

Software Requirement Specification Document for Project LipDrive

Ziad Thabet, Amr Nabih, Karim Azmi, Youssef Samy
Supervised by: Dr Mai El-Shehaly and Eng Silvia Soliman

June 6, 2018

1 Introduction

1.1 Purpose of this document

The purpose of this document is to present a detailed description of the LipDrive system. LipDrive system is based on lip-reading, in which lip's movements are decoded into understandable words. This document will explain the purpose and features of the system, and describe its interfaces and it will also explain what the system will do, the constraints under which it must operate, and how the system will react to user-generated and environment-incurred stimuli. This software requirements specification (SRS) document is, therefore, intended for the stakeholders and developers of the LipDrive system; and will be proposed to Microsoft Egypt. This document is also presented in partial fulfillment of a graduation project at Misr International University (MIU).

1.2 Scope of this project

The LipDrive system aims to develop a deep learning approach for real-time detection of spoken words in an autonomous vehicle setting. Not only does the system work on the word level but also on the sentence level such as "*LipDrive turn left*", "*LipDrive go home*". The system will facilitate the detection of words at variant speeds of speech. This system aims to help a driver in an autonomous vehicle (AV), to deliver certain commands to the vehicle in a noisy environment, like passengers talking or radio is on, through lip-reading. This will increase the accuracy of speech recognition between the driver and the vehicle. These commands could be as: "*Start Ignition, stop the car*", or "*set direction to a certain destination*". Most of the commands will need to be done instantaneously, like "*change lane*" or "*reduce the speed limit*". This project would help AV to receive commands in real-time without turning off music or asking other passengers to stop talking. The system is composed of a camera which is set in the AV, that will detect the driver's lips movement and based on a trained data, the system will be able to detect the spoken command.

1.3 Overview of this Document

Lip-reading, according to the Cambridge dictionary, is to understand what someone is saying by watching the movements of their mouth. Lip-reading plays a vital role in human communication and speech understanding however, it is a difficult task to be done by humans. The project aims to enhance the accuracy of speech recognition through lip-reading. Using deep learning approach, the system detects the spoken words of a driver in an Autonomous Vehicle. The system aims to facilitate speech recognition in a noisy environment, which facilitates the communication between the vehicle and the driver. Through a mobile application, the driver will be able to deliver commands to the vehicle anytime and in any environmental conditions. For example, the driver wants to get direction for a certain destination. Whether the driver is alone, there are passengers talking, or there the radio is on, the driver will be able to deliver the command easily without changing in any of these conditions.

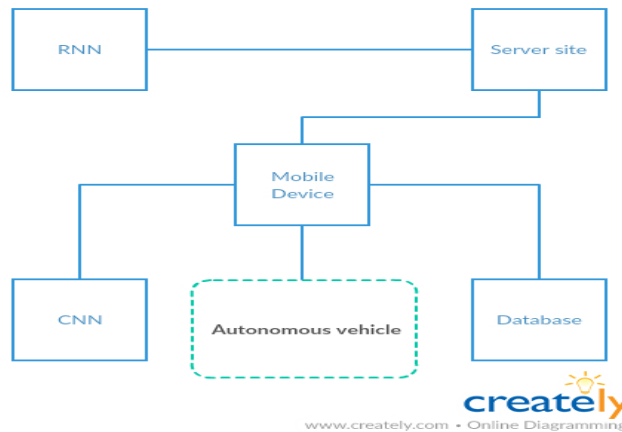


Figure 1: Block Diagram

1.4 Business Context

The Dubai's Roads and Transport Authority (RTA), has carried out the first test run of an autonomous vehicle, it is expected to be set to launch operations by 2020. According to the article entitled "RTA is the first in the world in transport sector services technologies.", RTA's mission and vision are mentioned to be:

1. Vision: Safe and Smooth Transport for All
2. Mission: Develop integrated and sustainable transportation systems and provide distinguished services to all stakeholders to support Dubais comprehensive growth plans through preparing policies and legislations, adapt-

ing technologies and innovative approaches, and implementing world-class practices and standards

The project idea was presented to Dr. Ismail Hisham Zohdy, Chief Specialist/Program Manager of Self-Driving Transport in Roads and Transport Authority (RTA) in Dubai, and has showed his interest to this project, stating that he is going to be part of this project's stakeholder team.

Lip Drive Business Model

Key Partners

Roads and Transport Authority (RTA) in Dubai.

The Dubai's Roads and Transport Authority (RTA), has carried out the first test run of an autonomous vehicle, it is expected to be set to launch operations by 2020.

Banks

Online Banking-Internet Services in which Banks provides to its clients to get various services without having to go to the bank personally.

Key Activities

Developing, maintaining and updating the machine learning model(s)

Collecting and synthesizing training data-sets.

Key Resources

Software Developers

Value Proposition

Speech Recognition Enhancement in Autonomous Vehicles

The main problem reported by the vast majority of IPA users, when communicating to an intelligent personal assistant like Siri or Google, is the vague detection of words due to the surrounding noise. Using lip reading approach, LipDrive aims to enhance the accuracy of speech recognition and the detection of the commands given by a driver.

Two-Factor Authentication

A major source of concern reported by users when typing a password, that there are many people around. Thus, most of the users find that the most secure authentication method, after providing username and password, is to have something that reads your lips.

Customer Relations

Personal assistance

Self-Services

Automated Services

Distribution Channels

Mobile Application

Through mobile application, users can deliver commands to an AV. In addition, it can be used as a Two-Factor Authentication

Customer Segments

Autonomous Vehicles Drivers

Intelligent Personal Assistant Users

Online Banking Clients

Cost Structure

Software Developers Salaries

Revenue Streams

Banks: A subscription for extra security procedures; giving subscribed customers the option of the two factor authentication.

Autonomous Vehicle Drivers: An extra feature that allows drivers to interact with the vehicle; such feature would be an optional add on with a specific price. And it would be available for higher models, which would be an incentive that would push higher models' sales.

2 General Description

2.1 Product Functions

The system aims to enhance speech recognition through lip reading. Thus, the main goal of this project is to develop a deep learning approach for the real-time detection of spoken words in autonomous vehicles setting. The training of a deep-layered neural networks is being used to convert lips movement to written words.

2.2 Similar System Information

The system is a mobile application in which will be able to deliver commands to an autonomous vehicle. The system doesn't intend to be stand alone application but a part of the autonomous vehicle setting. A connection will be set between the application and the vehicle to deliver the commands easily.

2.3 User Characteristics

The system aims to target large number of users who needs a real enhancement in speech recognition. These users are classified into two classes.

1. Data curators: users interested in training the model with a specific set of commands or sentences.
2. Speakers: users who speak to LipDrive and wish to have their lips read by the software.

2.4 User Problem Statement

Users have stated that the main problem, when communicating to an intelligent personal assistant like Siri or Google, is the vague detection of words due to the surrounding noise. On the 10th of November 2018, we have launched a questionnaire to identify users' problem with IPA. After collecting 130 responses, 79 of them use IPA and mostly 58.8 percent have faced problems ordering an IPA due to the surrounding noise. This leads to the dissatisfaction of some users that 74.5 percent of them stop using the application, while 17 percent of them ask people to stop talking/or reduce the noise.

Have you faced problems ordering an IPA due to surrounding noise?

80 responses

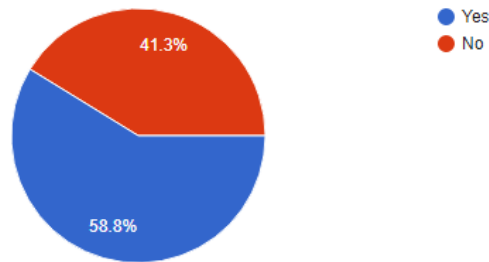


Figure 2: Statistics 1

Section 2: Tell us about your use of technology

If yes, what do you usually do?

47 responses

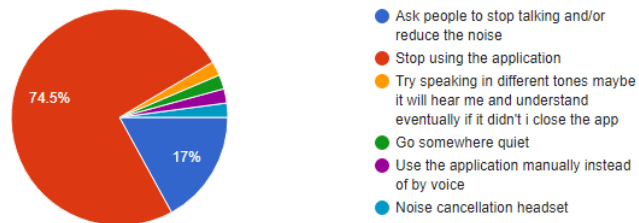


Figure 3: Statistics2 2

2.5 User Objectives

The solution is designed for individual users who are using an intelligent personal assistant, like Siri or google, frequently, in which they are able to give commands to the mobile or the autonomous vehicle. Moreover, the user is able to deliver commands in any noisy environment, as the commands are delivered through lip reading only.

2.6 General Constraints

One of the main constraints of the system is the variant light conditions that the camera could face, the lip reading process is mainly conducted under the ideal lighting conditions. In addition, the position of the speaker to the camera, the camera should be positioned in front of the speaker's face for clear detection. Not only the position of the speaker, but also the distance between the speaker and the camera has to be near enough to detect the lips clearly. Speaker should consider not to be far to deliver the command. Different accents would be challenging to the system to detect the words.

3 Functional Requirements

3.1 read

Table 1: read

Scope	Use case 8.7
Description	this function reads from capture video
Action	this function grabs and retrieve a frame from the captured video
Input	an image empty to retrieve the frame in it
Output	return true or false if the image was taken successfully or not
Precondition	camera is pre set before calling the function
Post-condition	the features is extracted from the frame for other processing
Dependencies	the feature extraction needs a frame to process on and this function provides it
Priority	10/10

3.2 Classification

3.2.0.1 run_epoch

Table 2: run_epoch

Scope	use case 8.13, 8.14 ,8.15
Description	This function start training and testing data
Action	it takes object from model and start training the data
Input	Model which represent the network that we will be working on Session where it encapsulates the environment that tensor-flow is working on
Output	None
Precondition	we have to have object from Model and valid session
Post-condition	the RNN if detect command classify
Dependencies	classifying
Priority	10/10

3.3 build_rnn_graph

Table 3: build_rnn_graph

Scope	Use Case 8.13, 8.14, 8.15
Description	This function build network based on CUDNN structure
Action	it takes object from Input and and boolean whether it is training or not and build graph based on them
Input	Input and boolean
Output	Network
Precondition	we have to call getGraph first
Post-condition	the RNN is created
Dependencies	classifying
Priority	10/10

3.4 login

Table 4: login

Scope	Use Case 8.1 & Non-Functional 6.1
Description	Enables the user to get into the system
Action	Checking if the user and password exists in the database
Input	String name, String password
Output	Boolean true or false
Precondition	The user must be already signed up
Post-condition	None
Dependencies	User won't be able to access the system if the user cant login
Priority	10/10

3.5 Signup

Table 5: Signup

Scope	Use Case 8.4 & Non-Functional 6.1
Description	This function is to create account for user
Action	Takes the information from user and insert it into the database
Input	String firstname,String lastname,String username,String password,String email
Output	boolean
Precondition	check if the user exists
Post-condition	The new account is created
Dependencies	Sign in
Priority	10/10

3.6 realTimeExtraction

Table 6: realTimeExtraction

Scope	use case 8.7 & 8.8
Description	this function extract features from a real time image
Action	takes a camera object and reads a frame out of it and retrieve the features extracted from it
Input	camera object
Output	vector of of points as features
Precondition	camera is intialized with the operating system webcam
Post-condition	features then go through the rnn to classify the feature
Dependencies	the classification of the rnn needs the real time fed images
Priority	10/10

3.7 getFeatures

Table 7: getFeatures

Scope	use case 8.7, 8.8, 8.13, 8.14 & 8.15
Description	this function extract features from a set of videos
Action	takes a videos object and reads a video by video out of it and retrieve the features extracted from it in a sequence labelled with the user word
Input	videos object
Output	sequence of frames of the extracted feature
Precondition	videos paths must be correct
Post-condition	features then go through the rnn to train, validate and test the feature
Dependencies	the training, validation and testing of the rnn needs the real time fed images
Priority	10/10

3.8 interpolation

Table 8: interpolation

Scope	use case 8.13, 8.14, 8.15
Description	interpolate frames of videos to resize it with more frames
Action	takes video resize it to the number of frames required
Input	video and number of frames
Output	modified video
Precondition	number of frames required must not a negative value or zero
Post-condition	none
Dependencies	needed to modify video with small number of frames to stretch it
Priority	10/10

3.9 captureFrame

Table 9: captureFrame

Scope	use case 9.7 & 8.8
Description	capture the frame
Action	returns current frame from camera
Input	none
Output	image
Precondition	camera must be intialized
Post-condition	none
Dependencies	real time feature extraction depends on it
Priority	9/10

3.10 add

Table 10: add

Scope	nonfunctional requirement 6.2
Description	This is a function that is implemented by all models to add data
Action	Adding new object in the database
Input	None
Output	Boolean true or false
Precondition	validate data entered
Post-condition	object is added in the database
Dependencies	None
Priority	9/10

3.11 Encrypte

Table 11: Encrypte

Scope	Non-Functional 6.2
Description	This function is to encrypt our data to keep it secured
Action	Encrypt data using RSA algorithm
Input	String
Output	String
Precondition	data is raw
Post-condition	Data is be encrypted
Dependencies	All inherited classes
Priority	9/10

3.12 Decrypte

Table 12: Decrypte

Scope	Non-Functional 6.2
Description	This function is to decrypte our data to keep it secured
Action	Decrypte data using RSA algorithm
Input	String
Output	String
Precondition	data is encrypted
Post-condition	data return to its normal state
Dependencies	All inherited classes
Priority	9/10

3.13 changeFrames

Table 13: changeFrames

Scope	Use Case 8.1, 8.3, 8.4, 8.6, 8.7, 8.9, 8.12 & Non-Functional 6.2
Description	This function is to switch between two different frames
Action	It changes from frame to another frame
Input	currentFrame
Output	nextFrame
Precondition	None
Post-condition	None
Dependencies	Nav_Bar, Main_Frame, History_frame, Settings_Frame
Priority	8/10

3.14 showPassword

Table 14: showPassword

Scope	use case 8.1, 8.4
Description	This function is to show password
Action	change the stars to a visible word
Input	None
Output	None
Precondition	Hidden
Post-condition	Make it visible
Dependencies	SignUP_Frame, Forget_Password_Frame, SignIN_Frame
Priority	8/10

3.15 delete

Table 15: delete

Scope	nonfunctional requirement 6.2
Description	This is a function that is implemented by all models to delete data
Action	Deleting object in the database
Input	id of the object
Output	Boolean true or false
Precondition	Object must be already in the database
Post-condition	the object is deleted from the database
Dependencies	None
Priority	8/10

3.16 updateSentence

Table 16: updateSentence

Scope	use case 8.7 & 8.8
Description	This function displays the spoken sentence
Action	Change sentence on the screen to what the vectors was classified to
Input	None
Output	None
Precondition	classification must be done
Post-condition	None
Dependencies	None
Priority	8/10

3.17 saveToHistory

Table 17: saveToHistory

Scope	use case 8.8
Description	This function saves the command into the database
Action	calls add function in Command_model
Input	None
Output	None
Precondition	There must be command
Post-condition	add function from command_model
Dependencies	None
Priority	8/10

3.18 modify

Table 18: modify

Scope	nonfunctional requirement 6.2
Description	This is a function that is implemented by all models to modify data
Action	modify model in the database
Input	None
Output	Boolean true or false
Precondition	object must be already in the database
Post-condition	object is modified if true
Dependencies	none
Priority	7/10

3.19 clearHistory

Table 19: clearHistory

Scope	use case 8.9, 8.10
Description	This is function to clear history of the user
Action	Delete all records of user history in the database
Input	None
Output	Boolean true or false
Precondition	User must be logged in
Post-condition	all of user history is deleted
Dependencies	None
Priority	7/10

3.20 logOut

Table 20: logOut

Scope	use case 8.3
Description	This is function to log the user out of the system
Action	Sign out and return to login screen
Input	None
Output	Boolean true or false
Precondition	User must be logged in
Post-condition	redirect to main screen
Dependencies	The user wont be able to use the system until the user logs in again
Priority	7/10

3.21 fillHistory

Table 21: fillHistory

Scope	Use Case 8.9
Description	This function is to show the history of the commands by the user
Action	Fetch the all history data from the database for the user and display it
Input	None
Output	None
Precondition	The user must have history & the application is connected to the server
Post-condition	The History table is filled by an array of history objects
Dependencies	None
Priority	5/10

3.22 showHistoryDetails

Table 22: showHistoryDetails

Scope	use case 8.9
Description	This function displays the history details frame and sends the historyObject to it
Action	It displays the command's information saved into history-Object
Input	currentFrame, nextFrame, object from historyModel
Output	None
Precondition	fill History must be called
Post-condition	history details is showed to the user
Dependencies	It depends on History_Frame
Priority	5/10

3.23 showDataInformation

Table 23: showDataInformation

Scope	Use Case 8.9
Description	This is function is to view the information of a command given by the user time of the command location
Action	Fetch the details of a certain command
Input	ID of selected History
Output	None
Precondition	The history frame must contain histories to be able to show its details
Post-condition	It shows the details of a chosen command
Dependencies	It depends on Functional-Requirement fillHistory & showHistoryDetails
Priority	3/10

4 Interface Requirements

4.1 User Interfaces

4.1.1 GUI

1. Login Screens

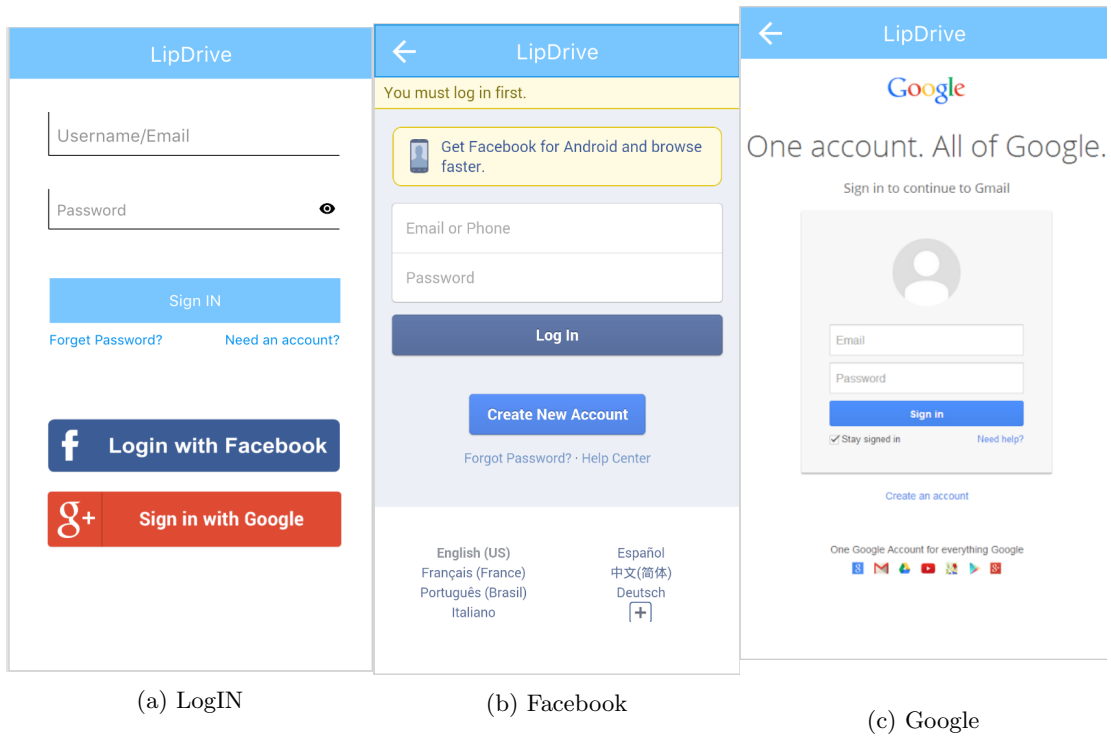
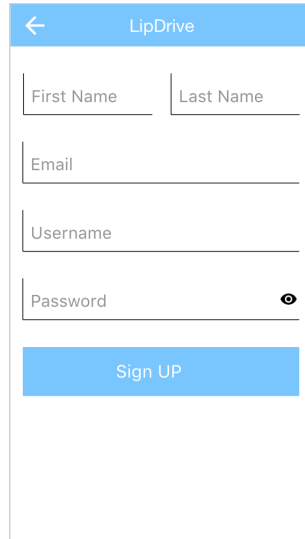


Figure 4: SignIN

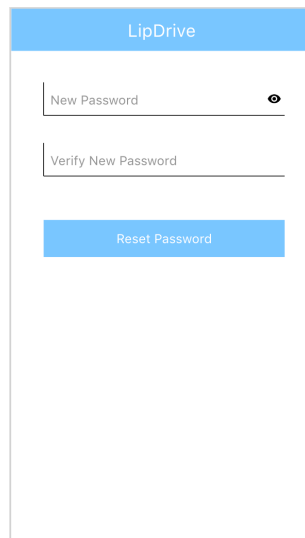
2. SignUP Screen



The screenshot shows the SignUP screen of the LipDrive application. At the top, there is a blue header bar with a white back arrow on the left and the text "LipDrive" on the right. Below the header, there are five input fields arranged vertically. The first field is split into two boxes labeled "First Name" and "Last Name". The second field is labeled "Email". The third field is labeled "Username". The fourth field is labeled "Password" and has a small eye icon to its right, indicating a password toggle. Below these fields is a prominent blue button with the text "Sign UP" in white.

Figure 5: SignUP

3. Forget my password Screen



The screenshot shows the "Forget my password" screen of the LipDrive application. At the top, there is a blue header bar with the text "LipDrive" centered. Below the header, there are two input fields arranged vertically. The first field is labeled "New Password" and has a small eye icon to its right. The second field is labeled "Verify New Password". Below these fields is a prominent blue button with the text "Reset Password" in white.

Figure 6: Forgot password

4. Main Screens

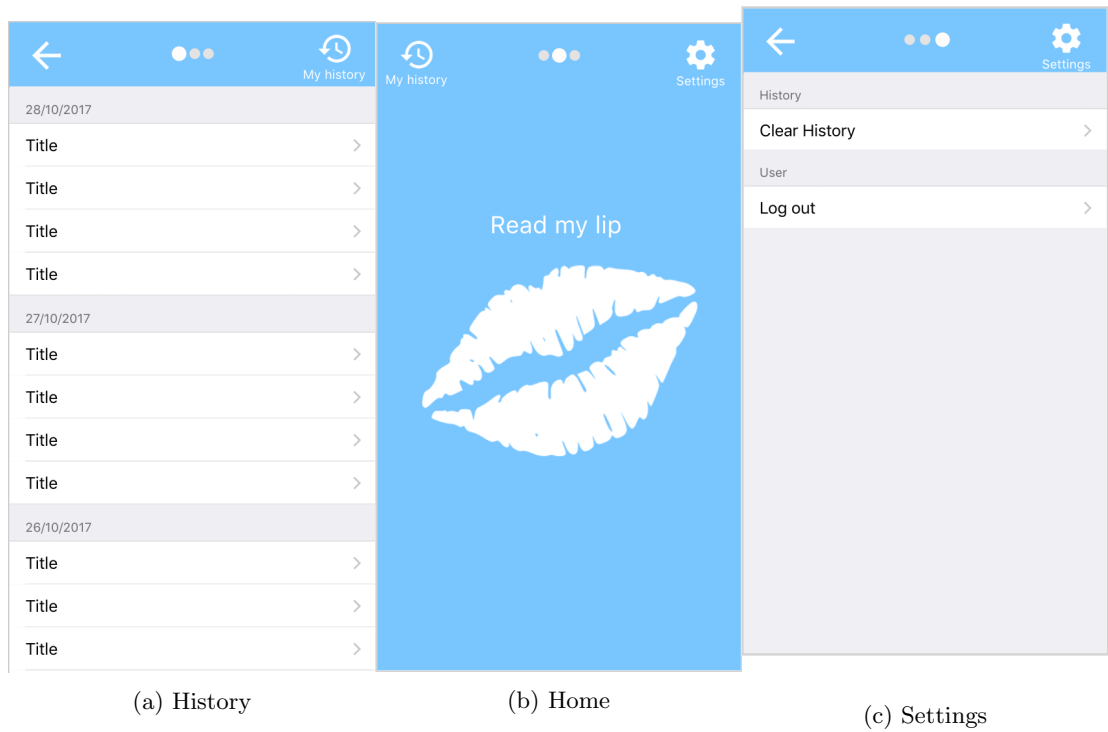


Figure 7: Main Screens

5. Camera Screen

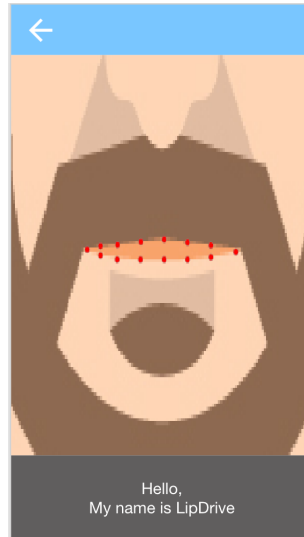


Figure 8: Camera

6. History details Screen

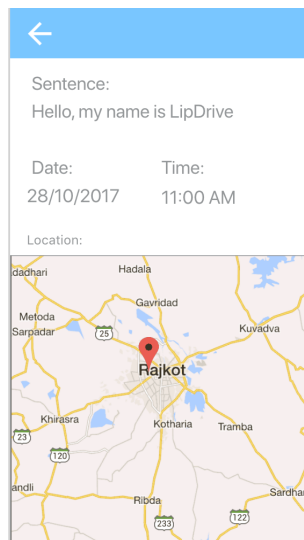


Figure 9: History Details

4.1.2 CLI

1. git
 - (a) to pull \$ git pull
 - (b) to add files \$ git add . or \$ git add *
 - (c) to make a commit \$ git commit -m "what sort of commit"
 - (d) to push to server \$ git push -u origin master
 - (e) to get status \$ git status

4.1.3 API

1. Facebook Login API
2. Google Login API
3. Google Maps API
4. DLib
5. Tensorflow
6. openCV

5 Design Constraints

5.1 Software constraints

The application runs on android 4.4 or IOS 9.5

6 Performance Requirements

For speech recognition, the system shall be able to process at least 25 frames per second and 40 frames per second in peak load.

For model training, the system must be able to handle large training datasets to ensure model accuracy. Sample run times for different training datasets are listed in Table 24.

Table 24: Performance measures from different datasets

Data set name	Data set Type	Frames Processed Per Second
MIRACL-VC1	Frame per frame	11
The GRID audiovisual sentence corpus	full video	40

7 Other non-functional attributes

7.1 Security

The username and password should be encrypted and the data transmitted to database should be saved securely

7.2 Maintainability

The system could be improved by different developers so its maintainability should be easy by documenting the code and the design and by using different design patterns as MVC design pattern, Singleton design pattern, Strategy design pattern

7.3 Portability

The system may be deployed on different mobile platforms (iOS/Android)

7.4 Usability

- Learnability: Proportion of functionalities or tasks mastered doesn't need time to be learned
- Memorability: This system is easy to be memorized due to the small number of tasks the user will do

7.5 Reliability

- Speed: The system aims to detect commands on real-time processing
- Accuracy: The accuracy of detection should be high that the user shall not re-order the command

8 Preliminary Object-Oriented Domain Analysis

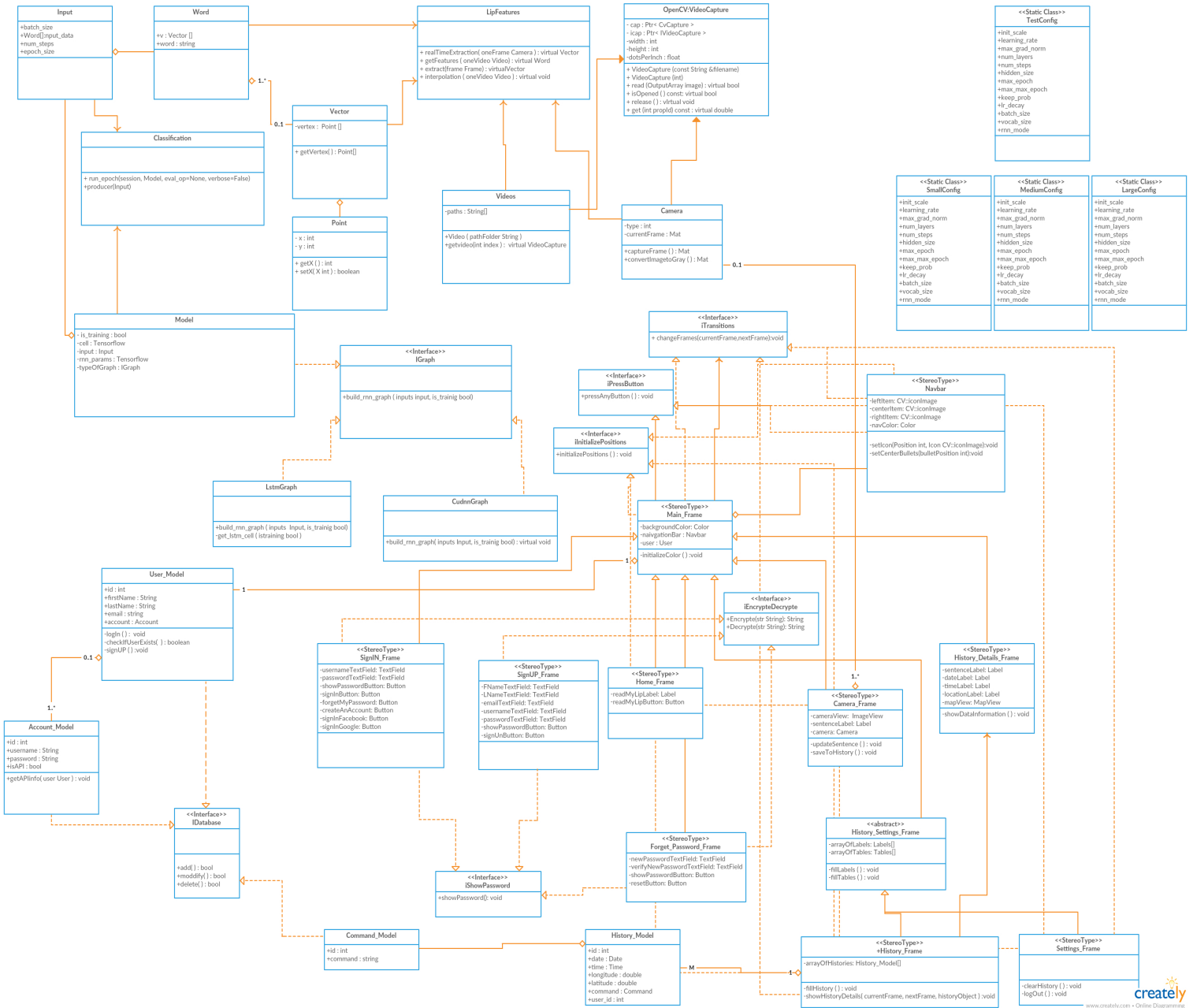


Figure 10: Class Diagram



8.1 Main_Frame Class

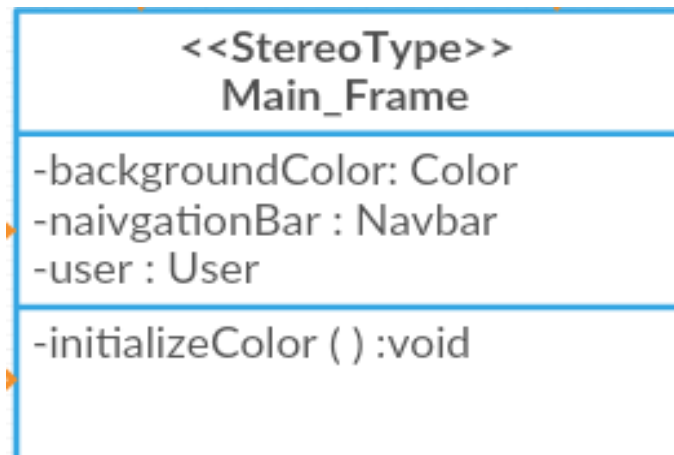


Figure 11: Main frame

1. Class name: `Main_Frame`
2. List of Superclasses: `iTransitions`, `iInitializePositions`, `iPressButton`
3. List of Subclasses: `Home_Frame`, `Camera_Frame`, `History_Details_Frame`, `History_Settings_Frame`, `SignUP_Frame`, `SignIN_Frame`, `Forget_Password_Frame`
4. Purpose: This class is used to define the basic frame with its background-color
5. Collaborations: This class must interact with the "Navbar" class to add the Navigation bar to the frame by a composition relationship. This class implements the "iTransitions" interface to be able to switch between different frames, the "iInitializePositions" interface to be able to set the positions of UI elements and display them, and it implements the "iPress-Button" interface to handle the Event Listeners to any button. This class has an association relationship with the "iTransitions" interface to let sub-classes able to switch frames, this class has also object user that is hold through out the project
6. Attributes:
 - (a) color `backgroundColor`
 - (b) Navbar `NavigationBar`
7. Operations:

- (a) void initializeColor(): This method is used to set the background color of the frame
- (b) void pressAnyButton(): This method is used to handle the events while interacting with buttons
- (c) void initializePositions(): This method is used to initialize the positions of UI components on the screen
- (d) void changeFrames(currentFrame,nextFrame): This method is used to switch between two different frames

8.2 Navbar Class

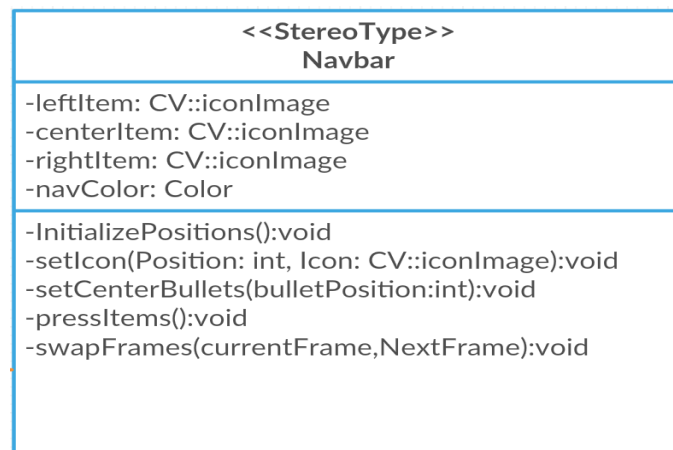


Figure 12: Navbar

1. Class name: Navbar
2. List of Superclasses: None
3. List of Subclasses: None
4. Purpose: This class is used to define different types of navigation bar according to different pages
5. Collaborations: This class helps in the composition of "Main_Frame" class to add the navigation bar to the top of the Frame
6. Attributes:
 - (a) CV::iconImage leftItem
 - (b) CV::iconImage centerItem
 - (c) CV::iconImage rightItem

(d) Color navColor

7. Operations:

- (a) void initializePositions(): This method is used to define the positions and the dimensions of the navigation bar
- (b) void setIcon(int Position, CV::iconImage Icon): This method is used to set the image icon for three different icons positions: the left/center/right icons
- (c) void setCenterBullets(int bulletPosition): This method is used in the "Home_Frame", "History_Frame" or "Settings_Frame" to add the three navigation bullets
- (d) void pressItems(): This method is used to handle events of the buttons in the navigation bar
- (e) void swapFrames(currentFrame, NextFrame): This method is used to switch between different frames

8.3 SignIN_Frame Class

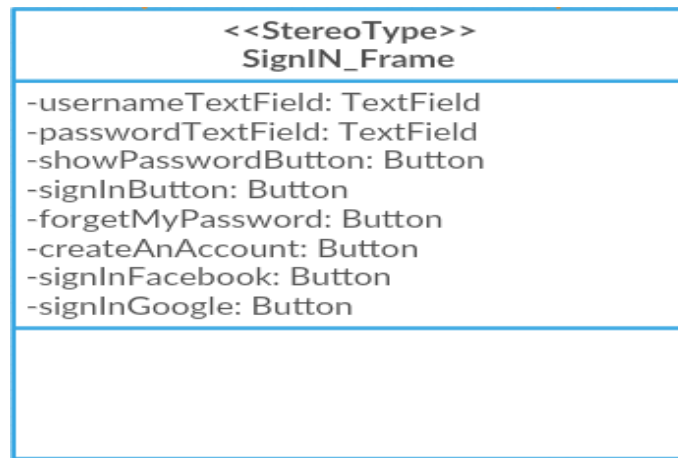


Figure 13: Sign in frame

1. Class name: SignIN_Frame
2. List of Superclasses: iShowPassword, iEncrypteDecrypte, Main_Frame
3. List of Subclasses: None
4. Purpose: This class is used to login to your application

5. Collaborations: This class inherits the "Main_Frame" class to initialize the layouts and navigation bar for this frame. It implements the "iShow-Password" interface to be able to show the hidden password in the passwordTextField, and it also implements the "iEncryptDecrypt" interface to encrypt and decrypt the password while logging in
6. Attributes:
 - (a) TextField usernameTextField
 - (b) TextField passwordTextField
 - (c) Button showPasswordButton
 - (d) Button signInButton
 - (e) Button forgetMyPasswordButton
 - (f) Button createAnAccount
 - (g) Button signInFacebook
 - (h) Button signInGoogle
7. Operations:
 - (a) void showPassword(): This method is used to show the password-TextField content
 - (b) String Encrypt(String str): This method is used to encrypt the passed string and returns the encrypted one
 - (c) String Decrypt(String str): This method is used to decrypt the passed string and returns the decrypted one

8.4 SignUP_Frame Class

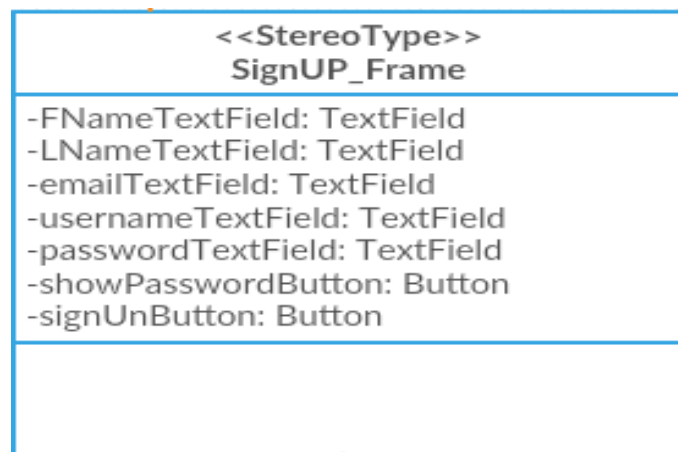


Figure 14: Sign up frame

1. Class name: SignUP_Frame
2. List of Superclasses: iShowPassword, iEncryptDecrypt, Main_Frame
3. List of Subclasses: None
4. Purpose: This class is used to sign up to your application
5. Collaborations: This class inherits the "Main_Frame" class to initialize the layouts and navigation bar for this frame. It implements the "iShowPassword" interface to be able to show the hidden password in the passwordTextField, and it also implements the "iEncryptDecrypt" interface to encrypt and decrypt the password while signing up in
6. Attributes:
 - (a) TextField FNameTextField
 - (b) TextField LNameTextField
 - (c) TextField emailTextField
 - (d) TextField usernameTextField
 - (e) TextField passwordTextField
 - (f) Button showPasswordButton
 - (g) Button signUpButton
7. Operations:
 - (a) void showPassword(): This method is used to show the password-TextField content
 - (b) String Encrypt(String str): This method is used to encrypt the passed string and returns the encrypted one
 - (c) String Decrypt(String str): This method is used to decrypt the passed string and returns the decrypted one

8.5 Forget_Password_Frame Class

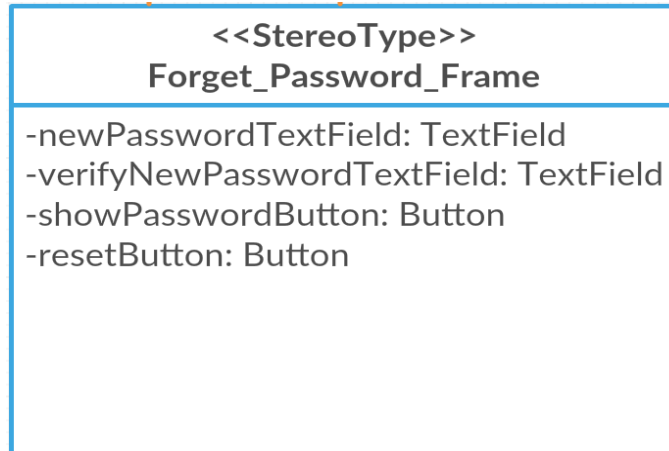


Figure 15: Forget password frame

1. Class name: `Forget_Password_Frame`
2. List of Superclasses: `iShowPassword`, `iEncryptDecrypte`, `Main_Frame`
3. List of Subclasses: None
4. Purpose: This class is used to change user's password if he forgot it
5. Collaborations: This class inherits the "Main_Frame" class to initialize the layouts and navigation bar for this frame. It implements the "iShowPassword" interface to be able to show the hidden password in the `passwordTextField`, and it also implements the "iEncryptDecrypt" interface to encrypt and decrypt the password while signing up in
6. Attributes:
 - (a) `TextField newPasswordTextField`
 - (b) `TextField verifyNewPasswordTextField`
 - (c) `Button showPasswordButton`
 - (d) `Button signUpButton`
7. Operations:
 - (a) `void showPassword()`: This method is used to show the `passwordTextField` content
 - (b) `String Encrypt(String str)`: This method is used to encrypt the passed string and returns the encrypted one
 - (c) `String Decrypt(String str)`: This method is used to decrypt the passed string and returns the decrypted one

8.6 Home_Frame Class

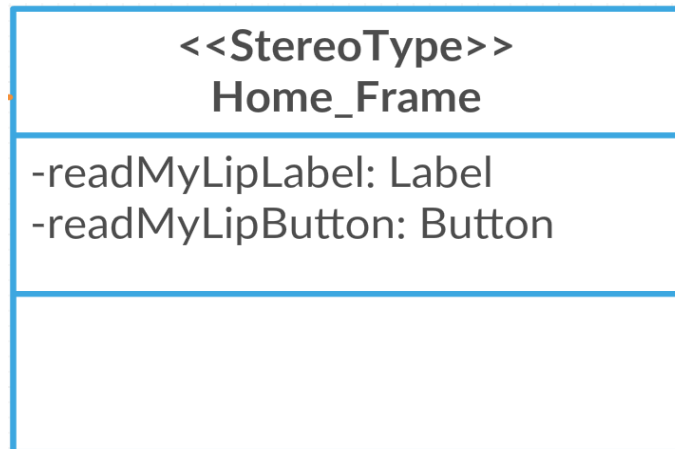


Figure 16: Home frame

1. Class name: Home_Frame
2. List of Superclasses: Main_Frame
3. List of Subclasses: None
4. Purpose: This class is the home page our application
5. Collaborations: This class inherits the "Main_Frame" class to initialize the layouts and navigation bar for this frame.
6. Attributes:
 - (a) Label readMyLipLabel
 - (b) Button readMyLipButton
7. Operations: None

8.7 Camera_Frame Class

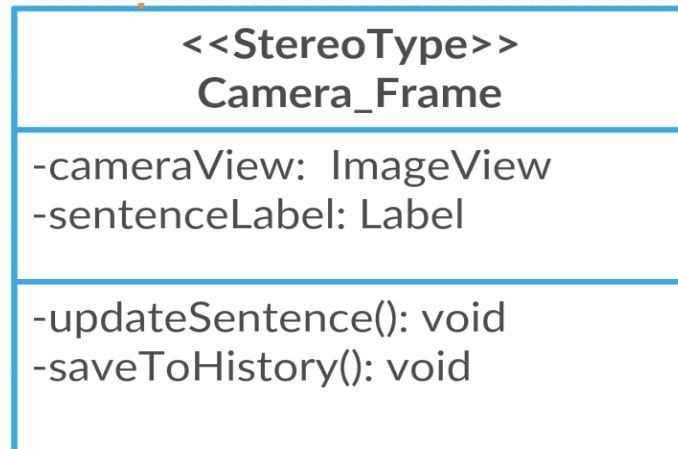


Figure 17: Camera

1. Class name: Camera_Frame
2. List of Superclasses: Main_Frame
3. List of Subclasses: None
4. Purpose: This class is used to take real-time images, extract features from them and classify them into commands (sentences)
5. Collaborations: This class inherits the "Main_Frame" class to initialize the layouts and navigation bar for this frame.
6. Attributes:
 - (a) ImageView cameraView
 - (b) Label sentenceLabel
7. Operations:
 - (a) void updateSentence(): This method is used to update the sentence label
 - (b) void saveToHistory(): This method is used to save the sentences into the database

8.8 History_Settings_Frame Class

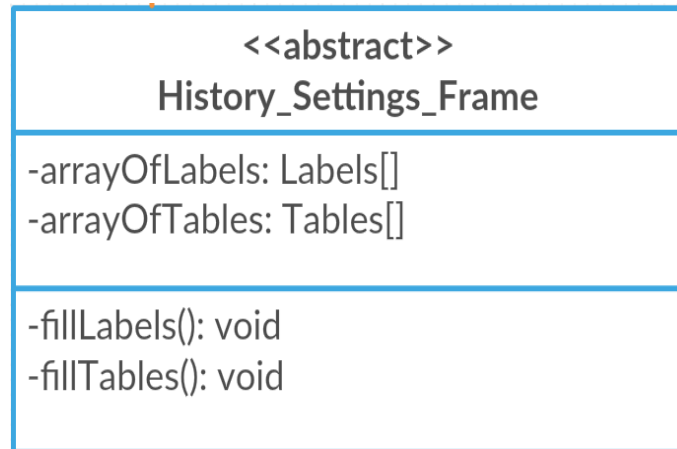


Figure 18: History Settings

1. Class name: `History_Settings_Frame`
2. List of Superclasses: `Main_Frame`
3. List of Subclasses: `History_Frame`, `Settings_Frame`
4. Purpose: This class is used to initialize the tables for histories and settings
5. Collaborations: This class inherits the "Main_Frame" class to initialize the layouts and navigation bar for this frame.
6. Attributes:
 - (a) `Label[] arrayOfLabels`
 - (b) `Tables[] arrayOfTables`
7. Operations:
 - (a) `void fillLabels()`: This method is used to display the array of history labels (Dates)
 - (b) `void fillTables()`: This method is used to display the corresponding table for each history label

8.9 History_Frame Class

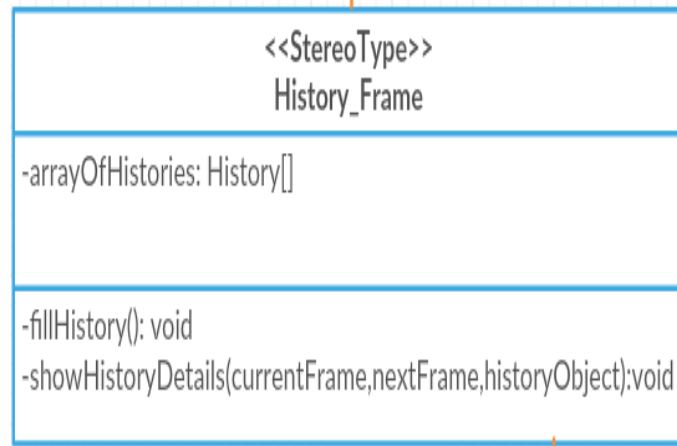


Figure 19: History

1. Class name: History_Frame
2. List of Superclasses: History_Settings_Frame
3. List of Subclasses: None
4. Purpose: This class is used to get all user's histories from the database and let user interact with each record to show its details
5. Collaborations: This class inherits the "History_Settings_Frame" class to initialize the labels and tables. This class has an association relationship with the "History_Details_Frame" class to switch from the current frame to the History_Details frame and send the history object to it
6. Attributes:
 - (a) History[] arrayOfHistories
7. Operations:
 - (a) void fillHistory(): This is method is used to fill the labels and tables with the retrieved data
 - (b) void showHistoryDetails(currentFrame,nextFrame,historyObject): This method is used to switch between the current frame and the History_Details frame and pass the history object to its constructor

8.10 Settings_Frame Class

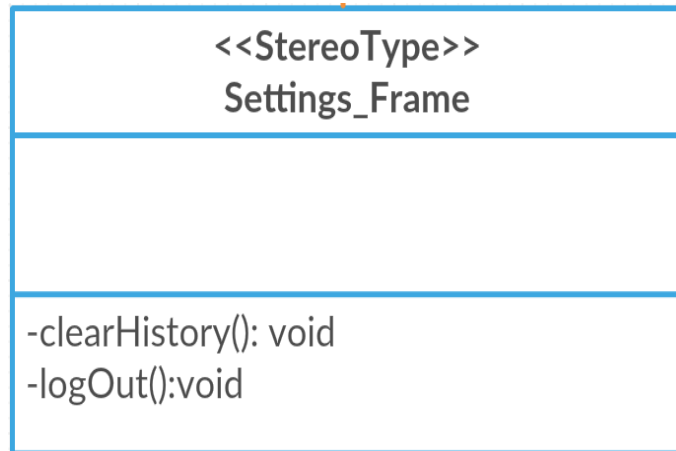


Figure 20: Settings frame

1. Class name: Settings_Frame
2. List of Superclasses: History_Settings_Frame
3. List of Subclasses: None
4. Purpose: This class is used to get all settings options
5. Collaborations: This class inherits the "History_Settings_Frame" class to initialize the labels and tables.
6. Attributes: None
7. Operations:
 - (a) void clearHistory(): This is method is used to clear the history records into the database
 - (b) void logout(): This method is used to destroy application's sessions and redirect to the SignIN_Frame

8.11 History_Details_Frame Class

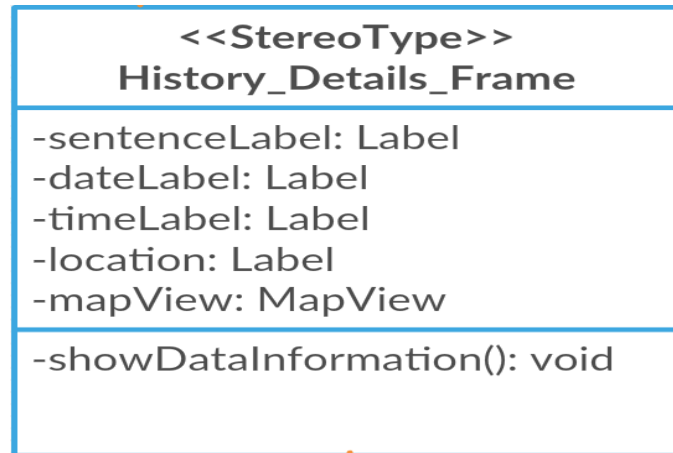


Figure 21: History details frame

1. Class name: `History_Details_Frame`
2. List of Superclasses: `Main_Frame`
3. List of Subclasses: None
4. Purpose: This class is used to display the details of the selected history
5. Collaborations: This class inherits the "Main_Frame" class to initialize the layouts and navigation bar for this frame.
6. Attributes:
 - (a) Label `sentenceLabel`
 - (b) Label `dateLabel`
 - (c) Label `timeLabel`
 - (d) Label `locationLabel`
 - (e) MapView `mapView`
7. Operations:
 - (a) `void showDataInformation():` This method is used to display the retrieved data from the constructor

8.12 Input Class

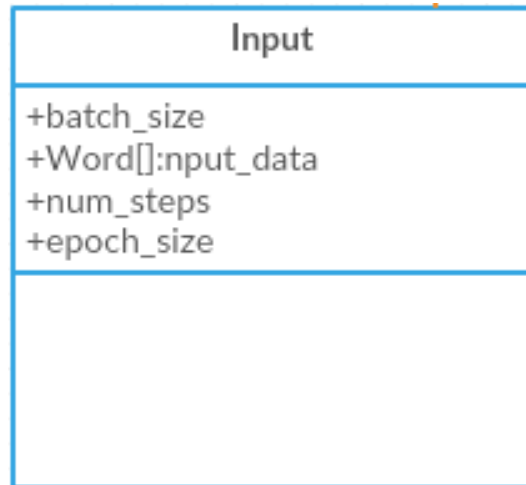


Figure 22: Input Class

1. Class name: Input
2. List of Superclasses: None
3. List of subclasses: None
4. Purpose: This class is used to have the data that will be trained and tested
5. Collaborations: Word through words
6. Attributes:
 - (a) Word words array of words
 - (b) Int batch_size
 - (c) Int num_steps
 - (d) Int epoch_size
7. Operations: None

8.13 Model Class

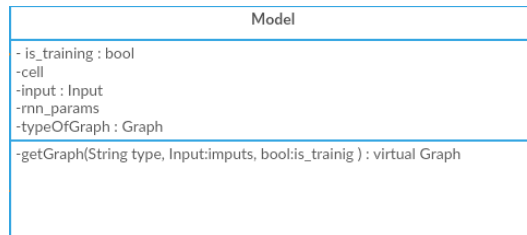


Figure 23: Model Class

1. Class name: Model
2. List of Superclasses: None
3. List of subclasses: None
4. Purpose: This class is used to have the model that will be trained on
5. Collaborations: Input through input, Grapg through typeOfGraph
6. Attributes:
 - (a) Bool is_trianing
 - (b) Input input the data
 - (c) Graph this wil be refrence on the algorithim will be used
7. Operations: None

8.14 Classification Class

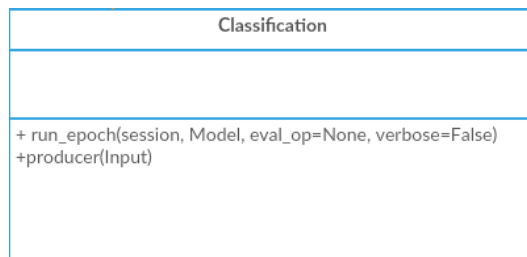


Figure 24: Classification Class

1. Class name: Classification
2. List of Superclasses: None

3. List of subclasses: None
4. Purpose: This class is to run train and testing
5. Collaborations: None
6. Attributes: None
7. Operations:
 - (a) run_epoch(session, Model)
This function is to classify the data
 - (b) producer(Input)
This function is to turn the data into batches and tensors

8.15 Point Class

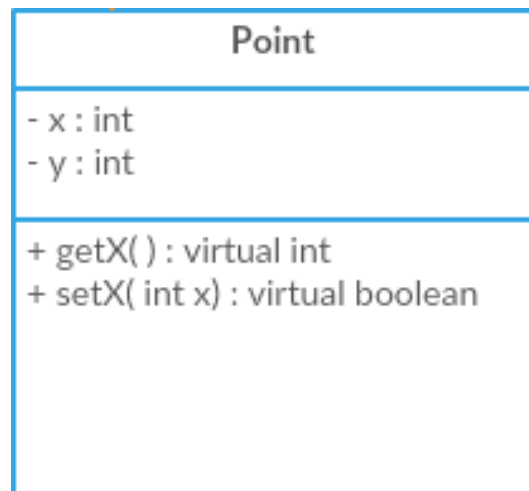


Figure 25: Point Class

1. Class name: Point
2. List of Superclasses: None
3. List of subclasses: None
4. Purpose: the position of the point relative to x axis and y to axis
5. Collaborations: None
6. Attributes:
 - (a) Int x

(b) Int y

7. Operations: None

8.16 Vector Class

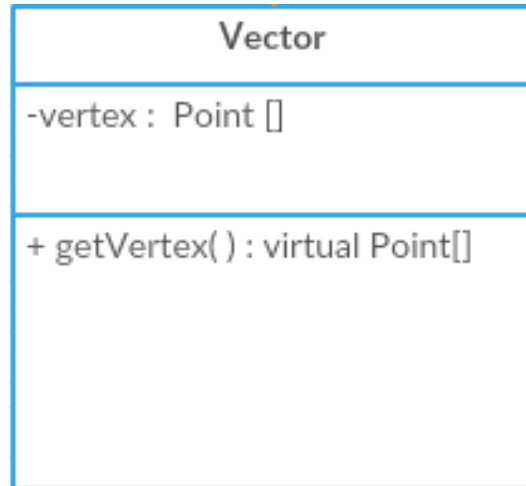


Figure 26: Vector Class

1. Class name: Vector
2. List of Superclasses: None
3. List of subclasses: None
4. Purpose: To hold the position of each point of the lips in one frame
5. Collaborations: Point through vertex
6. Attributes:
 - (a) Vector Vertex array of points
7. Operations: None

8.17 Word Class

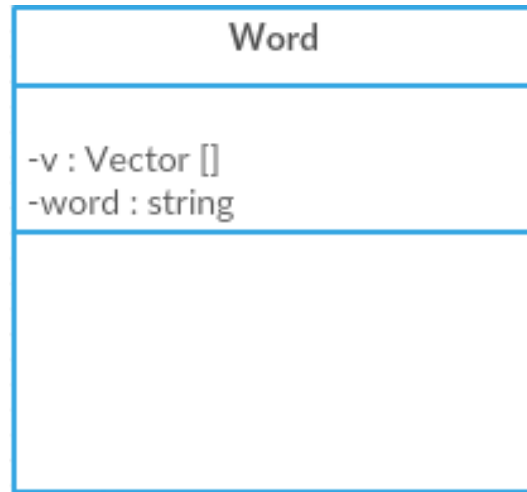


Figure 27: Word Class

1. Class name: Word
2. List of Superclasses: None
3. List of subclasses: None
4. Purpose: To hold the position of each point of the lips for the whole word
5. Collaborations: Vector through vec
6. Attributes:
 - (a) Vec array of Vector
 - (b) String the word
7. Operations: None

8.18 LstmGraph Class

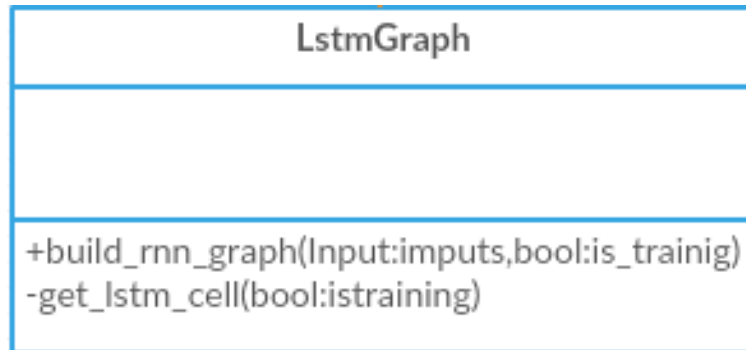


Figure 28: LSTMGraph Class

1. Class name: LstmGraph
2. List of Superclasses: IGraph
3. List of subclasses: None
4. Purpose: to build graph RNN based on Lstm
5. Collaborations: None
6. Attributes: None
7. Operations:
 - (a) `build_rnn_graph(Input:inputs,bool:is_trainig)`
this function chooses which RNN graph will be our classifier it is from interface Graph
 - (b) `get_lstm_cell(Config:config,bool:istraining)`
this function is the one making stack of LSTM layers in the graph

8.19 CudnnGraph

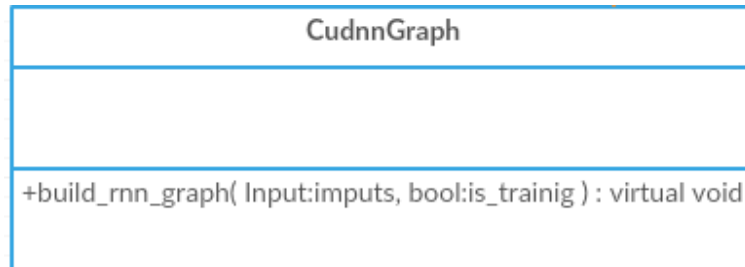


Figure 29: CUDNNGraph Class

1. Class name: CudnnGraph
2. List of Superclasses: IGraph
3. List of subclasses: None
4. Purpose: to build graph RNN based on Cudnn
5. Collaborations: None
6. Attributes: None
7. Operations:
 - (a) build_rnn_graph(Input:inputs,bool:is_trainig)
this function chooses which RNN graph will be our classifier it is from interface Graph

8.20 Resolution Class

1. Class name: Resolution
2. List of Superclasses: None
3. List of subclasses: None
4. Purpose: Used to hold frame width and height and pixel per inch
5. Collaborations: None
6. Attributes
 - (a) int height
 - (b) int width
 - (c) int pixelPerInch
7. Operations : None

8.21 OpenCV::VideoCapture Class

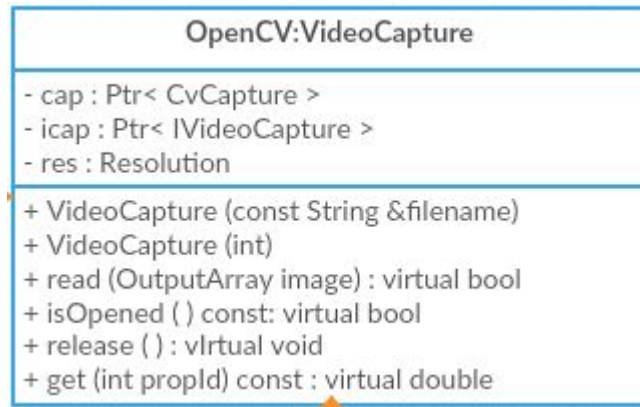


Figure 30: Video:CV

1. Class name: OpenCV::VideoCapture
2. List of Superclasses: A/N
3. List of subclasses: Videos, Camera
4. Purpose: Handles the camera and video files operations
5. Collaborations: this class aggregates Resolution class to hold the captured image resolution
6. Attributes
 - (a) Pointer CvCapture cap
 - (b) Pointer IVideoCapture icap
 - (c) Resolution res
7. Operations
 - (a) VideoCapture (const String &filename): This function is to open a video with a certain path
 - (b) VideoCapture (int): This function is to open camera
 - (c) bool read (OutputArray image): Grabs and returns the next video frame
 - (d) bool isOpened () const: Checks if video capturing is intialized or not
 - (e) void release (): destroy video capture
 - (f) double get (int propId) const: Returns the specified VideoCapture property

8.22 LipFeatures Class

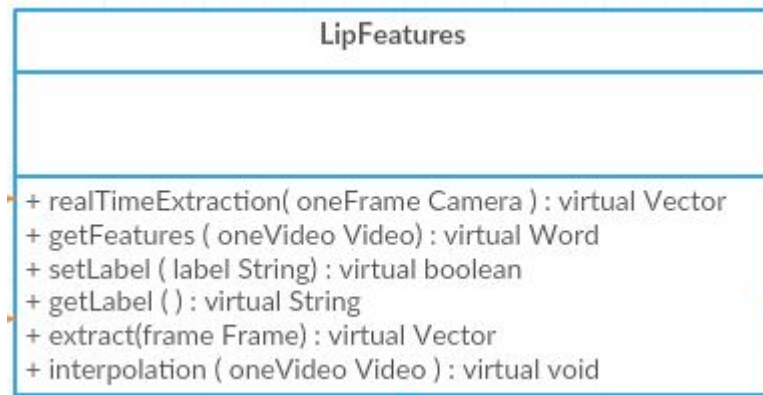


Figure 31: Lip Features

1. Class name: LipFeatures
2. List of Superclasses: None
3. List of subclasses: None
4. Purpose: using DLib CNN this class is used to handle extracting and formatting the features out of the captures images
5. Collaborations: this class associate with class Word and Vector to convert the image into extracted feature, camera to get real time fed images, video to get frames for one video
6. Attributes : None
7. Operations
 - (a) Vector realTimeExtraction(cam Camera): gets camera frame and extract
 - (b) Word videoExtraction (oneVideo Video): This function process on video frame by frame and extract the features out of it
 - (c) bool read (OutputArray image): Grabs and returns the next video frame
 - (d) Vector extractFeature(frame Frame): this function takes a frame (image) specifies features in it
 - (e) Video interpolation (oneVideo Video , NumberOfFrames int): this function takes the video and see the required number of frames and fill the empty spaces with average between two images

8.23 Videos class



Figure 32: Video Class

1. Class name: Videos
2. List of Superclasses: OpenCV:VideoCapture Class
3. Purpose: this class handle an area of paths of videos and handle loading and getting the videos
4. Collaborations: this class inherit OpenCV:VideoCapture Class to open camera and handle capturing frames
5. Attributes
 - (a) String[] paths
6. Operations
 - (a) Video (pathFolder String): constructor takes the folder path and collect all the paths of videos in it
 - (b) VideoCapture getvideo(int index): takes the index and returns the video to the index sent

8.24 Camera Class

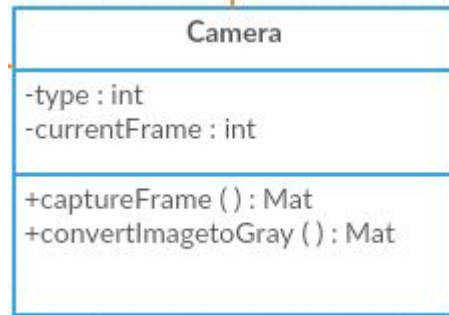


Figure 33: Camera class

1. Class name: Camera
2. List of Superclasses: OpenCV:VideoCapture Class
3. Purpose: this class is used to handle a real time videos coming from a camera
4. Collaborations: this class inherit OpenCV:VideoCapture Class to load videos
5. Attributes
 - (a) int type
 - (b) int currentFrame
6. Operations
 - (a) Mat captureFrame (): return cameras current frame
 - (b) Mat convertImagetoGray (Mat image): convert the image into gray scale

8.25 User_Model Class

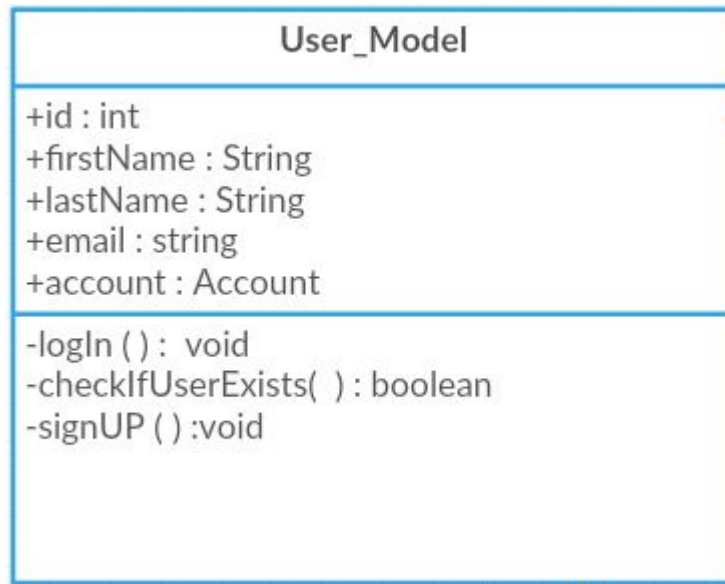


Figure 34: User_Model class

1. Class name: User_Model
2. List of Superclasses: IDatabase
3. Purpose: this class is used hold user info inside it
4. Collaborations: this class has account for security and is object of Main frame to be a single user using the app
5. Attributes
 - (a) int id
 - (b) string firstName
 - (c) string lastName
 - (d) string email
 - (e) account Account
6. Operations
 - (a) void login () : creates a session for the user to stay logged in through all pages
 - (b) boolean checkIfUserExists() : check if the user exists returns true if not returns false

- (c) void signUP () : creates new user in the database
- (d) void add() : add new User
- (e) void modify() : edit existing user
- (f) void delete() : delete existing user

8.26 Account_Model Class

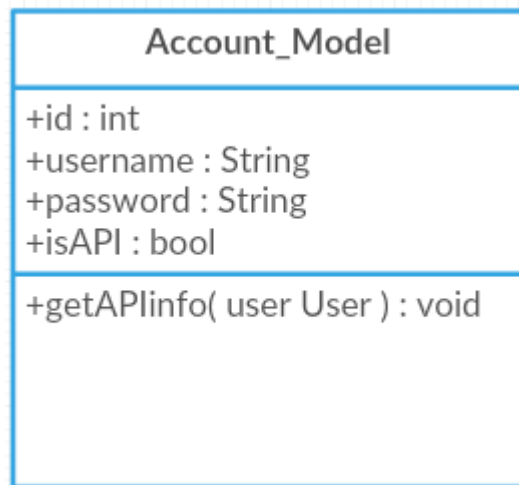


Figure 35: Account_Model class

1. Class name: Account_Model
2. List of Superclasses: IDatabase
3. Purpose: this class is used to hold account info
4. Collaborations: this class is an object of inside user
5. Attributes
 - (a) int id
 - (b) string username
 - (c) string password
 - (d) bool isAPI
6. Operations
 - (a) void getAPIinfo(user User) :return from a specific API the user info
 - (b) void add() : add new account
 - (c) void modify() : edit existing account
 - (d) void delete() : delete existing account

8.27 Command_Model Class

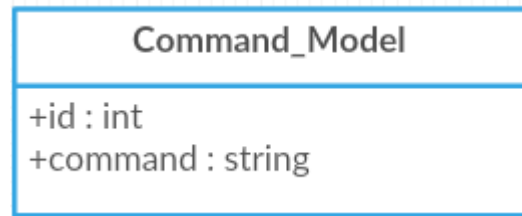


Figure 36: Command_Model class

1. Class name: Command_Model
2. List of Superclasses: IDatabase
3. Purpose: this class is used to hold Command info
4. Collaborations: this class is an object of inside History Model
5. Attributes
 - (a) int id
 - (b) string command
6. Operations
 - (a) void add() : add new command
 - (b) void modify() : edit existing command
 - (c) void delete() : delete existing command

8.28 History_Model Class

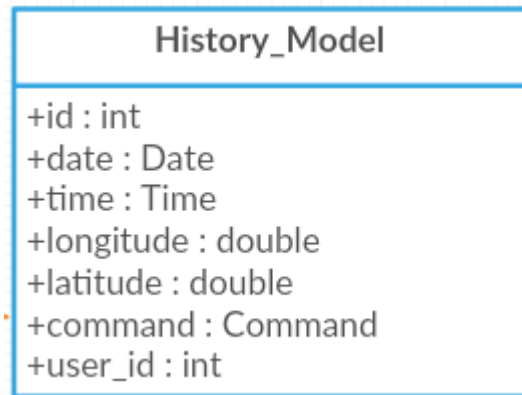


Figure 37: History_Model class

1. Class name: History_Model
2. List of Superclasses: IDatabase
3. Purpose: this class is used to hold History info
4. Collaborations: this class is an object of inside user
5. Attributes
 - (a) int id
 - (b) Time time
 - (c) Date date
 - (d) longitude double
 - (e) latitude double
 - (f) Command command
 - (g) user_id int
6. Operations
 - (a) void add() : add new history
 - (b) void modify() : edit existing history
 - (c) void delete() : delete existing history

9 Operational Scenarios

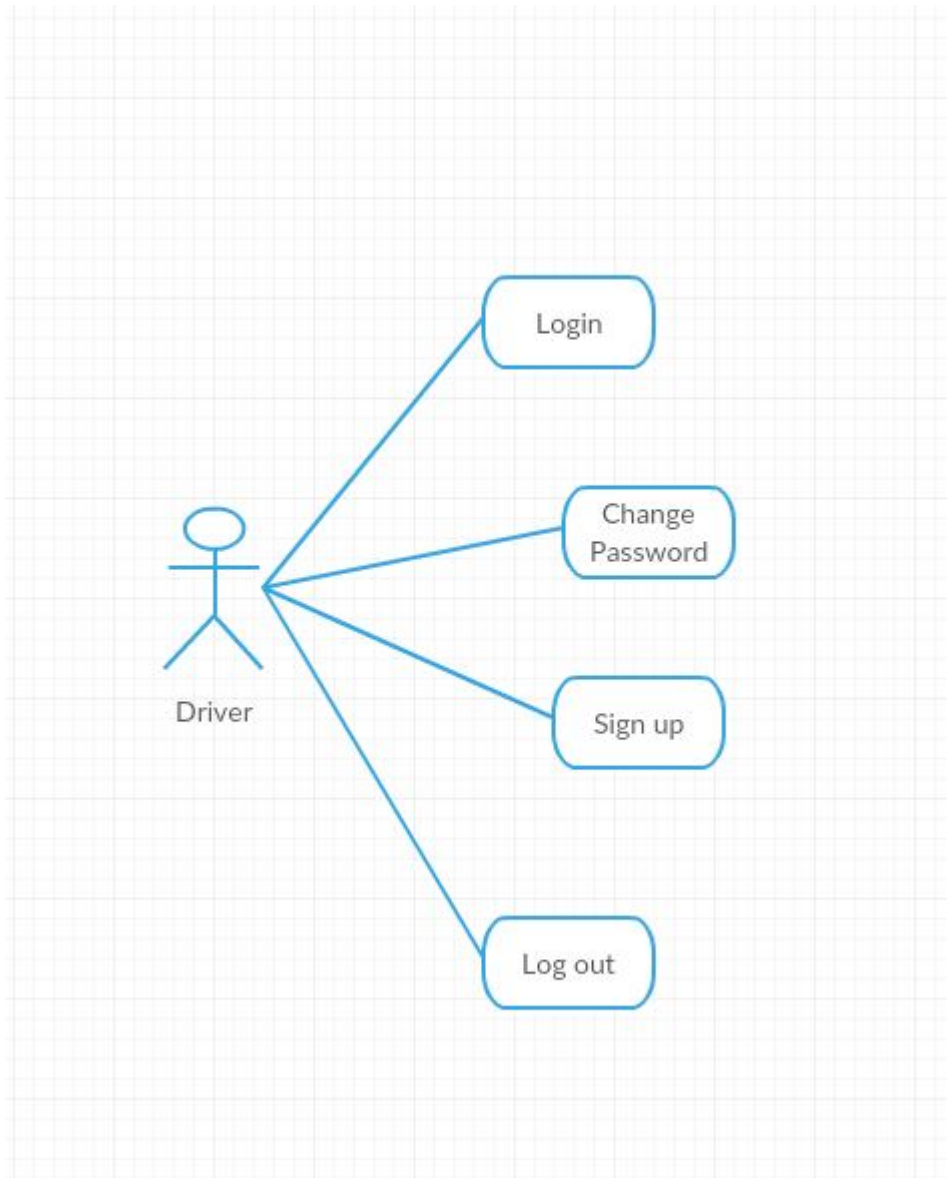


Figure 38: use case 1

9.1 Use Case: Login [using username and password]

- Scope: Login form

- Level: User goal
- Intention in context: The intention of the user is to login by using username and password to use the application
- Primary actor: Registered user
- Main success scenario:
 1. User enters username and password
 2. System validates these inputs
 3. System login to profile
- Extensions:
 - 2.a User enters invalid input
 - * System informs the user, then back to step 1
 - 2.b User cancels the login
 - * Use case ends in failure

9.2 Use Case: Login [using Lip movement sequence]

- Scope: Login Form
- Level: User goal
- Intention in context: The intention of the user is to login by using lip movement sequence to use the application
- Primary actor: Registered user
- Main success scenario:
 1. User opens camera and uses lips movement sequence
 2. System validates these sequence
 3. System login to profile
- Extensions:
 - 2.a User moves invalid sequence
 - * System informs the user, then back to step 1
 - 2.b User cancels the login
 - * Use case ends in failure

9.3 Use Case: Logout

- Scope: Settings menu
- Level: User goal
- Intention in context: The intention of the user is to logout from the application
- Primary actor: Registered user
- Main success scenario:
 1. user opens settings, then presses logout button to logout.
 2. System logout from user's profile
- Extensions: none

9.4 Use Case: Sign-Up [using username and password]

- Scope: Registration form
- Level: User goal
- Intention in context: The intention of the user is to create an account on the application.
- Primary actor: Registering user
- Main success scenario:
 1. User enters first name, last name and e-mail.
 2. User enters username and password and confirm password.
 3. System validates that password and confirm password are the same
 4. System login to profile
- Extensions:
 - 2.a User enters different password in password and confirm password field
 - * System informs user, then back to step 1
 - 2.b User cancels to Signup
 - * Use case ends in failure

9.5 Use Case: change password [using input password]

- Scope: change password form
- Level: User goal
- Intention in context: The intention of the user is to change password if forgotten
- Primary actor: Registered user
- Main success scenario:
 1. User enters new password
 2. System asks to enter the password again for confirmation
 3. System changes the password
- Extensions:
 - 2.a User enters different password than that in the confirmation field
 - * System informs the user, then back to step 1
 - 2.b User cancels to Signup
 - * Use case ends in failure

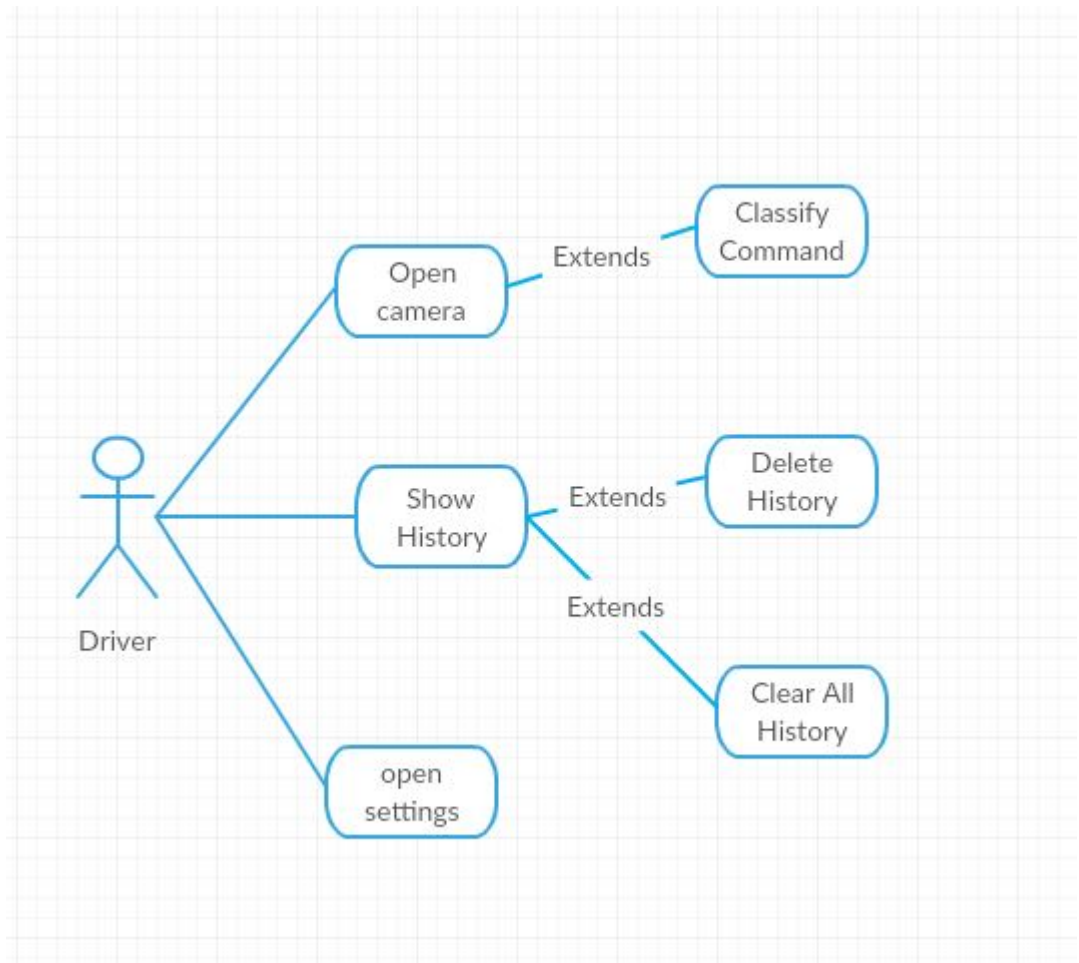


Figure 39: use case 2

9.6 Use Case: Open Camera

- Scope: LipDrive main screen
- Level: User goal
- Intention in context: The intention of the user is to open camera to start classifying a certain command
- Primary actor: Registered user
- Main success scenario:
 1. User presses on the Lip button to open the camera

2. Application opens the camera

- Extensions: none

9.7 Use Case: Classify a Command

- Scope: Camera frame
- Level: User goal
- Intention in context: The intention of the user is to order a certain command to the camera
- Primary actor: Registered user
- Main success scenario:
 1. User starts ordering the camera
 2. System extract the features of the lips
 3. System starts to compare the lips movements with the trained command
 4. System classifies the command
- Extensions:
 - 2.a System fails to extract the lips' features
 - * System informs user to adjust camera clearly, then back to step 1
 - 2.b User cancels to classify a command
 - * Use case ends in failure
 - 3.a System couldn't find the classified ordered
 - * System informs user to re-order the command, then back to step 1
 - 3.b User cancels to classify a command
 - * Use case ends in failure

9.8 Use Case: Show History

- Scope: History view
- Level: User goal
- Intention in context: The intention of the user is to show all previous classified orders
- Primary actor: Registered user

- Main success scenario:
 1. User opens History view
 2. Histories are the shown by the application by date
- Extensions: none

9.9 Use Case: Delete History

- Scope: History view
- Level: User goal
- Intention in context: The intention of the user is to delete specific history
- Primary actor: Registered user
- Main success scenario:
 1. User opens History view
 2. Histories are the shown by the application by date
 3. User select a record in the history
 4. User deletes selected history
- Extensions: none

9.10 Use Case: Clear All History

- Scope: Settings view
- Level: User goal
- Intention in context: The intention of the user is to clear all previous classified orders
- Primary actor: Registered user
- Main success scenario:
 1. User opens settings view
 2. User presses on clear all history button
 3. System clears all history
- Extensions: none

9.11 Use Case: Open Settings

- Scope: Settings view
- Level: User goal
- Intention in context: The intention of the user is to open application settings
- Primary actor: Registered user
- Main success scenario:
 1. User opens settings view
 2. System shows all available settings
- Extensions: none

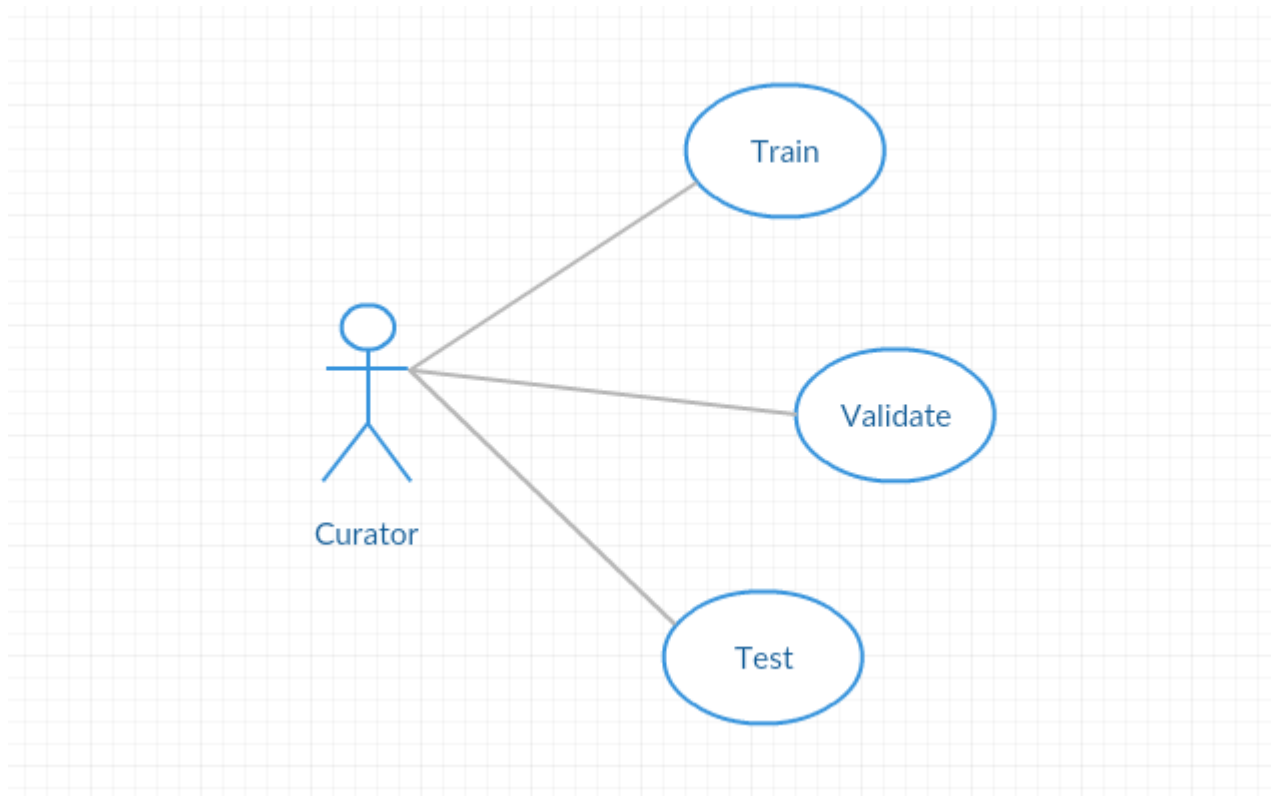


Figure 40: use case 1

9.12 Use Case: Train

- Scope: Command Line
- Level: user goal
- Intention in context: The intention of the user to insert a training dataset into the RNN
- Primary actor: User Curator
- Main success scenario:
 1. The user choose a specific folder with dataset in it
 2. User then should press start training
 3. System starts extracting features and feed it to the RNN
 4. System outputs results of training
- Extensions:
 - 2.a User enters invalid path
 - * System informs the user, then back to step 1
 - 2.b User cancels the training process
 - * Use case ends in failure
 - 2.c User enters invalid dataset extensions
 - * Use case ends in failure

9.13 Use Case: Test

- Scope: Command Line
- Level: user goal
- Intention in context: The intention of the user to insert a testing dataset into the RNN
- Primary actor: User Curator
- Main success scenario:
 1. The user choose a specific folder with dataset in it
 2. User then should press start testing
 3. System starts extracting features and feed it to the RNN
 4. System outputs results of testing
- Extensions:
 - 2.a User enters invalid path

- * System informs the user, then back to step 1
- 2.b User cancels the testing process
 - * Use case ends in failure
- 2.c User enters invalid dataset extensions
 - * Use case ends in failure

9.14 Use Case: Validate

- Scope: Command Line
- Level: user goal
- Intention in context: The intention of the user to insert a validating dataset into the RNN
- Primary actor: User Curator
- Main success scenario:
 1. the user choose a specific folder with dataset in it
 2. user then should press start validating
 3. system start extracting features and feed it to the RNN
 4. system output results of validating
- Extensions:
 - 2.a User enters invalid path
 - * System informs the user, then back to step 1
 - 2.b User cancels the validating process
 - * Use case ends in failure
 - 2.c User enters invalid dataset extensions
 - * Use case ends in failure

10 Design Constraints

10.1 Software constraints

The application runs on android 4.4 or IOS 9.5

11 Preliminary Schedule Adjusted

Task Name	Start	Finish
Idea discussion	07/26/17	08/11/17
Idea Research	08/15/17	09/15/17
Survey and proposal	09/16/17	09/25/17
Proposal presentation	09/26/17	09/26/17
Designing application	10/03/17	10/06/17
Implementing prototype	10/07/17	10/10/17
Implementing GUI desig	10/11/17	10/13/17
Designing database	10/14/17	10/17/17
Class diagram	10/18/17	10/20/17
SRS writing	10/21/17	11/08/17
SRS presentation	11/09/17	11/09/17
Data set collection	11/10/17	11/17/17
Data classification	11/18/17	11/24/17
Implementing applicatio	11/25/17	12/22/17
SDD writing	12/23/17	01/16/18
SDD presentation	01/19/18	01/19/18
Validation and testing	01/31/18	03/27/18
Implementation evaluati	03/29/18	03/29/18
Writing paper	03/31/18	04/10/18
Delivering the paper	04/12/18	04/12/18
Writing thesis	04/21/18	05/31/18
Final presentation	06/26/18	06/27/18

Figure 41: Time plan

12 Preliminary Budget Adjusted

iPhone SE 16GB 6,950 LE

References

- [1] Neeru Rathee "A novel approach for lip Reading based on neural network",18 July 2016 in IEEE
- [2] Bor-Shing Lin, Yu-Hsien Yao, Ching-Feng Liu "Development of Novel Lip-reading Recognition Algorithm",09 January 2017 in IEEE
- [3] Chung, Joon Son, and Andrew Zisserman. *Lip Reading in the Wild*. Springer-Link, Springer, Cham, 20 Nov 2016.
- [4] Assael, Yannis M., et al. *LipNet: End-to-End Sentence-Level Lipreading.* , 16 Dec. 2016
- [5] Mengchen Liu, Jiaxin Shi, Zhen Li "Towards Better Analysis of Deep Convolutional Neural Networks", 09 August 2016 in IEEE
- [6] Sonu Lamba, Neeta Nain, Harendra Chahar "A Robust Multi-Model Approach for Face Detection in Crowd",24 April 2017 in IEEE
- [7] Amit Garg , Jonathan Noyola, Sameep Bagadia "Lip reading using CNN and LSTM", 2016 in stanford.edu
- [8] Fatemeh Sadat Lesani, Faranak Fotouhi Ghazvini, Rouhollah Dianat "Mobile phone security using automatic lip reading",16 July 2015 in IEEE
- [9] Daehyun Lee, Kyungsik Myung "Read My Lips, Login to the Virtual World",30 March 2017 in IEEE
- [10] Seksan Mathulapransan, Chien-Yao Wang, Aufaclav Zatu Kusum "A Survey of Visual Lip Reading and Lip-Password Verification",23 June 2016 in IEEE
- [11] N. Radha, A. Shahina, A. Nayeemulla Khan "A person identification system combining recognition of face and lip-read passwords",18 February 2016 in IEEE
- [12] N. Radha, A. Shahina, A. Nayeemulla Khan "Automated Lip Reading Technique for Password Authentication", September 2012 in ijais.org
- [13] A. Rekik, A. Ben-Hamadou, and W. Mahdi, A new visual speech recognition approach for RGB-D cameras, in *Image Analysis and Recognition - 11th International Conference, ICIAR 2014, Vilamoura, Portugal, October 22-24, 2014, Proceedings*, Part II, 2014, pp. 2128.