data glove which is like a normal driving glove. And this device developed with flex sensors in full length of each finger and the thumb. Sign language using people can use this wireless data gloves to translate sign language into speech language. So, common people can identify their signs. The structure of wireless glove is shown in figure 3 below,



Figure 3: Wireless Gloves

This will use signal or gestures instead of sound to communicate meaning simultaneously combining some of these methods to detect accurate thoughts of the speaker.

- Shapes of hand
- Hands movements and orientations
- Arms movements and orientations
- body movements and orientations
- Expression of face

Wireless data glove includes, accelerometer sensor to detect the movement of the three axis direction x, y and z.

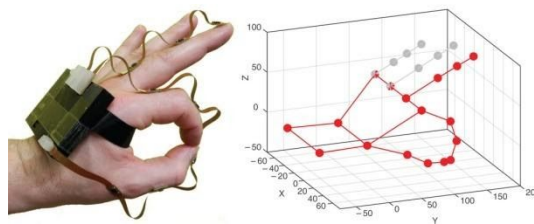


Figure 4: Axis operation



Figure 5: Sign Detection

In figure 4, detecting sensors are placed on the glove. The data will be capture through (which is shown in figure 5) the sensor and it will be sent to micro-controller. In micro-controller is used to processing and recognizing the data. Once data is recognized, the data will be sending to smart phone or other devices through the Bluetooth which is used to text to speech converting process. Here figure 5, LCD monitor is also used to mirroring recognized/ detected data from the microcontroller [8].

This will detect the actual position and motion of arms and the fingers. And sensors are flexible and bendable. All glove-based frameworks are the greater part of them are includes in on twist sensors that are direct to integrate in the texture yet offer less exactness on the fingertips and experience the hysteresis impact. Hardly any frameworks have so far developed 9D IMUs on the fingers, while none have been implemented or assessed for recognizing distinctive hand enunciations on the glove itself continuously. Here, detecting glove which can identify its wearer's hand stances and movement at a high exactness while keeping the size and measure of equipment parts to a base. This framework is planned around dealing with the information from IMUs situated on each finger as proficiently and quick as could be expected under the circumstances. This is assessed by identifying/recognizing all arm or finger shapes from the LSF alphabet. Sensor can be permitting it to be utilized as a data book section framework. We represent this by interfacing the glove to a cell phone for promptly showing the output of the glove.

Hand gesture recognition using machine learning.

Hand motion recognition will be doing a major role in human and technological interaction in the real world. Basically, the image will be detected and processed by the image of the hand and finally it will be recognized. Importantly, a small part of the image will be the hand. Most hand gesture detection technologies follow this fundamental. There are many examples in real life.

Example: Universal remote control system.

For this system mainly includes hand motion detection. Here, Using gestures and the skin tone color information segmentation of hands from the background will be detected. Likewise, the shading skin data is utilized to portion the hands in [10] for another conventional hand-based controller framework, despite the fact that for this situation the camera is initiated by a Pyroelectric Infrared (PIR) sensor to spare energy utilization. In [11], a visual hand motion interface for TVs is proposed, where the hand area is derived from the direction data figured by means of a frame channel system.

The proposed framework can legitimately detect hand motions from the entire picture without utilizing any picture district determination system. This is cultivated by a cautious plan of a profound CNN that can effectively and accurately detect hand signals acted in any area inside the picture under the testing circumstance that the hand that plays out the motion just takes up a decreased picture area with the entire picture. Much more, the picture is jumbled by background objects, among others human appearances. As a result, there are two fundamental rules that have situated the plan of the implemented deepL CNN: hand district contracting furthermore, over fitting [10][12][13]. The following sub-segments includes the implemented deepL CNN configuration, beginning by how a CNN can identify designs inserted in arbitrary areas in a picture, proceeding by tending to the area contracting and the over fitting plan limitations, and finishing with the portrayal of the proposed CNN.

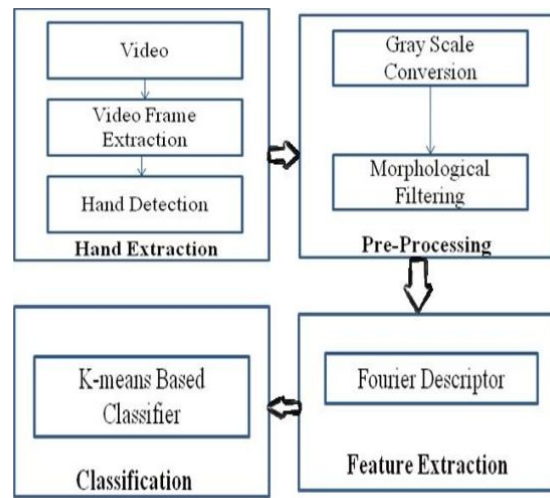


Figure 6: System Diagram

In figure 6, this system diagram uses a clustering based framework. It performs Dynamic hand gesture detection. Clustering based framework is more good classification accuracy compared to other frameworks. And here Fourier descriptor technique is used to extract the feature of this method. This technique will reduce time complexity and it will be more accurate. And this is the final output of dynamic hand gesture detection system which is mentioned in figure 6 below,

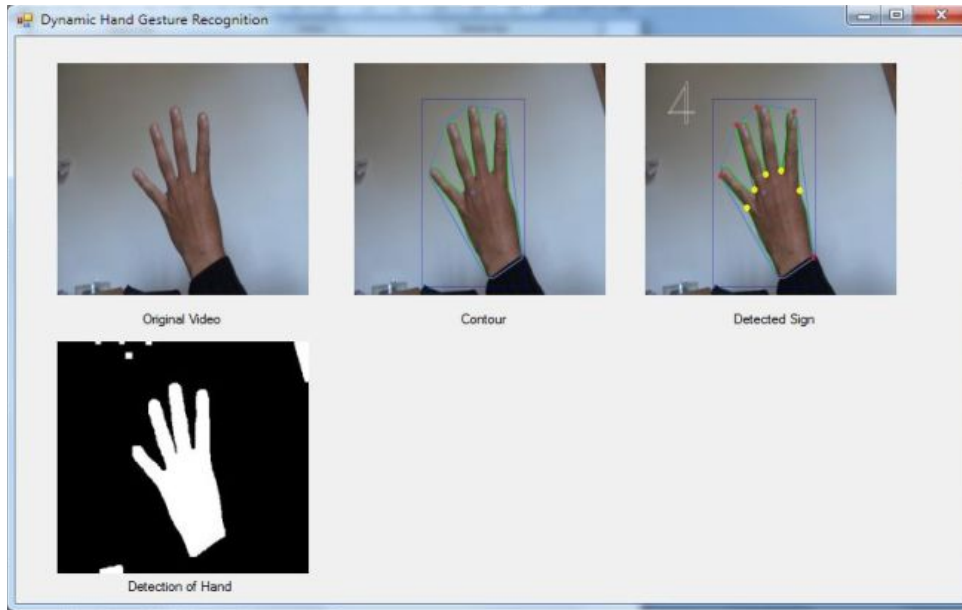


Figure 7: Hand gesture detection

A CNN has a convolution structure that limits the neural association between layers (going about as nearby responsive fields) and powers to have similar loads inside a layer. Along these lines, all the neurons associated with convolution channel tasks at one layer distinguish the very same component, yet at various confinements in the input picture. I.e., the channels are invariant to the interpretation of examples inside the picture [11]. To be specific,

$$\text{Conv}(X + X_0, W) = \text{Conv}(X, W) + X_0,$$

Here,

X- Image,

W- filter/channel,

X₀- Interpretation/translation,

Conv- Convolution Operation.

Figure 7 delineates this system, indicating the yield highlight guides of a convolution layer and some last identification.

Review of Other Products

Introduction

There are many solutions currently available now for the e-learning process and there are some e-library also available now and they are helpful for students to motivate learning. Kids education is different from adult learning because they need to learn in an interesting way so there are various web based products currently available like e-learningforkids.com, k12.com and abcy.com for kids education. Deaf/Dumb kids have to learn all those signs in their kids school so there are few mobile applications helps to learn sign language and corresponding english letter. There are some glove based solutions also available for sign to english translation.

Similar products

There are many mobile applications available to learn sign languages but mobile applications called Talking Hands and Hand Talk are most commonly known for learning sign language. There is a glove also available for sign language recognition

Talking Hands(Mobile Application)

This application helps to learn american sign language for everyone. This application contains signs representing Alphabet,Numbers,Shapes,Animals,Emotions/Feelings,Family Members,Action Words and Colors. This application has a beautiful user interface to attract everyone(including kids) and this application contains 2d images of signs. This application is currently available on android play store.

Usage:

- Able to learn american sign language signs in a structured way.
- Normal people or normal kids are able to hear corresponding voices of related signs.

Hand Talk

Hand Talk is the most advanced learning android application for learning american sign language. This application has animated 3d human avatar to express sign language that's really interesting for everyone to observe sign movements. It has a simple and attractive user interface.

Usage:

Hand Talk is able to input words and sentences as a text using keyboard and also input voice then animated 3d human avatar will express corresponding sign language for that particular word or sentence. Additionally able to rotate animated 3d human avatar so it is able to see any sign movements 360 view by rotating animated 3d human avatar.

Talking Hands.

Talking Hands is another solution currently available for sign language translation which is an innovative wearable device. This product is made by italian company called LiMiX and they have a good technical team. This kind of similar product is currently not available in the market because this is the most advanced product for sign language translation. This product must connect with an android application to get transted sign output.

Useage:

This Talking Hands solution helps to translate sign language in voice so this can be used in many situations because through this solution people don't know sign language who are able to understand sign language. Before using this solution, users need to install a mobile application so this application is available in 6 languages and those are Dutch, English, German, French, Spanish and Italian.

Key issues to use in the design and implementation

All these three existing solutions are based on sign language but there are many pros and cons in these three solutions. Talking Hands Mobile based application used to represent 2D images for sign languages so this might be difficult sometimes to learn sign language because students need to know the exact position of fingers. Furthermore this application has audio sound while views each letter and words so basically this is not primary for deaf kids too. In Hand Talk mobile application user needs to type the words or can able to input voice then animated 3d human avatar will execute corresponding sign but many kids can't able to type words and deaf/dumb kids can't able to input voice in Hand Talk application. LiMix's Talking Hands wearable solution is perfectly fit for adult hands but this wearable kit is not suitable for kids because these wearable are large for their hands. The goal of this research is providing a system to self learn processes for deaf/mute pre-school children but they need support while they use Hand Talk and Talking Hands mobile applications because they might have confusions of finger positions so they need someone who can check after learning they are able to use right symbols in real life.

Conclusions

Talking Hands and Hand Talk mobile applications are better options for learning american sign language so to solve these kinds of problems the proposed system is helpful to evaluate after they learned all letters in american sign language. The proposed system contains tests which help to try american sign hand gestures for corresponding english letters so through this system until they are familiar with american sign language. The proposed system is suitable for every deaf/dumb kids' hands which means various hand sizes can be used for evaluation because LiMix's Talking Hands solution is not fit for every kid's hands. Finally proposed helps to practice sign language in an interesting way and the system has a nice user interface which is easy to use.

Requirement Analysis

Requirement analysis is the most important part to identify needs of the research so different kinds of data collection methods are used to accomplish this research successfully. Interviews, Surveys and Observation are the main data collection methods used in this research.

Interviews

Interviews are useful to identify problems and are able to deliver the solution based on needs of the user. Before developing the proposed solution interviews were conducted with kindergarten teachers and some parents of deaf kids.

Results from kindergarten teachers.

Most of the students are able to pay attention upto 30 minutes while teachers conduct lessons but some students can't understand the lessons from the beginning so they need additional time to understand lessons due to this reason sometimes lessons take more time. Other students who already understood the lessons get bored when the same lessons are repeated. Many students like the lessons if teachers are presented with good and attractive visuals and many students like to learn things while drawing or try out something themselves. Especially students are happy when they proved can able to do something to teachers.

Results from deaf kids' parents.

Deaf/Dumb kids always like to learn things observing and trying out by themselves but they are always looking for something interesting. Most of the students are unable to sit down in the same place for a long time and they always like to move. Most of the students are interested to learn when lessons are visually presented well. Some students are mentally got upset when they feel difficult to understand lessons and teachers also mistreat them when they make mistakes or students miss to observe lessons. Most of the deaf/dumb students have

Observation

Observation is another effective requirement gathering method. Major things can be observed in kindergarten and following observations are really helpful to identify users' need. Teachers are getting bored while they repeat lessons and sometimes some teachers are getting angry with students . Some students are not listening to lessons like drawing on a desk while the teacher

conducts lessons and folding papers from their notebooks. All the students pay attention to lessons while teachers act like animals when they conduct lessons about animals.

Surveys.

Surveys are another data collection method to identify user requirements and problems. Survey forms are provided to kindergarten kids and needed to read it loudly because kindergarten kids cannot read by themselves. The survey form contains 8 questions which support identifying students' abilities and disabilities.

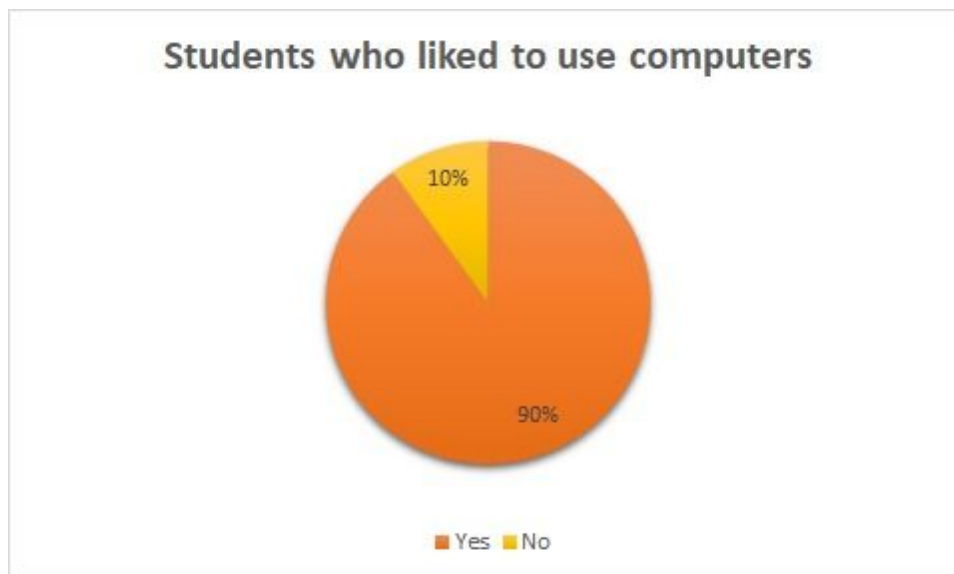


Figure 8: Question 1

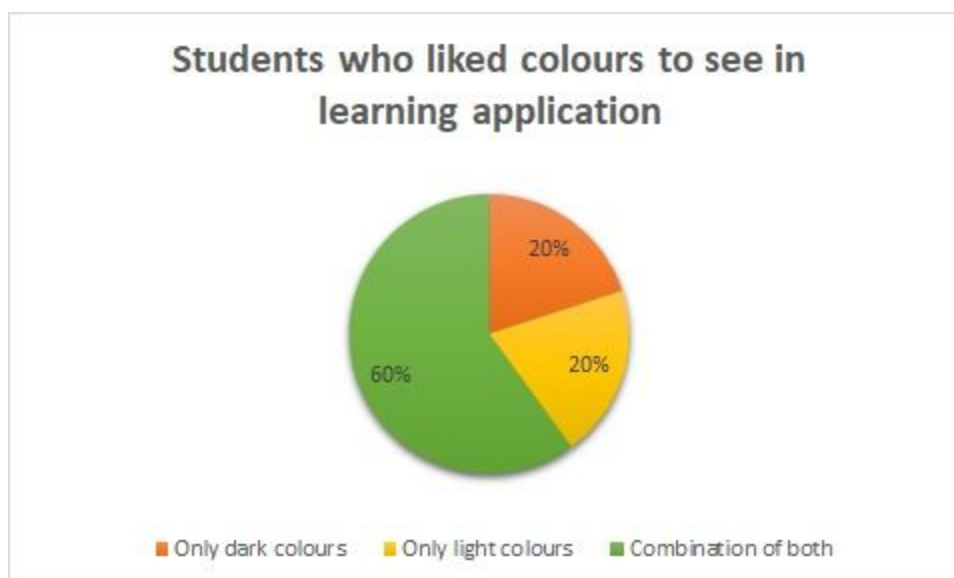
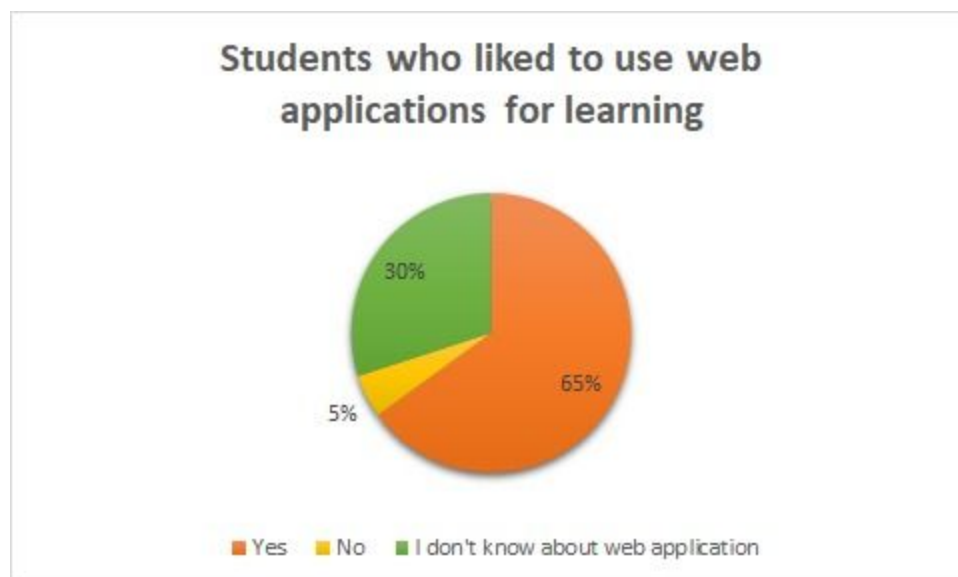
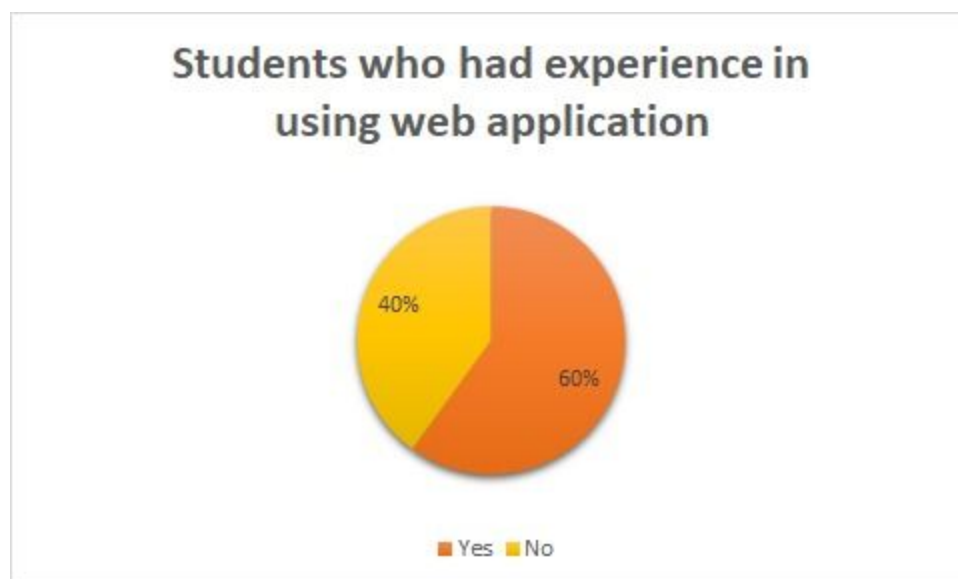


Figure 9: Question 2*Figure 10: Question 3**Figure 11: Question 4*

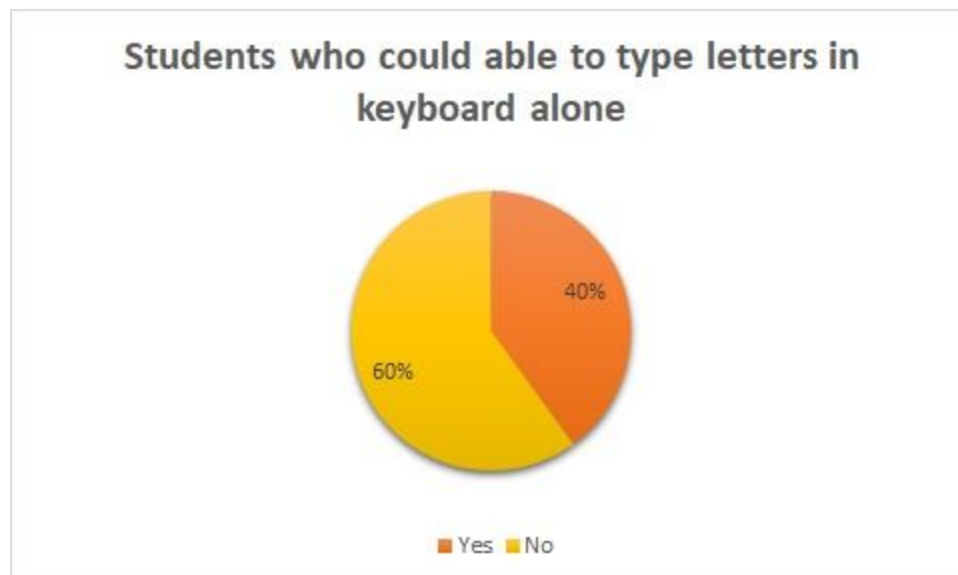


Figure 12: Question 5

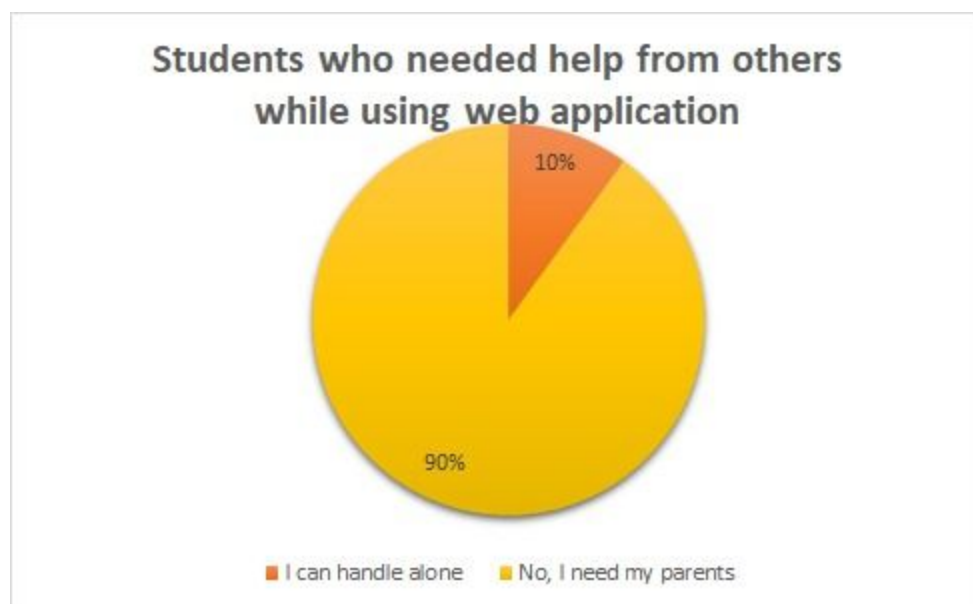


Figure 13: Question 6

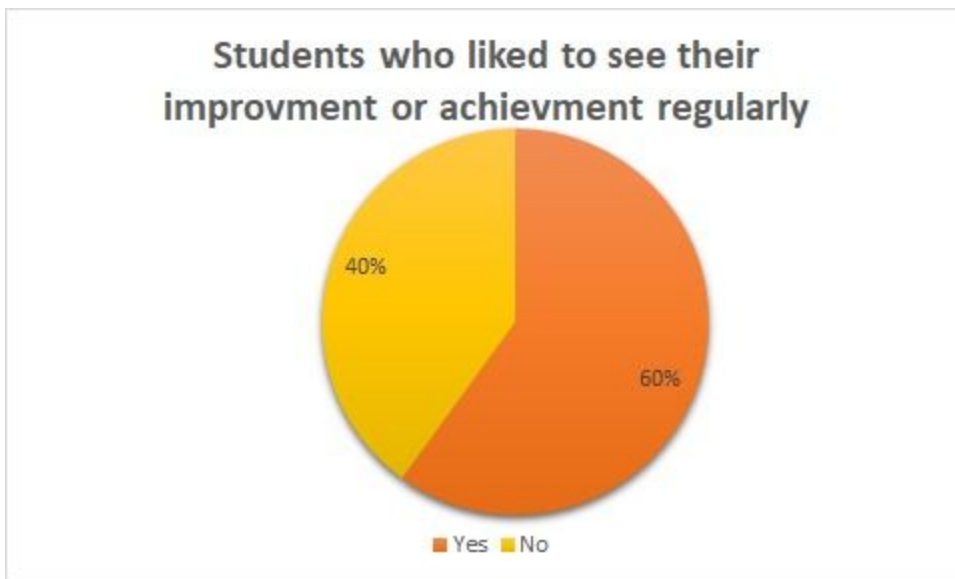


Figure 14: Question 7

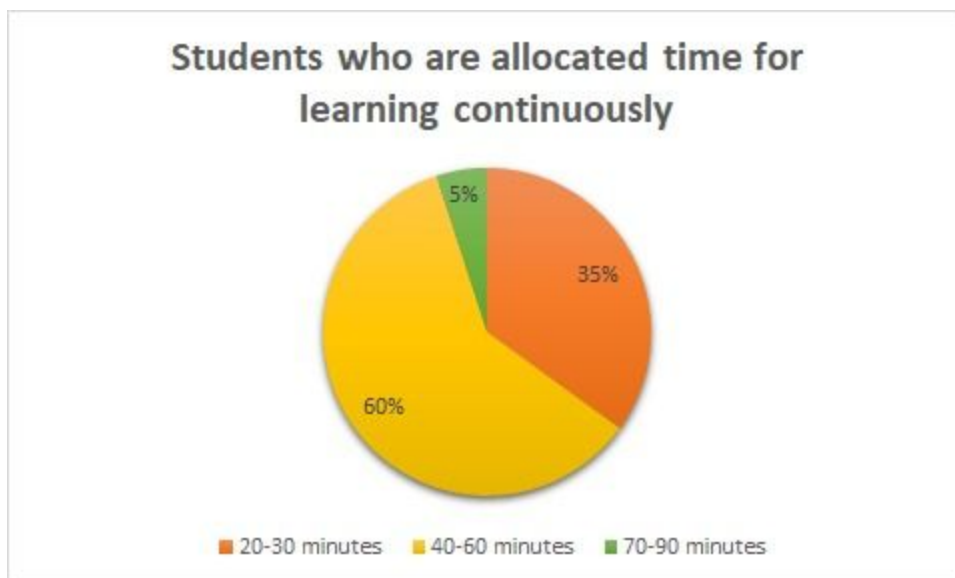


Figure 15: Question 8

Creation of effective solution

Data collection helps to identify users' issues and requirements so implementation of the proposed solution needed to satisfy users. Survey forms are provided to kindergarten kids and results of the survey useful to scheduled user training and preparing user manuals. From interviews can able identify deaf/dumb students have face issues when they are not able understand lesson within a limited time and teachers also getting bored when they repeating lessons so to mitigate these problem proposed solution must have feature students can able to learn until they familiar with sign gestures and can able test themselves when they learnt sign gestures for english letters. In a deaf/dumb kindergarten class room teacher needs to check students often and make sure their finger positions too. The creation of the proposed solution must include detecting correct sign gestures for corresponding english letters. From the results of the survey most of the students like to see their progress so the proposed system must include features to store their attempt results and through this deaf/deaf students are able to view their progress after every attempt. Before deaf/dumb students use the proposed sytem they required to register but students can't do this process alone so parents need to make registration for each students. There are some parents not able handle the web application so before implementation have to provide user training to use this system. Once parents successful completed registration they have provide username and password to login proposed system but deaf/dumb kids can't able to do this process often so to overcome this issue password and username can able stored in a browser and proposed system's url needed to bookmarked this process needed ro done by parents. After saving passwords,username and bookmark of the url of the proposed application student can able to access them when they like to learn and the main purpose of the proposed system is to created self learning deaf/dumb student so these process helps to use the system without any interruption. Students like to send messages to teachers so sign gustes need to be converted to letters in the proposed system so this process helps to get support from teachers if they have any issues.

Functional Requirement

Following functional requirements are motivated to design and develop proposed solution.

- 1.Students parents and teachers need to register before using the system.
- 2.Students' parents need to change their details when they want
- 2.Stduents need to learn english sign letters so they need english letters with corresponding design.
- 3.Students need to try signs with their hands by themselves to learn english sign language letters.
- 4.Students need to see their process so they store their scores after they try sign gestures.
- 5.Students need to send messages using signs to teachers when they need to communicate with teachers.
- 6.Teachers need to view students' messages.
- 7.Teachers are able to delete messages.

Design

For any solution designing part is also considered as the most important part and to design a solution have to follow the set of procedures. There are various methodologies currently used for design software solutions and there are many diagrams used for software design. Data Flow Diagram and Entity Relationship Diagram are used for designing processes of any software solution. User experience design, database design and model design were done to accomplish research.

Methodology

Available methodologies and selected approach

In the software development process there are many methodologies followed to develop a software based solution. Waterfall, Agile, Rapid Application Development, Spiral and Scrum methodologies are used to develop a software solution. Each methodologies have advantages and disadvantages so before choosing a methodology have to consider advantages and disadvantages which could be suitable for development of software solutions. After making a clear analysis, which helps to choose a correct software development methodology to complete this proposed solution.

Methodologies which are used in software development.

Waterfall

Waterfall methodology has 6 important phases and those are requirement gathering and analysis, system design, implementation, testing, deployment of the system and maintenance. This methodology is always considered as a traditional methodology and this methodology required to the previous phase must finish before the start of the next phase. In this waterfall methodology outcome of the previous phase is input for the next phase. Waterfall methodology takes more time to deliver a product in a limited time.

Agile.

Agile methodology is useful because it's allows to continuously adapt based on user needs. The proposed solution is developed with agile methodology because this system needs to develop incrementally and have short time to deliver the project. During user training can able get the output can able make the changes according to the user

Platform

Choosing the correct platform is the most important part for each software solution. Software applications are specifically created for many platforms which are desktop, mobile and web based applications. Computers, tablets and smartphones have different operating systems so it's really difficult to create an application for each platform which means it contains more time and more effort. Web applications are suitable to use a particular application in any device which means it is able to be used in any operating system so web application is more useful for proposed solutions also. Web Browsers are more important to use web applications so there are various web browsers currently available now. Google chrome, Mozilla firefox, Safari, Internet Explorer, Microsoft edge and Brave are well known web browsers.

Client-Server Architecture.

Most of the web based applications are based on client-server architecture. Client-Server architecture is a distributed computing model which is about data communication between Client and server. Client is able to request to access the data and client needs to wait until getting the reply from the server so through this process client can access the data from the server once the server accepts the client's request. The proposed system has a client-server model for data communication.

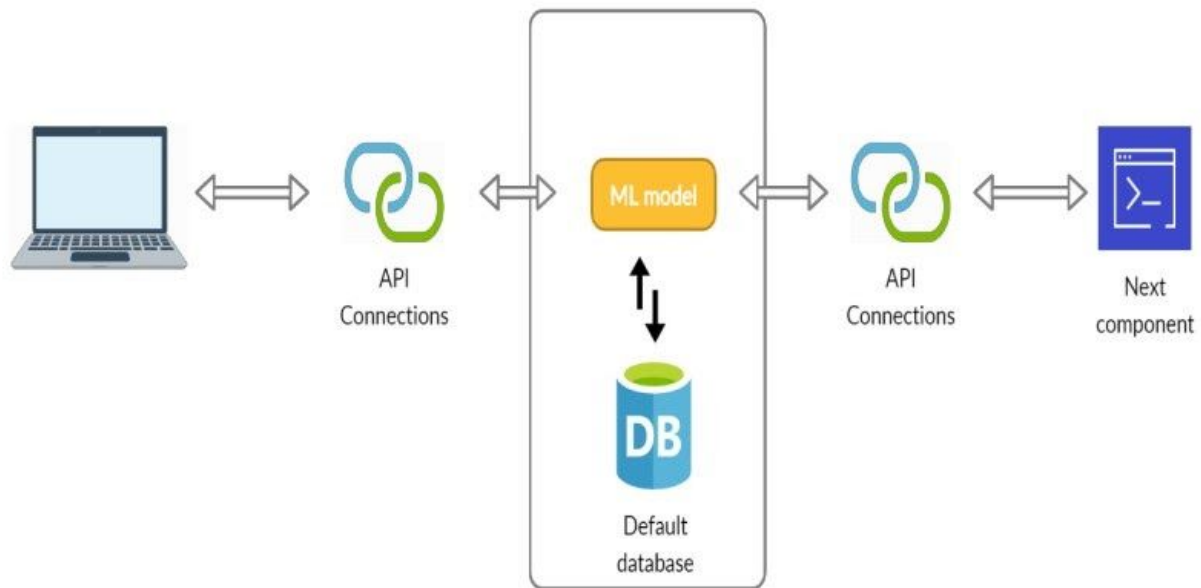


Figure 16: Architecture of the proposed system

User Interface Design

User interface design is an important process in every web based application and before design user interfaces have to consider how users are able to use web applications without any hesitations. User interfaces should be simple and easy to use so proposed solution's user interfaces created for easy to use and which have forms and buttons. Forms are used to get user input from users and buttons are used to submit details in forms. The users of the proposed solution are deaf/dumb kids so user interface designed with attractive elements

Student Register

[STUDENT](#) | [TEACHER](#)

Student SignUp Form


Name	<input type="text"/>
Year	<input type="text"/>
Email	<input type="text"/>
User Name	<input type="text"/>
Password	<input type="password"/>


[CREATE](#)

Figure 17: Student registration

Student Login

[STUDENT](#) | [TEACHER](#)

 sample



[LOGIN](#) [SIGNUP](#)

Figure 18: Login

Navigation Bar

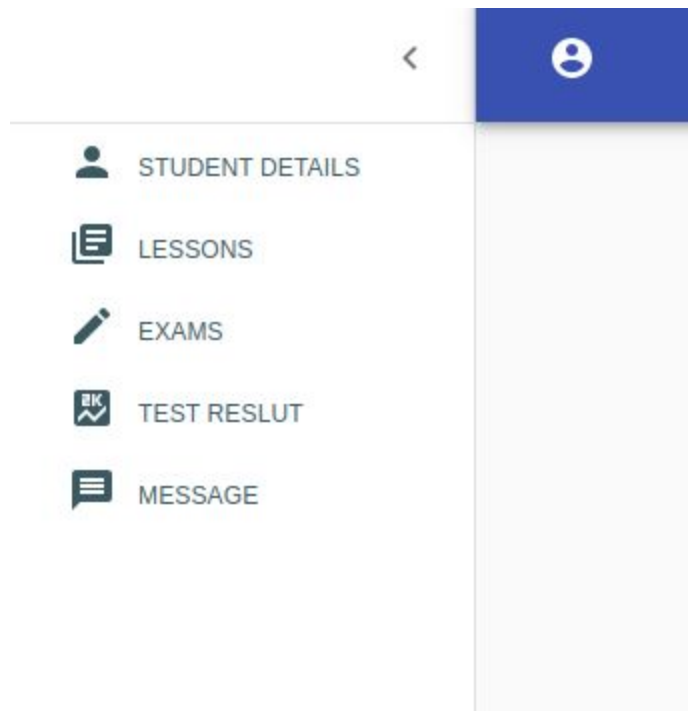


Figure 19: Navigation Bar

Student Details

Student Details Form


User ID	<input type="text" value="1"/>
User Name	<input type="text" value="sample"/>
Name	<input type="text" value="example student"/>
Email	<input type="text" value="example@email.com"/>
Year	<input type="text" value="1"/>
Password	<input type="password" value="....."/> 
Created On	<input type="text" value="2020-11-01 04:41:02"/>
Updated On	<input type="text"/>

Figure 20: Student details

User Interface for learning sign gestures

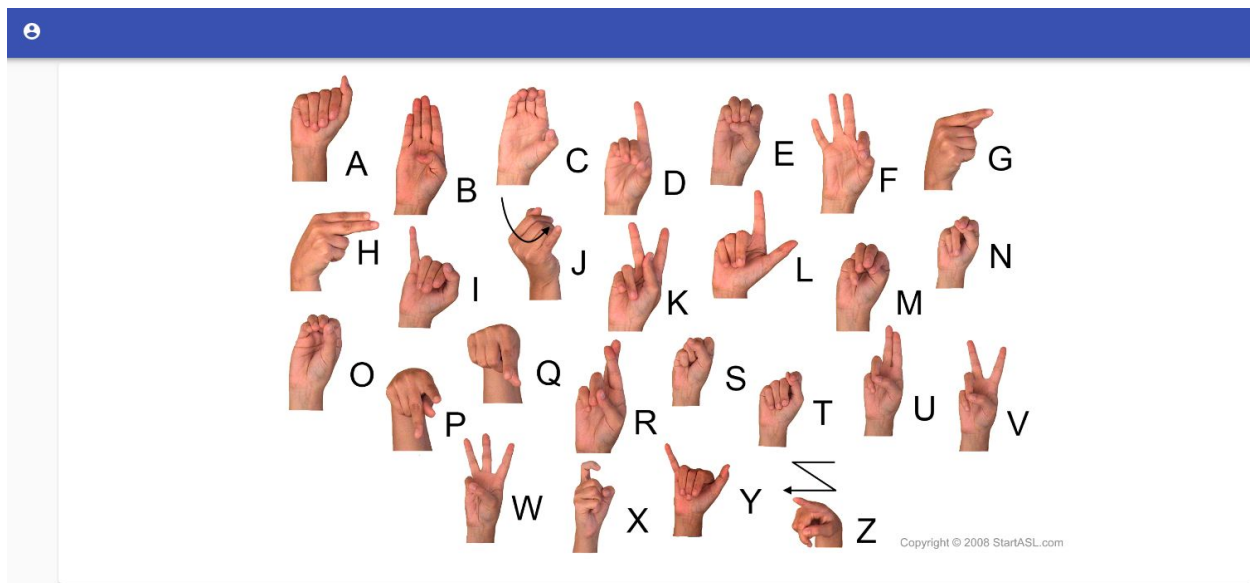


Figure 21: Learning user interface

Test user interface to try sign gestures.

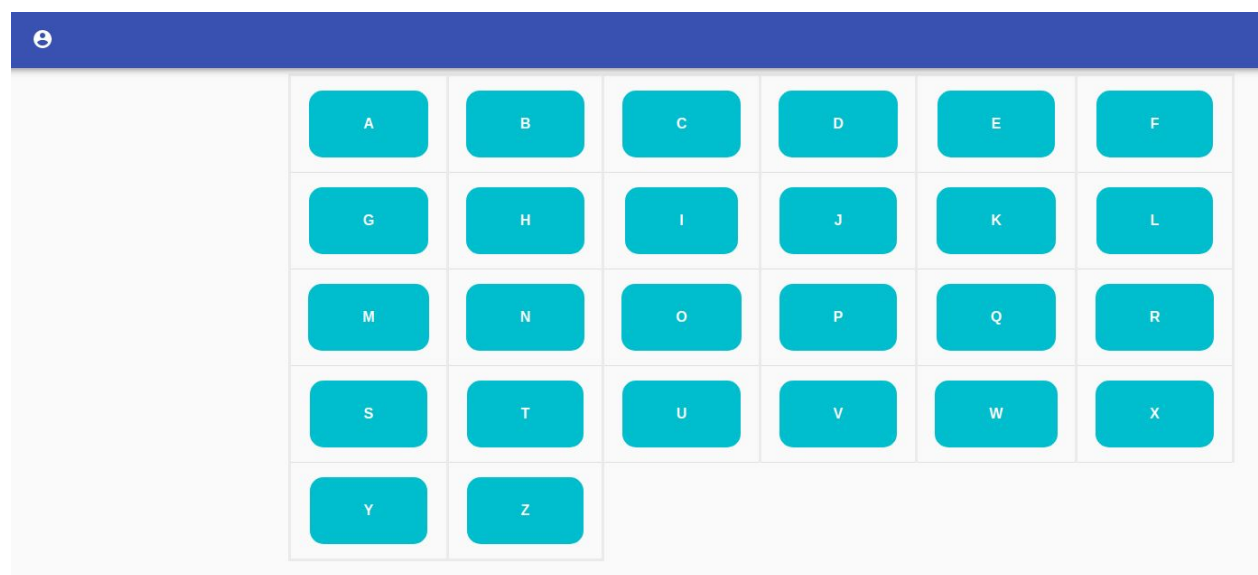


Figure 22: Test user interface

Camera to capture sign gesture

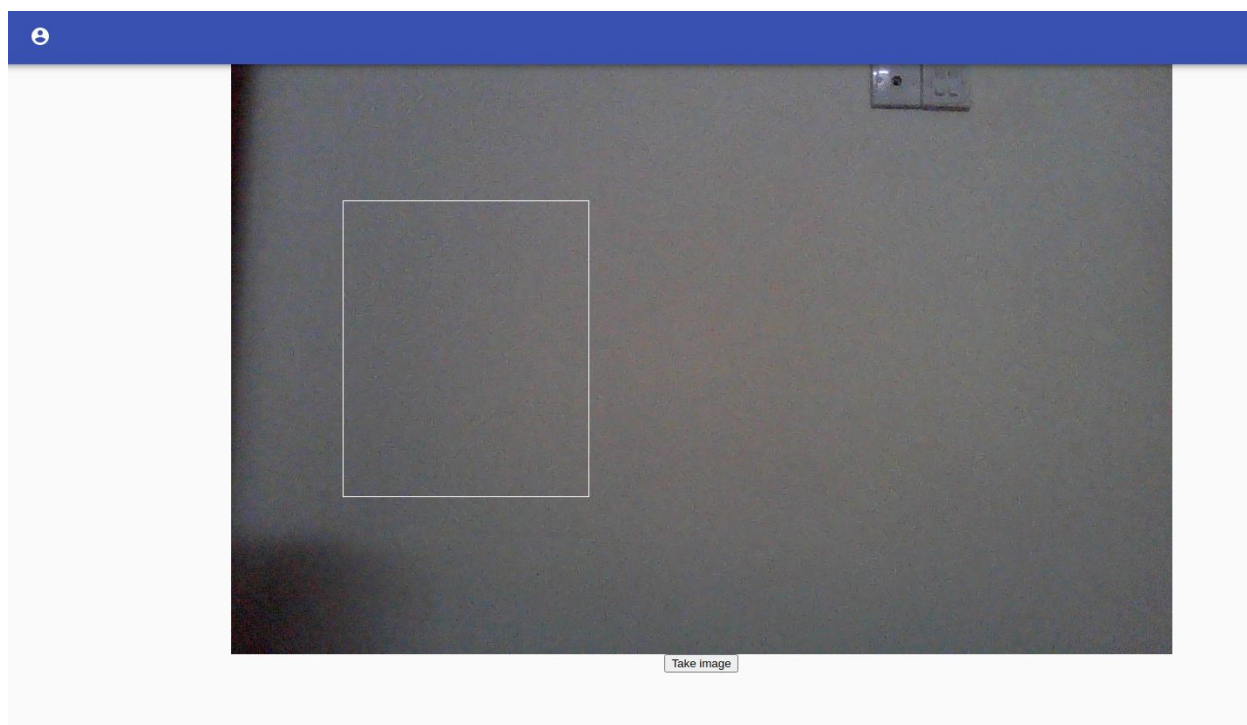


Figure 23: Camera user interface to capture

Result is loading

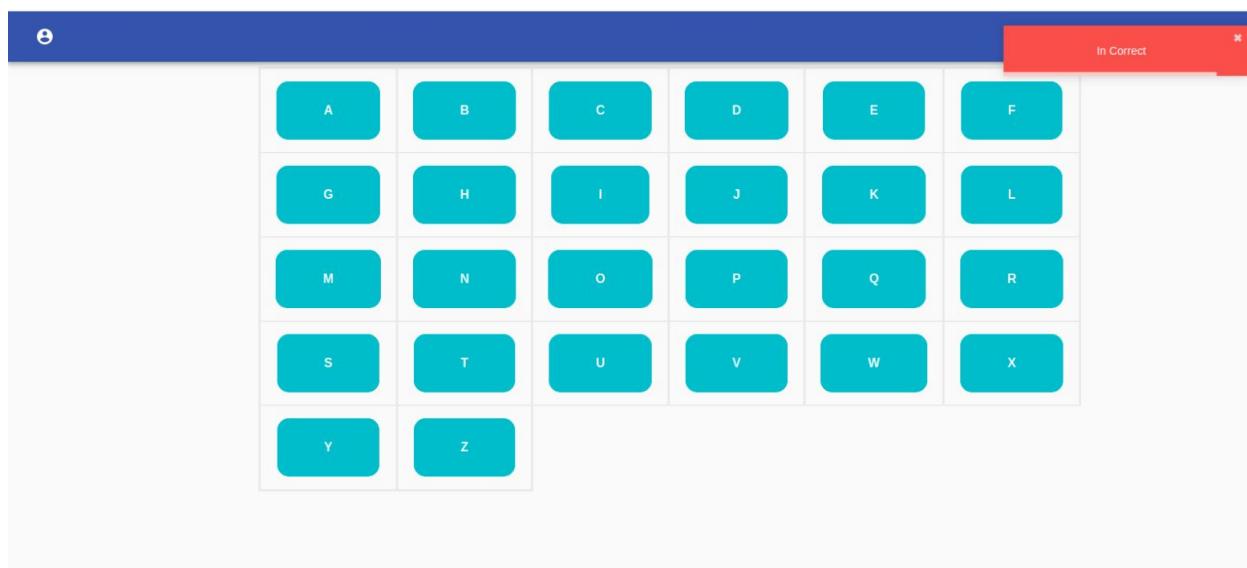
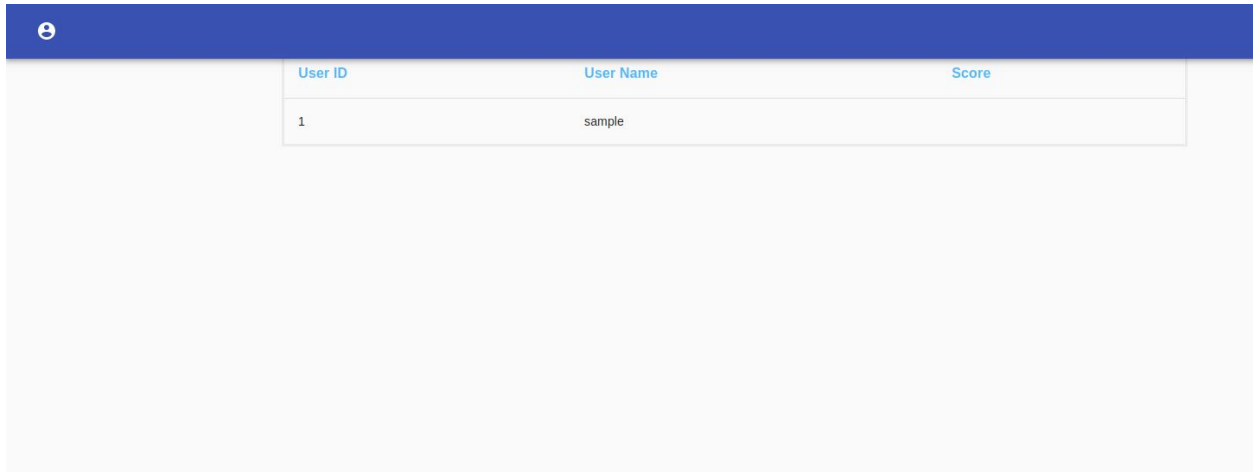


Figure 24: Result is loading after the attempt

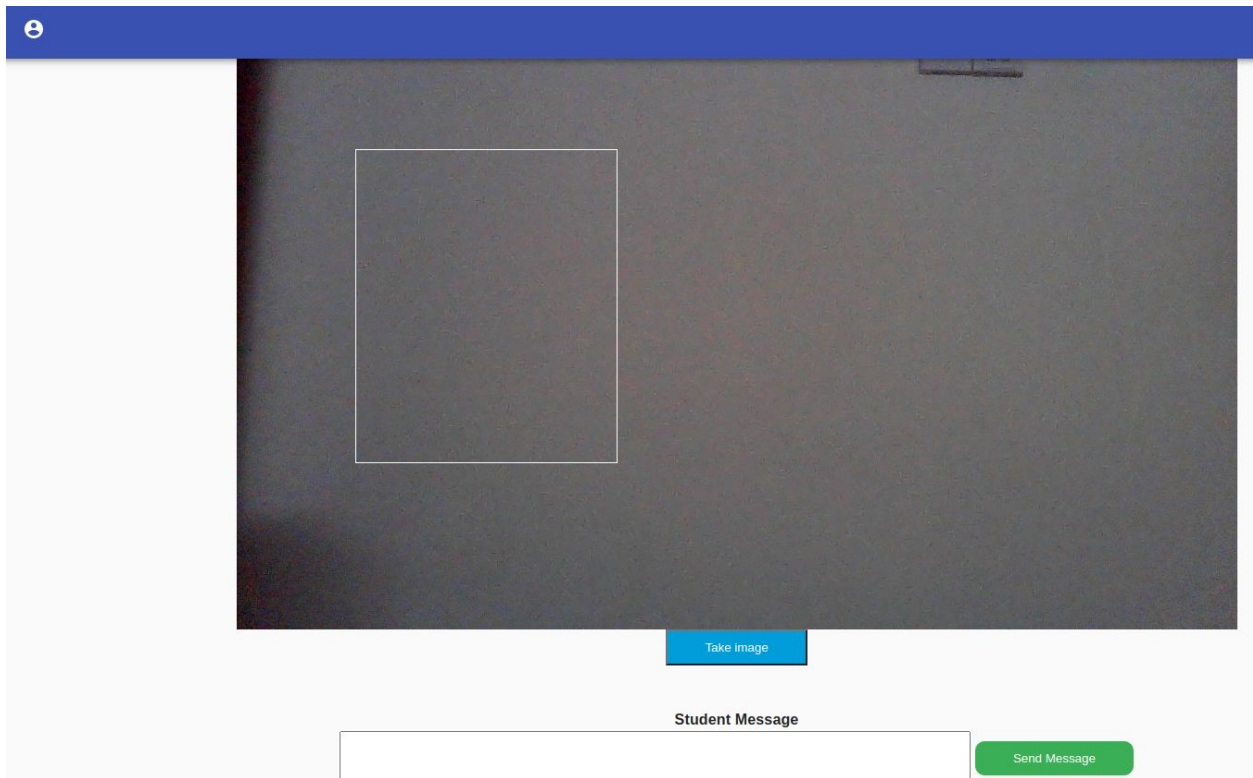
Store test results



User ID	User Name	Score
1	sample	

Figure 25: Test results

Send Message



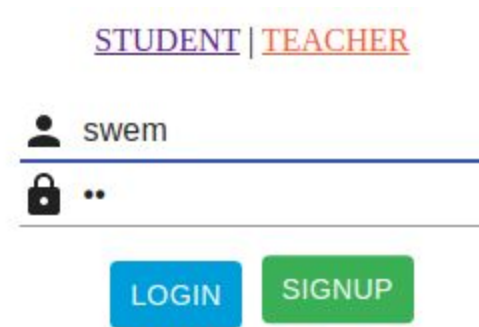
Take image

Student Message

Send Message

Figure 26: Send Message User Interface

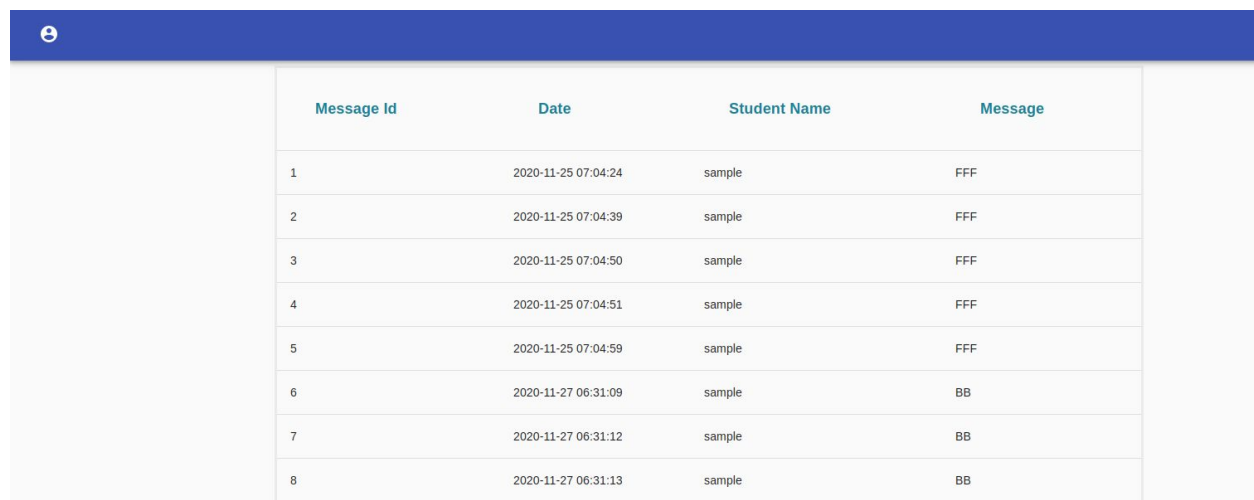
Teacher



The interface shows a header with 'STUDENT' in purple and 'TEACHER' in orange, separated by a vertical line. Below this is a user selection area with a person icon and the text 'swem'. A horizontal line separates this from a lock icon and two dots. At the bottom are two buttons: a blue 'LOGIN' button and a green 'SIGNUP' button.

Figure 27: Teacher login

Show Messages



The interface features a blue header bar with a white circular icon containing a plus sign. Below the header is a table with four columns: 'Message Id', 'Date', 'Student Name', and 'Message'. The table contains eight rows of data.

Message Id	Date	Student Name	Message
1	2020-11-25 07:04:24	sample	FFF
2	2020-11-25 07:04:39	sample	FFF
3	2020-11-25 07:04:50	sample	FFF
4	2020-11-25 07:04:51	sample	FFF
5	2020-11-25 07:04:59	sample	FFF
6	2020-11-27 06:31:09	sample	BB
7	2020-11-27 06:31:12	sample	BB
8	2020-11-27 06:31:13	sample	BB

Figure 28: View Message user interface

Database Design

Database helps to store details in application so databases contain different tables to store data. Proposed solution has tables for store data which are a message table, student table, student exam table, user table. Student table is created to store student details and the student table has the following columns like id, student, user, message, created_on and updated_on tables. User table is created to store user details which has id, username, password, name, email, role, created_on and updated_on. Message table is created for store messages which are from students like to try with sign language and send to the teacher so the table has id, student, user, message, created_on and updated_on. Student exam table is created to store scores once they tried to show corresponding signs for english letters. Student exam table has an id, exam, student(id), correct, incorrect, created_on and updated_on.

REST API Design

Application Programming Interface which has a set of standards to communicate between applications. Each web application has a back end so there are many API architectures that help to develop backend parts in web applications. Following API architectures are used for API creation REST, SOAP and GraphQL. The proposed solution has developed with REST API architecture which has HTTP methods to handle data. GET, PUT, POST and DELETE methods are used in REST API.

GET- This method is used to get the data from the resource.

PUT- This method is used to update existing data from resources.

POST- This method is used to insert new data to a resource.

DELETE- This method is used to delete existing data from a resource.

API endpoints are created to manipulate data for different resources and endpoints are Uniform Resource Locators (URL). Following endpoints are used in proposed solutions to manipulate with data.

url/predict-Created for making predictions and out the results.

url/letterpredict- Created for conversion of gesture to letter for message feature

url/user- Created for create and manipulate teachers' data

url/messenger- Created for send and receive messages

url/login- Created for teachers to use the proposed system

url/student/login-Created for students to use the proposed system

url/student/exam- Created for maintaining test scores in the proposed system.

url/student-Created for create and manipulate students' data

Image Classification Models for prediction.

Machine learning is the process machines learn from data and make decisions if new data arrives. In this process algorithm design is the most important part so statistics and programming helps to design effective algorithms for machines. Learning algorithms can be 3 various types which are supervised learning, unsupervised learning and reinforcement learning. Supervised learning is a process machine learning everything under a set of guidance so this process labeled data is helpful to identify how output comes. Unsupervised learning is a process machine that doesn't get any support or guidance with feeding data and also there is no labeled data but machines need to identify hidden parts and make predictions according to that. Python and R programming commonly used for creating machine learning algorithms.

Deep learning and Neural Networks

Machine learning is a superset of deep learning and deep learning helps to solve real world problems effectively. Usually deep learning needs a large data set to train models and deep learning concept came from as an inspiration for human brain structure. Human brains are a combination of neurons and the way the human brain transfers the information is passing one neuron to another neuron so from inspiration of human brain architecture neural networks also have neuron architecture to transfer the data. Neural networks have several layers and each layer has a set of neurons and each previous layer neurons are connected with next layer neurons. Neural networks have input, out layer and have hidden layers. Input layer helps pass individual features from the dataset to pass the model. Hidden layers receive output of the input layer as input and connections between each neuron with input layer have their own weight. Weights are the strength of the connection between one layer to another layer neurons. Output layer identifies class and output data corresponding class. Output layer is the last year and input is the first layer in a neural network. Hidden layer is located between input layer and output layer. There are various types of layers currently available and convolutional, recurrent, dense, normalization and pooling layers are useful layers.

CNN

Convolutional Neural Networks have been proven to work well for computer vision problems. Several researches have shown that CNNs outperforms other approaches such as SVMs for computer vision problems such as classification, which is the kind of problem that is dealt with in this research.

CNNs are mainly composed of Convolutional Layers and Pooling Layers. These layers specialize in computer vision tasks. Convolutional Layers are composed of a filter which is applied on the input image which results in a feature map. The filters focus on different features on the image. Each filter results in a different feature map and the feature map consists of the features detected by the filters. CNN also applies a padding function which decides the shape of the output from each Convolutional layer. If the same padding function is applied, the output shape will be the same as the input shape. If full padding function is applied, the output size increases as additional pixels are introduced into the input data. For this research, the same padding function is applied as it is proven to perform better and also reduce the amount of computational power required to train a model.

Pooling Layers are usually placed after a Convolutional layer and they generally pick the Average or Maximum value from each of these feature maps and pass them onto the subsequent layers. In this research the type of Pooling Operation that is applied is the Max Pooling operation which results in the maximum value from each feature map.

BatchNormalization layers are used to make the Neural networks more stable by normalizing the data that passes through the network. In this research the Batch Normalization layer is introduced after all the Convolutional and Pooling layers to normalize the data.

Flattening layers are always present in a CNN architecture after all Convolutional layers and before the Dense layers. The Convolutional layers generally work with multidimensional data (usually 2D) while the dense layers only work with single dimensional data. Therefore, the flattening layer is important as it flattens the data into a single dimension allowing the classification of the data using dense layers. While the convolutional layers are mainly used to learn the features in the image, the dense layers are used for the purpose of classification. There can be several dense layers in a model. The time taken to train a model increases with the number of dense layers and number of neurons per dense layer.

Dropout layers are used to reduce overfitting in a neural network by randomly turning off neurons in a layer. An overfitted model will always perform poorly when new data is presented

as the model has learnt the training data too perfectly. When building a network, the percentage of the number of neurons that can be turned off will have to be mentioned programmatically. In this research, dropout layers have also been used. When the dataset is smaller, the chances of a CNN overfitting on the data increases and therefore, including some steps to prevent overfitting is the best approach to ensure a good model with high accuracy and generalizability. The last layer of the network is always the layer that will perform the classification into the different classes that the network is supposed to be classifying. The number of neurons in the last layer (a dense layer) is the number of classes that the network will be classifying the data into. Each layer, both Convolutional and Dense, will have an activation function. The activation function's purpose is to get an output from a neuron in a layer of a network. The output of a neuron generally falls within the range 0 - 1. Activation functions can be either linear and non-linear. For most purposes, linear activation functions are not used as the input is exactly mapped to the output. Non-linear activations are most suitable for tasks such as the one dealt with in this research. Some of the activation functions that exist are ReLU (Rectified Linear Unit), LeakyReLU, Sigmoid, and Softmax. ReLU is the most used activation function in CNNs, The last layer also has an activation function but since the purpose is to classify an input into a specific class, it is more ideal if the output of the last layer can be a probability value which can determine which class the input data can be classified into. For example if the last layer has 4 neurons, each neuron will output a probability that will range between 0 and 1 while the total of all the probabilities should be equal to 1. The neuron with the highest probability will be taken as the output classification for the input data. This operation can be carried out by softmax activation function and therefore this is the activation function that is used with the last layer of the network.

After building a model by defining each of layers of the network, it is important to compile the network. The compiling of the model is the point where the loss function and the optimizer is defined for the model. These are very important before training the model. The loss function's output will be used to improve the model over the several iterations through which the model will be trained. The optimizer is the function that will be used to change course of the model during the training process. The optimizer will be used to adjust values such as the weights of the individual neurons in a network. The optimizer works in such a way that it minimizes or reduces the loss. The lower the loss, the better the model. The loss indicates the error of the model. The metrics used to measure the performance of model can also be specified in the compiling step. The metric chosen for this research is accuracy. But there are other metrics that can be used such as Precision, Recall, F1-Score. Accuracy was chosen as this seemed ideal for this research. The data used in this research was well balanced across all the classes and therefore using accuracy directly will not result in a skewed perspective of the performance of the model.

CNNs can be trained by using two main approaches: Training from scratch and transfer learning. Training from scratch is where the weights and bias of each of the neurons are randomly

assigned when the training commences. They are then learnt when the training commences. This can take longer and usually requires a quality dataset which consists of a large number of images. The architecture of the CNN can be defined by ourselves or an existing well proven architecture can be used.

The other commonly used approach is Transfer Learning. This is an approach where a pre trained is used instead of training from scratch. The weights and bias that have already been trained will be used to initialize the model instead of randomly initializing these values. Usually the weights are learnt from a much larger and generalized dataset. The weights are then used to train on a more specific dataset that doesn't consist of as many images as the original dataset. The most commonly used weights are the weights learnt from the dataset called ImageNet. For this approach, it is common to use architectures that are well researched and proven (Ex:- VGGNet, ResNet, Inception).

Data collection

It is important to pick high quality data for training a CNN as inaccurate data can result in inaccurate and inconsistent results from the trained model. Also the quantity of the data is important when Neural networks are used as they are data hungry. Neural Networks are shown to perform well only when large amounts of data are used to train. In the absence of data approaches such as Transfer Learning (as discussed earlier) can be used.

The two main datasets that was considered for this research was the Sign Language MNIST dataset and ASL Alphabet dataset. From the two datasets, the American Sign Language Alphabet dataset was picked. The MNIST dataset only consisted of the 26 letters in the English alphabet while the ASL Alphabet dataset also consists of three additional signs: space, backspace and nothing. This allows the user to add spaces in their text and also undo any incorrect letters that have been added. For a more usable implementation of a system that allows the use of Sign Language to type, such signs are extremely important. Also, the ASL Alphabet dataset consists of more data, 87,000 images for training across 29 classes. Therefore, this dataset is more ideal for the classification problem that this research will be dealing with.

The dataset was obtained from kaggle, a portal that consists of many different types of datasets for all sorts of purposes. <https://www.kaggle.com/grassknoted/asl-alphabet>

The dataset only consists of 29 images in the test set, promoting the use of real world images for the testing of the model. As you will see, our model performed exceptionally well in the training and testing phase and also performed well in real world images but not as well as it did for the images in this dataset.

Preprocessing

Preprocessing data is an important step that is carried out before data augmentation (if required) and training an algorithm. This step is usually carried out to enhance features and improve data quality. Also, a common step that is included in the preprocessing step is image resizing which is generally used to reduce the size of the image. Although larger images can result in a higher accuracy (upto a certain image size), there is a huge reduction in performance and increase in training time. For datasets such as this, preprocessing approaches can be used to only highlight edges and generate histograms which can then be applied on the image before being fed to the image to either train or predict. But these histograms don't usually generalize well.

The following code shows the loading of data and preprocessing of the data carried out as part of this research. Minimal preprocessing was carried out for this research to improve generalizability.

```
# Dictionary to Store all the labels for each of the class.
labels_dict = {'A':0,'B':1,'C':2,'D':3,'E':4,'F':5,'G':6,'H':7,'I':8,'J':9,'K':10,'L':11,'M':12,
               'N':13,'O':14,'P':15,'Q':16,'R':17,'S':18,'T':19,'U':20,'V':21,'W':22,'X':23,'Y':24,
               'Z':25,'space':26,'del':27,'nothing':28}

"""
Load the data to carry out the preprocessin tasks.
Initially the images are loaded from the training directoy only.
These images are loaded, resized to (64,64) and stored in a list which is then
converted into a Numpy Array. The Array is standardized by dividing it by 225.
The dataset is then split into training and testing sets - Images and Labels.
"""
def load_data():
    images = []
    labels = []
    size = 64,64
    for folder in os.listdir(train_dir):
        print(folder, end = ' | ')
        for image in os.listdir(train_dir + "/" + folder):
            temp_img = cv2.imread(train_dir + '/' + folder + '/' + image)
            temp_img = cv2.resize(temp_img, size)
            images.append(temp_img)
            labels.append(labels_dict[folder])

    images = np.array(images)
    images = images.astype('float32')/255.0

    labels = keras.utils.to_categorical(labels)

    X_train, X_test, Y_train, Y_test = train_test_split(images, labels, test_size = 0.05)

    print()
    print('Loaded', len(X_train),'images for training','Train data shape =',X_train.shape)
    print('Loaded', len(X_test),'images for testing','Test data shape =',X_test.shape)

    return X_train, X_test, Y_train, Y_test
```

Figure 29: load data

As can see from the above image, images are loaded from the directories and each image is labeled (the directory name is the label for the image). The images are resized to 64*64 which was the size chosen to train the model with. The next step is to convert the image into a numpy array as Neural Networks expect data in the form of floating point numbers and not images. These floating point numbers are the color values of each pixel in the image. These values can range between 0 and 255 and therefore it is important to normalize the data by dividing the data by 255. This ensures that the data will only range between 0 and 1. Besides these two preprocessing steps, no other preprocessing steps are applied. The dataset is split into train (95%) and test (5%) for both training and testing the model.

Models

For this research two models were trained although only one model was completely trained. This was performed to compare the previously mentioned approaches that can be taken to train CNN networks.

```
def create_model():

    model = Sequential()

    model.add(Conv2D(16, kernel_size = [3,3], padding = 'same', activation = 'relu', input_shape = (64,64,3)))
    model.add(Conv2D(32, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(MaxPool2D(pool_size = [3,3]))

    model.add(Conv2D(32, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(Conv2D(64, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(MaxPool2D(pool_size = [3,3]))

    model.add(Conv2D(128, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(Conv2D(256, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(MaxPool2D(pool_size = [3,3]))

    model.add(BatchNormalization())

    model.add(Flatten())
    model.add(Dropout(0.5))
    model.add(Dense(512, activation = 'relu', kernel_regularizer = regularizers.l2(0.001)))
    model.add(Dense(29, activation = 'softmax'))

    model.compile(optimizer = 'adam', loss = keras.losses.categorical_crossentropy, metrics = ["accuracy"])

    print("MODEL CREATED")
    model.summary()

    return model
```

Figure 30: Create model

The code in the screenshot above shows the model that was defined to be trained from scratch. The model consists of three Convolutional blocks (Convolutional layer with Pooling layer) followed by a Batch normalization layer before being flattened. The first two Convolutional blocks consists of a Conv2D layer with 16 and 32 neurons while the last Convolutional block consists of a deeper model. The input shape, as explained earlier is 64*64 and the 3 indicates the color channel (RGB). The Kernel Size for the Conv2D layers was set at 3*3. As ReLU is the most commonly used activation function, this function was used in all layers of the network. The pool size for the Max Pooling layers was also set to 3*3. The architecture of the model was designed by experimenting and drawing conclusions from other researchers.

The Dropout layer's dropout percentage was set to 50% meaning that 50% of the neurons will be turned off to prevent overfitting. The next layer is a Dense Layer which regularization applied (L2 regularization). The number of neurons for this layer was set to 512 (deep layer) and this layer is succeeded by the final dense layer which was used to produce outputs. Therefore, as discussed earlier, the softmax activation function was applied.

Finally after defining the model, the model was compiled by using the Adam optimizer and the Categorical Crossentropy as the loss function. Adam optimizer is the most commonly used optimizer and is also known to perform well for image classification problems. Categorical crossentropy was chosen as the loss function as the dataset that is being used is categorical data.

```
def create_model_t1():
    base_model=VGG16(weights='imagenet',include_top=False, input_shape=(64,64,3)) #imports the mobilenet model and discards the last 1000 neuron layer.
    x=base_model.output
    x=GlobalAveragePooling2D()(x)
    x=Dense(1024,activation='relu')(x) #we add dense layers so that the model can learn more complex functions and classify for better results.
    x=Dense(1024,activation='relu')(x) #dense layer 2
    x=Dense(512,activation='relu')(x) #dense layer 3
    preds=Dense(N_CLASSES,activation='softmax')(x) #final layer with softmax activation
    model=Model(inputs=base_model.input,outputs=preds)

    model.compile(optimizer='Adam',loss='categorical_crossentropy',metrics=['accuracy'])
    model.summary()

    return model
```

Figure 31: create model t1

The code in the image above shows the code used to define the Transfer Learning model. The most popular architecture called the VGG16 architecture was used and the weights were transferred from the imagenet dataset. The top layer was removed. This is because we want to initialize a new input layer with custom input size. Additional layers were added to the model on

top of the existing layers, mainly Dense Layers. These dense layers ensure that the model is very deep and complex. Configurations similar to the previous model was applied.

For both models, the metric used to measure the performance of the model is accuracy.

```

'''
Train the model using the Training Data.
10% of the data is used to the Validation.
'''

def fit_model():
    model_hist = model.fit(X_train, Y_train, batch_size = 64, epochs = 5, validation_split = 0.1)
    return model_hist

```

Figure 32: fit model

The code in the above image shows how the model was trained. The fit method calls the model training method. The input data needs to be specified when calling this method. Also the batch size, epoch and validation split are important but optional parameters. If not specified, the default values will be used. The batch size indicates the number of images that will be passed to the model in a single batch. The epochs is the number of times the dataset will pass through the model before the training is completed. The validation split percentage/fraction indicates what percentage of the dataset will be used for the validation of the model as soon as an epoch is completed. The validation split is in addition to the test set. The test set will be used to evaluate the model at the end after carrying out model training.

The Transfer Learning model was tried out but as the transfer learning model (VGG16) is very complex (16 layers), it seemed to take too long to complete even 1 epoch (Approximately 10 hours).

Therefore, the model built from scratch was trained and completed. The model was trained for 5 epochs.

```

# Fit the Model on the Training Data and use 10% of it for Validation
curr_model_hist = fit_model()

Epoch 1/5
1163/1163 [=====] - 763s 656ms/step - loss: 0.8538 - accuracy: 0.8228 - val_loss: 0.3535 - val_accuracy: 0.9333
Epoch 2/5
1163/1163 [=====] - 743s 639ms/step - loss: 0.1838 - accuracy: 0.9788 - val_loss: 0.1897 - val_accuracy: 0.9705
Epoch 3/5
1163/1163 [=====] - 719s 619ms/step - loss: 0.1637 - accuracy: 0.9819 - val_loss: 0.1671 - val_accuracy: 0.9782
Epoch 4/5
1163/1163 [=====] - 720s 619ms/step - loss: 0.1369 - accuracy: 0.9878 - val_loss: 0.0830 - val_accuracy: 0.9975
Epoch 5/5
1163/1163 [=====] - 722s 620ms/step - loss: 0.1458 - accuracy: 0.9863 - val_loss: 0.1407 - val_accuracy: 0.9868

```

Figure 33: Epoch

The model was trained for 5 epochs and the training history was plot into a graph towards the end. The plot can be seen below. The plot contains the accuracy and the loss and how it changed during the training. The plot shows both, loss and accuracy, for train and test.

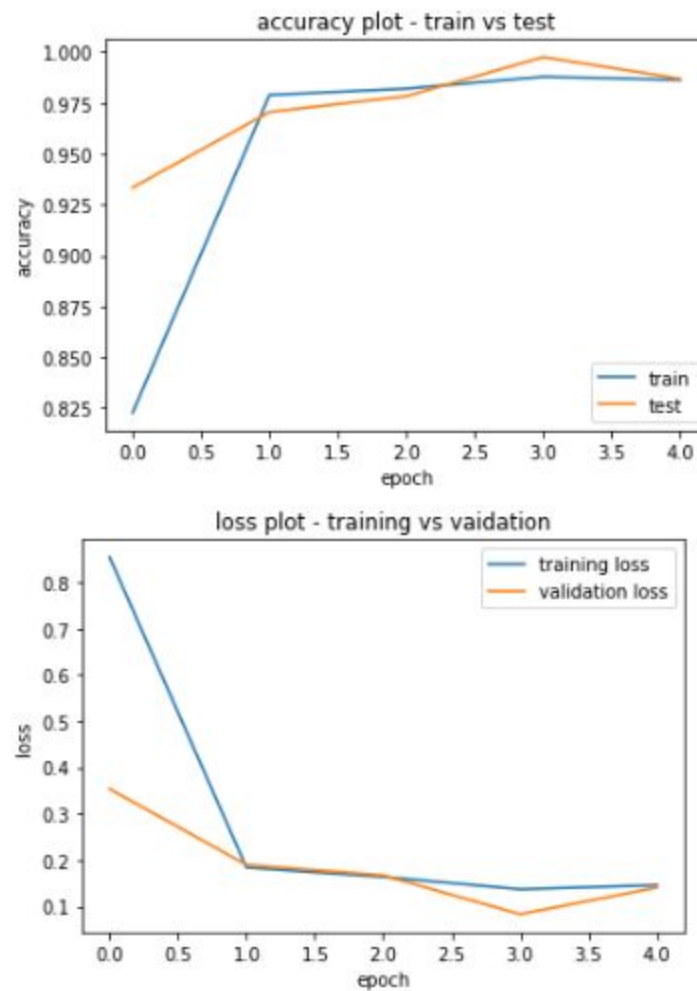


Figure 34: Accuracy and loss plot

Flow of the system.

Proposed solution has 6 major parts for deaf students which are user registration, lessons for deaf/dumb kids, self evaluation part for deaf/dumb kids, evaluation records and send messages to teacher. Teacher can able see messages from the deaf/dumb students in proposed system.

Login and Register for students.

Before using the system, registration is more important to use the proposed solution. Providing username and password helps to enter the system.

Lessons for deaf/dumb kids.

This User Interface contains sign gestures with corresponding english letters. This is very useful for deaf/dumb kids to learn sign language.

Self Evaluation for deaf/dumb kids.

This user interface has buttons with english letters and once the user press button camera user interface will appear. Students need to show the correct gesture and students need to place their hand gesture inside a bounded box. Once they capture the image prediction result will appear as “Well done correct answer” if the letter is correct.

Maintaining Evaluation records.

This user interface is useful to view students’ scores once they attempt the tests and it is useful to see progress.

Sending messages to teachers.

This user interface has a camera to capture sign gusters and convert them into english letters. After conversion, converted texts are sent to the teacher.

Login and register as a teacher.

Teachers need to register the system before they start to use the system.

View messages from students.

Teacher can able to see messages which are from students

Delete messages from students.

Teachers are able to delete messages once they read.

ER Diagram

Student has attributes such as id, username, password, name, email, year, created_on and updated_on. Teacher has attributes such as id, username, password, name, email, role, created_on and updated_on. Message has attributes such as id, student(id), message and message delivered date. Test has attributes such as id, exam, student(id), correct predictions, incorrect predictions and time.

Many students can able to many tests. One student is able to send many messages to teacher. Many messages are received by many teachers.

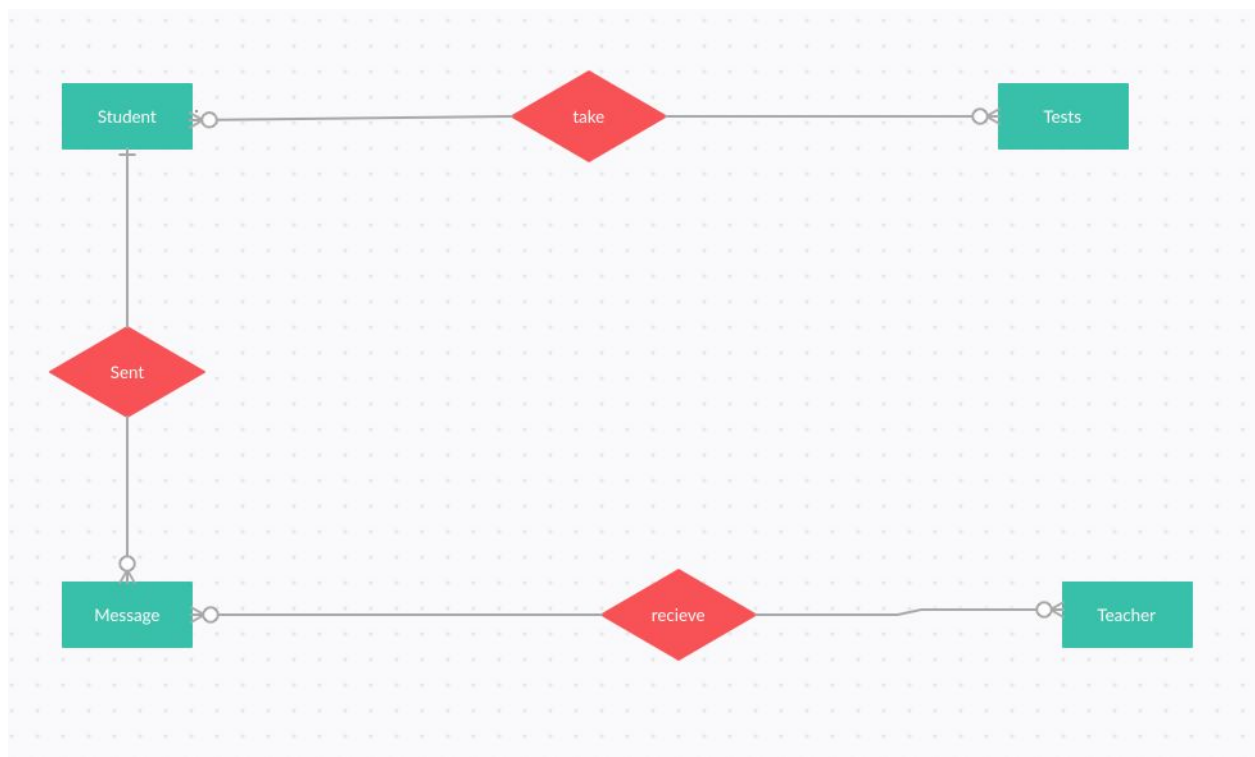


Figure 35: ER Diagram

Development

Introduction

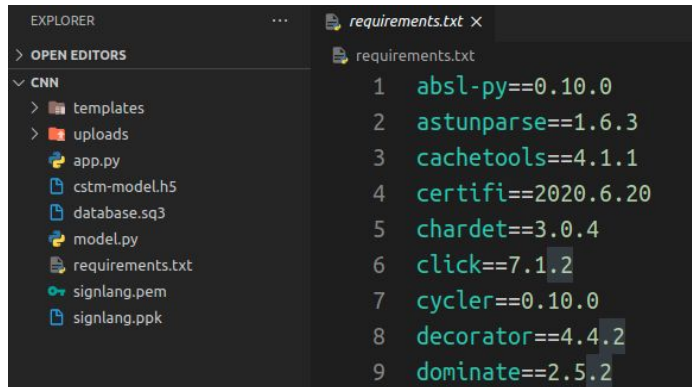
After the design process has to develop a solution according to that so have to execute everything with technology. ER Diagrams and Use case diagrams are helpful to develop solution

Codes

Proposed solution is created with Flask framework for backend and React framework for frontend. Flask is a python micro framework which is really useful to use with machine learning models because machine learning models are created with python so it's easier to integrate with Flask REST application programming interface. React framework is useful to develop user interfaces for proposed solution. SQLite is a database used for this solution to store data into a database. Proposed solution used the python 3.8.3 version to develop the entire back-end part.

Setting up the environment.

Before developing an application, setting up the environment for coding is the most important process. For the development of the proposed solution, we need to install required python libraries for development so to successfully complete this process every library name are included in requirements.txt file. Using pip is able to install libraries which are in requirement.txt. To install all libraries "pip install -r requirements.txt" command needs to be used.



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree for a project named 'CNN'. The tree includes folders 'templates' and 'uploads', and files 'app.py', 'cstm-model.h5', 'database.sq3', 'model.py', 'requirements.txt', 'signlang.pem', and 'signlang.ppk'. The 'requirements.txt' file is selected. On the right, the editor window shows the contents of 'requirements.txt', which is a list of Python dependencies with their versions, numbered 1 through 9.

```
1 absl-py==0.10.0
2 astunparse==1.6.3
3 cachetools==4.1.1
4 certifi==2020.6.20
5 chardet==3.0.4
6 click==7.1.2
7 cycler==0.10.0
8 decorator==4.4.2
9 dominate==2.5.2
```

Figure 36: requirements.txt

Importing libraries

During the development process various libraries needed to be used so importing libraries are most important before starting to use libraries.

```
app.py
1  from flask import Flask, flash, render_template, request, jsonify
2  from flask_bootstrap import Bootstrap
3  from flask_cors import CORS, cross_origin
4  from werkzeug.utils import secure_filename
5  import yaml
6  import os
7  import sys
8  import sqlite3 as sql
9  from flask import jsonify
10 ## Import Libraries related to Image Classification
11 import keras
12 keras.backend.clear_session()
13 from keras.models import load_model
14 from skimage import io
15 from skimage import transform
16 import numpy as np
17 import cv2
18 from keras.preprocessing.image import ImageDataGenerator
19 from keras.preprocessing import image
20 import json
21 import uuid
22 import base64
```

Figure 37: Importing libraries

Initializing flask app

In this code “app = Flask(__name__)” helps to initialize the flask application. CORS is used to apply CORS policy. This “ app.config['UPLOAD_FOLDER'] = 'uploads' ” code helps to set

uploads folders where images are needed to store. This “ app.secret_key = 'LPje8E8Jkx' ” helps to manage cookies.

```
app = Flask(__name__)
CORS(app, support_credentials=True)
app.config['UPLOAD_FOLDER'] = 'uploads'
app.config['CORS_HEADERS'] = 'Content-Type'
app.secret_key = 'LPje8E8Jkx'
```

Figure 38: Question 2

Database Connection

Establishing database connection is most important before handling a database, so the following “db_connect()” function is used to establish connection with SQLite.

```
def db_connect():
    conn = sql.connect('database.sqlite3')
    return conn

def db_close(conn):
    conn.close()
```

Figure 39: Database Connection

Converting to json

This function is created to return values in a JSON format so when some data needed to convert data into JSON format at that time calling this function would be helpful.

```
def dict_factory(cursor, row):  
    d = {}  
    for idx, col in enumerate(cursor.description):  
        d[col[0]] = row[idx]  
    return d
```

Figure 40: Question 2

Predict signs

This function is created to make predictions and it gets the actual value from letter buttons after it compares predicted value and actual value(letter valued pass through button). If the actual value is equal to the predicted value then it will return “match” : “True” otherwise it will return “match” : “False”.

```
@app.route('/predict', methods=['POST'])
@cross_origin(origin='*')
def predict():
    if request.method == 'POST':
        filename=""
        predictions=[]
        print("IN HERE")

        try:
            # check if the post request has the file part
            print("INSIDE TRY")
            if 'file' not in request.form:
                return jsonify({"failed":True})
            print("READING FILE")
            # file = request.files['file']
            # Encode Base64 Image

            b = request.form['file']
            z = b.split(',')
            print(len(z))
            #print(z)
            filename = "image_"+str(uuid.uuid4())+".jpg"
            #im = Image.open(io.BytesIO(base64.b64decode(z))).save(filename)
            with open(os.path.join(app.config['UPLOAD_FOLDER'], filename),"wb") as fh:
                fh.write(base64.b64decode(z[1]))

            actual = request.form['actual']
```

Figure 41: Predict

```

print("Checking File")
if filename:
    print("VALID. SAVING")
    # filename = secure_filename(file.filename)
    # file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
    ## Load Saved Image and Load Model
    print("LOADING FILE")
    img = cv2.imread(os.path.join(app.config['UPLOAD_FOLDER'], filename))
    # img = img[173:(173+316), 679:(679+316)]
    img = img[190:(190+323), 50:(50+323)]
    cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], "sample.jpg"), img)

    print("RESIZE FILE")
    img = cv2.resize(img, (64,64))
    cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], "sample_resized.jpg"), img)
    imgs=[]
    imgs.append(img)
    print("NUMPY FILE")
    imgs=np.asarray(imgs)
    imgs.astype('float32')/255
    """ Start Loading Model """
    print("LOAD MODEL")
    model = load_model('cstm-model.h5')
    """ Post Loading Model - Obtain predictions """
    predictions = model.predict(imgs)

```

Figure 42: Predict Cont 1

```
predictions = model.predict(imgs)

print(predictions)
""" Get Predicted Label """
labels=['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z']
predicted_class = labels[np.argmax(predictions)]

print("Predicted Class: "+predicted_class)
print("Actual Class: "+actual)
if predicted_class == actual:
    return jsonify({"match":True})
else:
    return jsonify({"match":False})

except Exception as e:
    print("Failed to Obtain predictions", sys.exc_info()[0])
    print(e)

return jsonify({"failed":True})
```

Figure 43: Predict cont 2

Letter Predict for message.

This “letterpredict” function is created to detect sign gestures and output of the results used as a input for sending message.

```
@app.route('/letterpredict', methods=['POST'])
@cross_origin(origin='*')
def letterpredict():
    if request.method == 'POST':
        filename=""
        predictions=[]
        print("IN HERE")

        try:
            # check if the post request has the file part
            print("INSIDE TRY")
            if 'file' not in request.form:
                return jsonify({"failed":True})
            print("READING FILE")
            # Encode Base64 Image

            b = request.form['file']
            z = b.split(',')
            print(len(z))
            #print(z)
            filename = "image_"+str(uuid.uuid4())+".jpg"
            #im = Image.open(io.BytesIO(base64.b64decode(z))).save(filename)
            with open(os.path.join(app.config['UPLOAD_FOLDER'], filename),"wb") as fh:
                fh.write(base64.b64decode(z[1]))
```

Figure 44: letterpredict

```

# if user does not select file, browser also
# submit a empty part without filename
print("Checking File")
if filename:
    print("VALID. SAVING")
    # filename = secure_filename(file.filename)
    # file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
    ## Load Saved Image and Load Model
    print("LOADING FILE")
    img = cv2.imread(os.path.join(app.config['UPLOAD_FOLDER'], filename))
    # img = img[173:(173+316), 679:(679+316)]
    img = img[190:(190+323), 50:(50+323)]
    cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], "sample.jpg"), img)

    print("RESIZE FILE")
    img = cv2.resize(img, (64,64))
    cv2.imwrite(os.path.join(app.config['UPLOAD_FOLDER'], "sample_resized.jpg"), img)
    imgs=[]
    imgs.append(img)
    print("NUMPY FILE")
    imgs=np.asarray(imgs)
    imgs.astype('float32')/255
    """ Start Loading Model """
    print("LOAD MODEL")
    model = load_model('cstm-model.h5')
    """ Post Loading Model - Obtain predictions """
    predictions = model.predict(imgs)

```

Figure 45: letterpredict cont 1

```
print(predictions)
""" Get Predicted Label """
labels=['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X']
predicted_class = labels[np.argmax(predictions)]

print("Predicted Class: "+predicted_class)

return jsonify({"letter":predicted_class})

except Exception as e:
    print("Failed to Obtain predictions", sys.exc_info()[0])
    print(e)

return jsonify({"failed":True})
```

Figure 46: letter predict 2

Message

This function is created for sending messages, view messages and delete messages. POST method is useful to send messages to database, GET method used to see all messages and DELETE method used to delete messages.

POST

```
@app.route('/messenger',methods=['GET','POST','PUT','DELETE'])
@cross_origin(origin=('*'))
def messenger():
    if request.method == 'POST':
        # Connect to SQLite Database
        conn = db_connect()

        try:
            return_val=False
            print("READ JSON")
            ## Read Data in POST Request - JSON
            json_data = request.json
            print(json_data)
            studentid=json_data["studentid"]
            message=json_data["message"]

            with conn:
                cur = conn.cursor()
                cur.execute("INSERT INTO message (student, message) VALUES(?, ?)", (studentid, message))
                conn.commit()
                return_val=True
        except sql.Error as error:
            print("Failed to insert data into sqlite table", error)
        finally:
            if (conn):
                # Close the Database Connection
                db_close(conn)
                print("The sqlite connection is closed")

    return jsonify({"return":return_val})
```

Figure 47: messenger POST

GET


```

elif request.method == 'GET':
    # Connect to SQLite Database
    conn = db_connect()

    try:
        # Retrieve all messages
        rows=[]

        with conn:
            conn.row_factory = dict_factory
            cur = conn.cursor()
            cur.execute("SELECT * FROM message")
            rows = cur.fetchall()

    except sql.Error as error:
        print("Failed to retrieve record from sqlite table", error)
    finally:
        if (conn):
            # Close the Database connection
            db_close(conn)
            print("The sqlite connection is closed")

    return jsonify(rows)

```

Figure 48: GET message

DELETE


```

elif request.method == 'DELETE':
    # Connect to the SQLite Database
    conn = db_connect()

    try:
        return_val = False
        ## Read ID and Delete Message
        json_data = request.json
        id = json_data['id']

        with conn:
            cur = conn.cursor()
            cur.execute("DELETE FROM message WHERE id=?", (id))

            if (cur.rowcount < 1):
                return_val = False
            else:
                return_val = True
    except sql.Error as error:
        print("Failed to delete record in sqlite table", error)
    finally:
        if (conn):
            # Close the Database Connection
            db_close(conn)
            print("The sqlite connection is closed")

    return jsonify({"return":return_val})

```

Figure 49: DELETE Message

This function is created to register new teacher, get teacher details and update teacher details.

```

202 @app.route('/user', methods=['GET','POST', 'PUT', 'DELETE'])
203 @cross_origin(origin=('*'))
204 def users():
205     if request.method == 'POST':
206         # Connect to the SQLite Database
207         conn = db_connect()
208
209         try:
210             return_val=False
211             ## Read Data in POST Request - JSON
212             json_data = request.json
213             username=json_data['username']
214             password=json_data['password']
215             name=json_data['name']
216             email=json_data['email']
217             role=json_data['role']
218
219             with conn:
220                 cur = conn.cursor()
221                 cur.execute("INSERT into user (username,password,name,email,role) V
222
223                 conn.commit()
224
225                 return_val=True
226         except sql.Error as error:
227             print("Failed to insert record into sqlite table", error)
228         finally:
229             if (conn):
230                 # Close the Database Connection
231                 db_close(conn)
232                 print("The sqlite connection is closed")

```

Figure 50: teachers POST

Teacher Login

This login is created for teacher login

```
@app.route('/login', methods=['POST'])
@cross_origin(origin='*')
def login():
    if request.method == 'POST':
        # Connect to SQLite Database
        conn = db_connect()

        try:
            return_val=False
            ## Read Data in POST Request - JSON
            json_data = request.json
            username=json_data["username"]
            password=json_data["password"]
            row=[]

            with conn:
                conn.row_factory = dict_factory
                cur = conn.cursor()
                cur.execute("SELECT * FROM user WHERE username=? AND password=?", (username,password))
                row=cur.fetchone()

        except sql.Error as error:
            print("Failed to get data from sqlite table", error)
        finally:
            if (conn):
                # Close the Database Connection
                db_close(conn)
                print("The sqlite connection is closed")

    return jsonify(row)
```

Figure 51: login POST

Student Login

```
@app.route('/student/login', methods=['POST'])
@cross_origin(origin='*')
def student_login():
    if request.method == 'POST':
        # Connect to SQLite Database
        conn = db_connect()

        try:
            return_val=False
            ## Read Data in POST Request - JSON
            json_data = request.json
            username=json_data["username"]
            password=json_data["password"]
            row=[]

            with conn:
                conn.row_factory = dict_factory
                cur = conn.cursor()
                cur.execute("SELECT * FROM student WHERE username=? AND password=?", (username,password))
                row=cur.fetchone()

        except sql.Error as error:
            print("Failed to get data from sqlite table", error)
        finally:
            if (conn):
                # Close the Database Connection
                db_close(conn)
                print("The sqlite connection is closed")
```

Figure 52: student login

Student Exam

```
@app.route('/student/exam', methods=['GET', 'POST', 'PUT', 'DELETE'])
@cross_origin(origin='*')
def student_exam():
    if request.method == 'POST':
        # Connect to SQLite Database
        conn = db_connect()

        try:
            return_val=False
            ## Read Data in POST Request - JSON
            json_data = request.json
            attempt=json_data["attemp"] # Attempt
            student=json_data["studentid"]
            correct=json_data["correct"]
            incorrect=json_data["incorrect"]

            with conn:
                cur=conn.cursor()
                cur.execute("INSERT INTO student_exam (exam, student, correct, incorrect) VALUES(?,?,?,?)", (attempt, student, correct, incorrect))

                conn.commit()

                return_val = True
        except sql.Error as error:
            print("Failed to insert record into sqlite table", error)
        finally:
            if (conn):
                # Close the Database Connection
                db_close(conn)
                print("The sqlite connection is closed")
```

Figure 53: student exam

Students

```

@app.route('/student', methods=['GET', 'POST', 'PUT', 'DELETE'])
@cross_origin(origin= '*')
def students():
    if request.method == 'POST':
        return_val = False
        # Connect to SQLite Database
        conn = db_connect()
        try:
            ## Read Data in POST Request - JSON
            json_data = request.json
            print(json_data)
            username=json_data["username"]
            password=json_data["password"]
            name=json_data["name"]
            email=json_data["email"]
            year=json_data["year"]

            with conn:
                cur = conn.cursor()
                cur.execute("INSERT INTO student (username,password,name,email,year) VALUES(?,?,?,?,?)", (username,password,

                conn.commit()

            return_val = True
        except sql.Error as error:
            print("Failed to insert record into sqlite table", error)
        finally:
            if (conn):
                # Close the Database Connection
                db_close(conn)

```

Figure 54: students

Evaluation

Evaluation is an important part which really helps to deliver solutions with expected quality. This solution developed with agile methodology so after each major development everything was tested. Results from the testing are able to develop this more effectively and can provide the system with user expected quality.

Data type Check tests

This system has user interfaces for student login, student registration, student details update teacher login and teacher registration so during testing it is able to check the data type of each input and which helps to make sure the correct type of the data.

API tests

API tests are most important to make sure can able to get existing data ,update existing data ,create new data and delete existing data. Insomnia tool is useful to test APIs and through insomnia can be able to see header details such as Content-Type, Server and Access-Control-Allow-Origin. In development needed to face issues with Content-Type so to overcome this problem every Content-Type change as application/json. In insomnia to test url have to mention url with methods such POST, GET, PUT and update. Values needed to pass in a JSON format except prediction because prediction is using multipart form to get the user input which is the image and actual value.

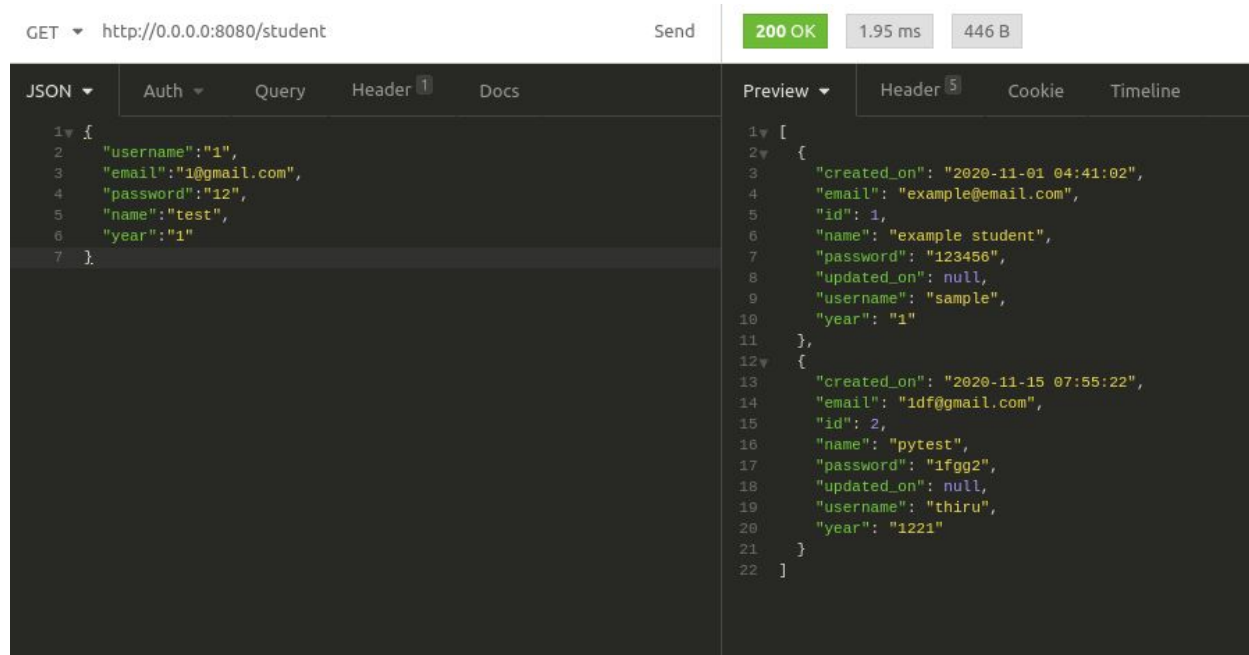


Figure 55: Insomnia

Once data is inserted with the POST method, need to check the database if the data is successfully inserted or not.

user Enter a SQL expression to filter results (use Ctrl+Space)

	id	username	password	name	email
1	1	admin	12345	administrator	admin@email.com
2	2	se	11	stepahn	rt5909f@greenwich.ac.uk

Figure 56: dbeaver

Developed Machine learning models tested.

Sign gesture detection is the most important part in this research so there are 2 machine learning models developed to detect english sign gestures. 2 models are compared with prediction time and accuracy. Implemented model performed well in real world scenario with less prediction time and accuracy also better than previous model(which is using transferred learning technique)

Final testing of project by users and clients

Testing the proposed solution with the user is the most important process to identify weakness in the proposed system. Testing process is happened while user training process so user training so from the first day of user training can able identify which area have to place bounded box because bounded box is import to place the hand for prediction. In user training, check letters can be visible for every kid so according to that made some changes to all kids. Designing the camera user interface button for capture sign gesture is the most important change needed to change after the user training because earlier doesn't have button for capture and during the training students feel difficult use capturing images with a timer (Earlier user interface has capture the image after 3 seconds once they press "ready" button. Students need to send messages so students have to use a spacebar and backspace on the keyboard because in the training process they have to train them to use both buttons on the keyboard.

Testing process completely after the compilation of the system.

1. Login and Register have validation check before send user data.(Working Fine)
2. After the Login process relocated to Students Details.(Working Fine)
3. Can able to edit and save student details on student details user interface.(Working Fine)
4. Once click the lesson button on the panel relocated to the lesson user interface.(Working Fine)
5. Once Clicked exam button on panel relocated to user interface with english letters.(Working Fine)
6. Once the letter clicked the selected letter button on the exam user interface relocated the camera user interface.(Working Fine)

7. Once captured image predicted result is loading and relocated to letter user interface.(Working Fine)
8. Once clicked message relocated to message user interface.(Working Fine)
9. Once capture the sign gesture with button predicted letter appeared in input box.(Working Fine)
10. Send predicted sentences to the teacher.(Working fine)
11. Teacher can able to see all messages from students.(Working Fine)

Your review of the product.

Proposed system is effective solution for kindergarden kids who are dumb/deaf and most of the time students can able to try this system while user training. Kids are really enjoying it while they try new sign language gestures and this system also has drawbacks because this system predicts english letters when the background of the captured image with light colors. If the background is darker then the model is difficult to identify some letters. Following letters are predicted well in this model A,B,D,E,F,G,I,J,K,L,N,O,U,Y,W,M and the following letters can be predicted by various background color differences Q,D,S,H,V and P. In this model R,T,S,Z,X and C can't be predicted well many times so this is a major drawback of the system.

Future Development.

This proposed system has issues with predicting some letters so the future has to fix those problems and needs to deliver the system without any errors. Datasets used to train models in this research with light background so it can't detect gestures properly with dark background which means having to select a dataset with light background and dark background. English Sign words are with movements so the cnn model can't develop the model in a more accurate way so in future can implement features like detecting words using RNN. RNN is helpful to develop models with more accuracy.

Conclusion

The purpose of the creation of the proposed solution is providing self learning platform for deaf/dumb kindergarden kids. This solution helps to avoid issues which are mentioned under the requirement analysis section. After the implementation of the proposed system is very useful for deaf/dum kindergarten kids who need to learn sign language. Most of the kindergarten kids have their own way of thinking and they like to learn everything in a practical way and many of them are getting bored while learning lessons again and again but through this solution they can attempt to try out things once they learn. Many kids can't memorize all words but once they try more and more they can enjoy learning with this proposed solution.

Teachers aren't able to pay attention to all the students because they don't have more time for each student so students need additional time to revise once they learn lessons. Deaf/ Dumb students while learning sign gesture teacher can able to check every student can able to show correct sign gesture so it's a problem. Once student login to system then they don't need to help from someone need to check he/she is showing correct gesture or not because system display answer once they try newsletter.

There are many solutions are currently available for learning sign gestures but those options doesn't have evolution part for deaf/dumb kids those are mentioned in review of other products so this system can be used as useful evaluation system once they learnt sign gestures with those applications.

References

1. Centers for Disease Control and Prevention (CDC). [Identifying infants with hearing loss - United States, 1999-2007](#). MMWR Morb Mortal Wkly Rep. 59(8): 220-223.
Vohr B. [Overview: infants and children with hearing loss—part I](#). Ment Retard Dev Disabil Res Rev. 2003;9:62–64.
2. NIDCD. (2019). *American Sign Language*. [online] Available at: <https://www.nidcd.nih.gov/health/american-sign-language>.
3. Ahmed, M.A., Zaidan, B.B., Zaidan, A.A., Salih, M.M. and Lakulu, M.M. bin (2018). A Review on Systems-Based Sensory Gloves for Sign Language Recognition State of the Art between 2007 and 2017. *Sensors*, [online] 18(7), p.2208. Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6069389/>.
4. Saldaña González, G., Cerezo Sánchez, J., Bustillo Díaz, M.M., Ata Pérez, A., Saldaña González, G., Cerezo Sánchez, J., Bustillo Díaz, M.M. and Ata Pérez, A. (2018). Recognition and Classification of Sign Language for Spanish. *Computación y Sistemas*, [online] 22(1), pp.271–277. Available at: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-55462018000100271 [Accessed 21 Oct. 2020].
5. Fullscale.io. 2020. [online] Available at: <https://fullscale.io/blog/machine-learning-computer-vision/> [Accessed 30 November 2020].
6. Contributor, R., 2020. *Accelerometers: What They Are & How They Work*. [online] livescience.com. Available at: <https://www.livescience.com/40102-accelerometers.html> [Accessed 30 November 2020].
7. Jeyasheel, G., 2020. *Shieldsquare Captcha*. [online] Iopscience.iop.org. Available at: <https://iopscience.iop.org/article/10.1088/1742-6596/1362/1/012034> [Accessed 30 November 2020].

8.2020. GLOVE BASED SIGN LANGUAGE INTERPRETER FOR DEAF AND APHONIC PEOPLES. [online] Bangalore, Karnataka, India. Available at: <<https://www.irjet.net/archives/V7/i6/IRJET-V7I670.pdf>> [Accessed 30 November 2020].

9.Kumar Mummadi, C., Peter Leo, F., Verma, K., Kasireddy, S. and Scholl, P., n.d. *Real-Time Embedded Recognition Of Sign Language Alphabet Fingerspelling In An IMU-Based Glove*. University of Freiburg.

10.Bao, P., Maqueda, A., del-Bianco, C. and García, N., n.d. *Tiny Hand Gesture Recognition Without Localization Via A Deep Convolutional Network*.

11.Trigueiros, P., Ribeiro, F. and Reis, L., n.d. *Vision-Based Portuguese Sign Language Recognition System*.

12.Liu, P., Li, X., Cui, H., Li, S. and Yuan, Y., 2020. *Hand Gesture Recognition Based On Single-Shot Multibox Detector Deep Learning*.

13.Brashear, H., Starner, T., Lukowicz, P. and Junker, H., 2020. *Using Multiple Sensors For Mobile Sign Language Recognition*.

Appendix A

Undergraduate Final Year Project Proposal

[Developing a Sign-Based Educational system for Deaf/Mute kindergarten children learn to read by themselves]

[Rishiharan Thirukumaran]

[BSc(Hons) in Computing]

[001090097]

11 Overview

Deaf/ Muted kids have so many issues to learn things by themselves because they always need some trained special person to teach them. There are so many difficulties there to handle deaf/mute kids like presenting lessons in an interesting way, difficult to monitor them with more kids which means it's difficult to pay attention to specific kids, some kids always need parents with them, some kids have shyness to ask questions with other kids. After analysing the real world problems

We can able to make a solution based on their needs which is more concerned about deaf/dumb kids. The solution motivate to learn for deaf/dumb kids like able to read english words and letters. This solution can be operated by parents eventually they can able to teach how to operate this for their kids but deaf/dumb kids can able to learn the things by them after they enter the system. From this solution students are able to try everything by themselves and are able to try many times. Students are able to ask the questions with sign language with messages. Questions or tests can be treated like a game so students always try to reach maximum benefit. All the records are saved into this system so parents are able to see the progress.

Methodology

step 1 - get the sample data

step 2 - image preprocessing

step 3- feature extraction

step 4- classification and recognition for the model.

step 5- get the output.

step 6- integrate the system with web api.

step 7- Checking the accuracy

step 8- Improve accuracy based on the tests

step 9- Create an educational system and integrate the api which contains the model.

step 10- Get the input and as a video input and display the output according to input.

step 11- Implement feature like user registration, profile management with latest records

step 12- Implement messaging service with sign to english interpretation.

21 Aim

Developing system to self learn processes for deaf/mute pre-school children. This process motivates students to be able to read english words by themself.

31 Objectives

1. All the users need to register before they are going to use the system.
2. Creating basic lessons for kindergarten kids with learning elements.
3. Create a sign to english interpreter model.
4. Integrate the games with sign to english interpretation system.
5. Maintain students' records with high scores.
6. creating messassaging services which translates questions sign to english asked by children.

41 Legal, Social, Ethical and Professional

Legal

Each country has an act for protecting customers data and their personal information so we have to secure users' information because we can't expose it to the public. For example if the user has

some disability or disease we have kept everything as confidential because if we missed to protect the data then users are able to take legal action against the product.

Social

Currently there are many school and institutions available for deaf/dumb kids. Every school special programmes for their students and they have highly skilled professionals or experts in teaching and handling kids. If the proposed system gets more popular then a huge amount of people start using online based education instead of attending school so eventually schools might lose some profit and they might feel difficult to pay for their employees too.

Ethical

The students and their learning ability differs from every student so we have to design the system according to that. We have to get issues or problems related to the learning process and can able to make changes according to the feedback. Feedbacks can be sent as a message. Message functionality has sign to text interpretation so students can easily send their feedback easily.

Professional

The proposed must be easy to use so it should have a simple and clear user interface. The system must have a username and password to access the system and customers need to set a valid password which has more strength and is unpredictable.

51 Planning (see appendix A)

Month	June				July					August				September				October					November				December			
	2020				2020					2020				2020				2020					2020				2020			
	1	2	3	4	1	2	3	4	5	1	2	3	4	1	2	3	4	1	2	3	4	5	1	2	3	4	1	2	3	4
Proposal Discussion																														
Project Proposal																														
Literature Survey & Research																														
Proposed plan/ method																														
System Implementation /Testing																														
Final documentation																														
Final project Presentation																														

61 Initial References

1. <http://www.limix.it/en/>
2. https://play.google.com/store/apps/details?id=br.com.handtalk&hl=en_US&gl=US
3. https://play.google.com/store/apps/details?id=com.teachersparadise.aslamericansignlanguage&hl=en_US&gl=US

Appendix B

Survey form which provided to kids

1. How much time do you like to allocate for learning continuously ?
 - 20 minutes to 30 minutes
 - 40 minutes to 60 minutes
 - 70 minutes to 90 minutes
2. Do you like to use computers?
 - Yes
 - No
3. Do you like to use web applications for learning?
 - Yes
 - No
 - I don't know anything about web applications.

4. Do you have any previous experience in using web applications?
 - Yes
 - No
5. What kind of colors do you like to see in a learning application?
 - Only dark colors
 - Only light colors
 - Combination of both
6. Are you able to type letters on a keyboard alone?
 - Yes
 - No
7. Do you need any help from others while using a web application?
 - I can handle alone
 - No, I need my parents
8. Do you like to see your progress or achievement regularly?
 - Yes
 - No