Software Requirements Specification

for

Arabic Language

Dialect Identification

**Version 1.0 approved**

**Prepared by Group 6**

**The American University in Cairo**

**9 / 12 / 2017**

**Table of Contents**

[**Introduction**](#_7t5xqx40qek0) **5**

[**Purpose**](#_3znysh7) **5**

[**Document Conventions**](#_kgmq556bphb7) **5**

[**Intended Audience and Reading Suggestions**](#_cbtxdvlauar) **5**

[**Product Scope**](#_ttiv5qv6bi7) **5**

[**Overall Description**](#_gpz41lma77xo) **6**

[**Product Perspective**](#_8e96cgj6un6v) **6**

[**Product Functions**](#_ewacr5g7ajt6) **6**

[**User Classes and Characteristics**](#_56u9am9se2g4) **6**

[**Operating Environment**](#_6jogt83oyek2) **6**

[**Design and Implementation Constraints**](#_52b2hc8iua8t) **7**

[**Assumptions and Dependencies**](#_m4zkh0c730qt) **7**

[**External Interface Requirements**](#_t9h0oma1kxwi) **8**

[**User Interfaces**](#_7b054ochcm2i) **8**

[**Hardware Interfaces**](#_557nkh3goya1)[**Software Interfaces**](#_8a9e6ukv6lsb) **10**

[**Communication Interfaces**](#_x7d6foz06opo) **11**

[**System Features**](#_8jk3iz9d60e5) **12**

[**Record Audio**](#_uyfnmsxm9cvq) **12**

[**View Supported Dialects**](#_aqotts9dlj9k) **12**

[**Send Speech Sample**](#_qqr2668x6t43) **13**

[**View History**](#_tcqzh1nm9eio) **13**

[**Identify Dialect**](#_tjn2g07w7vqu) **14**

[**Other Nonfunctional Requirements**](#_3whwml4) **15**

[**Performance Requirements**](#_2bn6wsx) **15**

[**Security Requirements**](#_goyz23u70upp) **15**

[**Software Quality Attributes**](#_71lj7q5eifze) **15**

[**Appendix A: Glossary**](#_8buoqhnzj9py) **16**

[**Appendix B: Analysis Models**](#_us5crxc9a9uo) **17**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| First draft | 15/11/2017 | Editing project description | 0.1 |
| Second draft | 21/11/2017 | Adding some system features | 0.2 |
| Final draft | 9/12/2017 | Editing use cases and activity diagrams | 1.0 |

# Introduction

## Purpose

This document identifies the requirements of Arabic Dialect Identification system. This is the first release of the SRS. This system is to be used by members in any society without having to be associated with a specific company or corporation.

## Document Conventions

The bullet point (●) is used when listing more than one point in the same part

## Intended Audience and Reading Suggestions

This document is to be presented to users as well as the thesis supervisor, thesis coordinator, and any other professor involved. This document includes a concise introduction to the system along with an overall description of the product and the features provided within it. It describes the functions briefly and the interactions between the system components.

## Product Scope

The motivation of the project is to facilitate the communication between different Arabs speaking different Arabic dialects, as well as serve as the first step in real-time translation. The inspiration of the project came from Microsoft Research Labs (ATLC) to be used in their product, Skype. The project aims to identify the Arabic dialect spoken when given a sample speech input.

# Overall Description

## Product Perspective

This is a new and self-contained product. It could also be integrated in other bigger applications such as a real-time translator.

## Product Functions

* Accept audio as input
* Detect the dialect spoken in the audio
* Print the dialect as output

## User Classes and Characteristics

2.3.1 Users of the API

* Application developers

2.3.2 Users of the mobile application

* Educational environments
* Work environments
* Social spheres

## Operating Environment

2.4.1 API

* 64-bit​ ​Linux
* Python​ ​2.7
* NVIDIA​ ​CUDA®​ ​7.5​ ​(CUDA​ ​8.0​ ​required​ ​for​ ​Pascal​ ​GPUs)
* NVIDIA​ ​cuDNN​ ​v4.0​ ​(minimum)​ ​or​ ​v5.1​ ​(recommended)
* Keras​ ​library​ ​with​ ​Tensorflow​ ​backend

2.4.2 Mobile Environment

* Android

## Design and Implementation Constraints

* Availability​ ​of​ ​data​ ​(Audio ​of​ Arabic speech with different dialects ​for​ ​training​ ​the​ ​neural​ ​network)
* Availability​ ​of​ ​PC’s​ ​(for​ ​training​ ​the​ ​neural​ ​network)
* Limited​ ​number​ ​of​ Arabic dialects ​recognized​ ​by​ ​the​ ​network

## Assumptions and Dependencies

2.6.1 Assumptions

* The​ ​release​ ​of​ ​a​ ​better​ ​framework​ ​for​ ​machine​ ​learning​ ​might​ ​affect​ ​the​ ​operating environment

2.6.2 Dependencies

* Pre-trained​ ​models​ ​for​ ​transfer​ ​learning​.

# External Interface Requirements

## User Interfaces

* + 1. Homepage

The user records a speech segment using the mic of the phone by pressing on the mic icon..

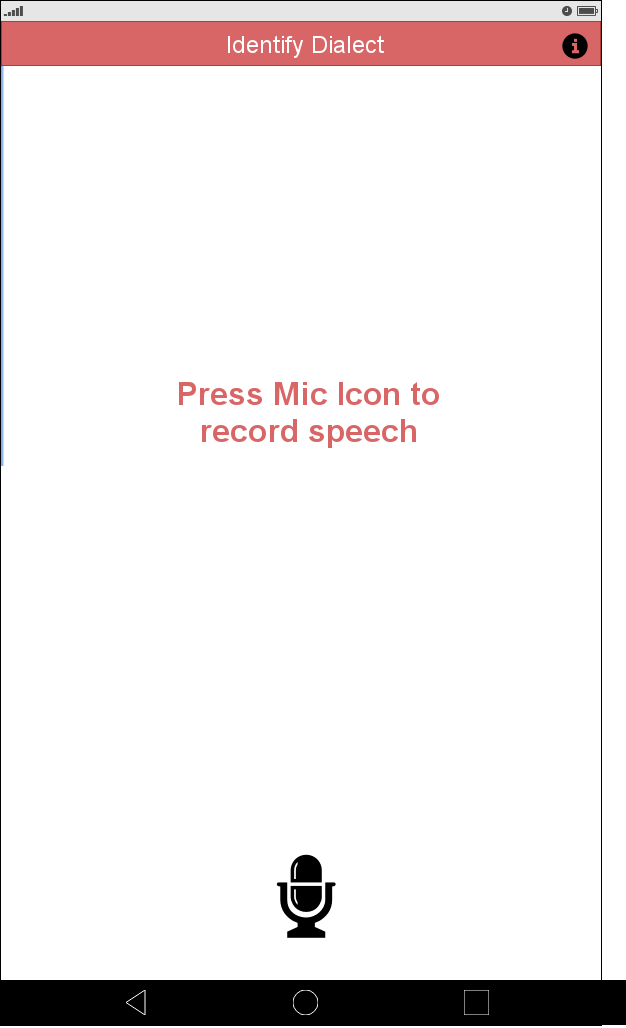
**

Figure 1 | Homepage

3.1.2 History

The user can view previously identified dialects using the history feature (if any).

**

Figure 2 | History

3.1.3 Information

The user can view the supported dialects that can be detected using the mobile application.

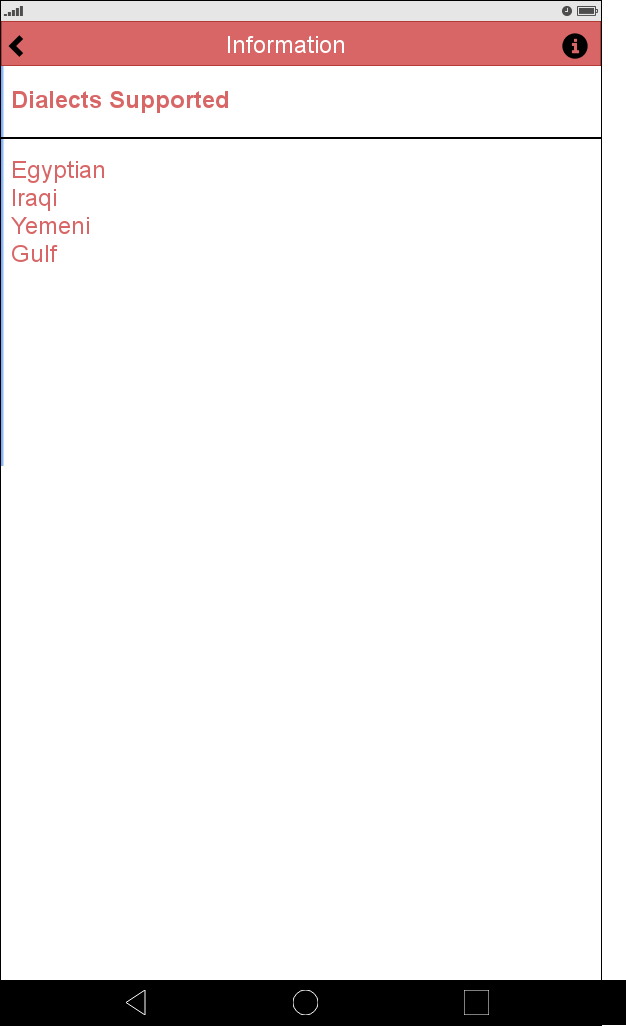
**

Figure 3 | Supported Dialects

## Hardware Interfaces The primary approach chosen towards solving the problem is the normal client-server architecture. The ultimate goal we will try to pursue is a no-server architecture, running everything on the stand-alone mobile app.

* + 1. Client-Server Architecture  
        The server must use high-end GPUs to be able to process neural network training. It should also have a lot of memory and disk space to be able to handle a bigger number of connected users
    2. No-Server Architecture

In the no-server architecture, everything would be embedded on the mobile device itself and there is no need for a server.

## Software Interfaces

* + 1. Client-Server Architecture

The server software will be running a server oriented Linux operating system such as CentOS or Ubuntu. Keras API is used with Tensorflow background to train and test the neural networks. The client side would be responsible for preprocessing the speech segments and sending it to the server for dialect identification.

## 3.3.2 No-Server Architecture

The mobile application would be able to preprocess speech segments and load the weights of the pre-trained networks and identify the dialect spoken.

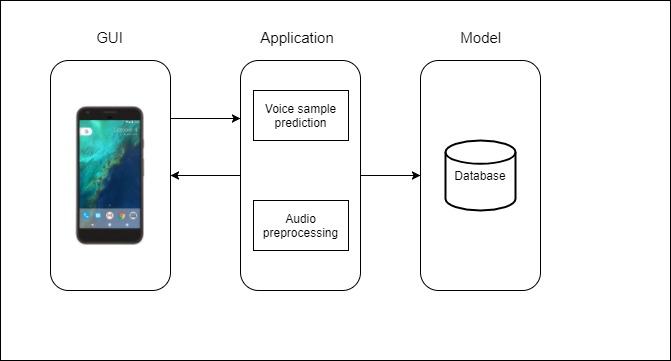


Figure 4 | System Architecture

## Communication Interfaces

HTTP communication between client and server, assuming a client-server architecture is chosen.

# System Features

## Record Audio

|  |  |
| --- | --- |
| **Use Case ID** | SF-01 |
| **Name** | Record Audio |
| **Precondition** | User touches record button in the mobile application |
| **Postcondition** | Speech is recorded |
| **Normal Flow** | Speech samples are recorded and sent to the server |
| **Actor** | User |

## 

## View Supported Dialects

|  |  |
| --- | --- |
| **Use Case ID** | SF-02 |
| **Name** | View Supported Dialects |
| **Precondition** | Network is trained on some dialects |
| **Postcondition** | A list containing supported dialects is shown |
| **Normal Flow** | User selects the option to view the supported dialects and the list is shown on their screen |
| **Actor** | User |

## Send Speech Sample

|  |  |
| --- | --- |
| **Use Case ID** | SF-03 |
| **Name** | Send Speech Sample |
| **Precondition** | Speech is being recorded by user’s device |
| **Postcondition** | Recorded samples continuously sent to the server |
| **Normal Flow** | User selects the record button, the app starts capturing their speech, and samples are sent to the server for prediction |
| **Actor** | - |

## 

## View History

|  |  |
| --- | --- |
| **Use Case ID** | SF-04 |
| **Name** | View History |
| **Precondition** | Neural Network is trained and user’s previous speech segments were identified |
| **Postcondition** | Dialect identification history is shown |
| **Normal Flow** | User selects the option to view history and the list is shown on their screen |
| **Actor** | User |

## 

## Identify Dialect

|  |  |
| --- | --- |
| **Use Case ID** | SF-05 |
| **Name** | Identify dialect |
| **Precondition** | Speech sample is sent to the server |
| **Postcondition** | Sample is fed to the network to predict the dialect spoken |
| **Normal Flow** | Server receives speech, them predicts dialect spoken |
| **Actor** | Server |

## 

# Other Nonfunctional Requirements

## Performance Requirements

* Real-time performance
* High classification accuracy
* Network data usage
* Battery consumption

## Security Requirements

* Speech samples are not saved to ensure user’s privacy

## Software Quality Attributes

* User-friendly GUI
* Sufficient, correct, high-quality training data

## Appendix A: Glossary

|  |  |
| --- | --- |
| **Acronym** | **Meaning** |
| GUI | Graphical User Interface |
| API | Application Programming Interface |
| HTTP | Hypertext Transfer Protocol |
| ATLC | Microsoft Advanced Technology Labs in Cairo |

## Appendix B: Analysis Models

**Use Case:**

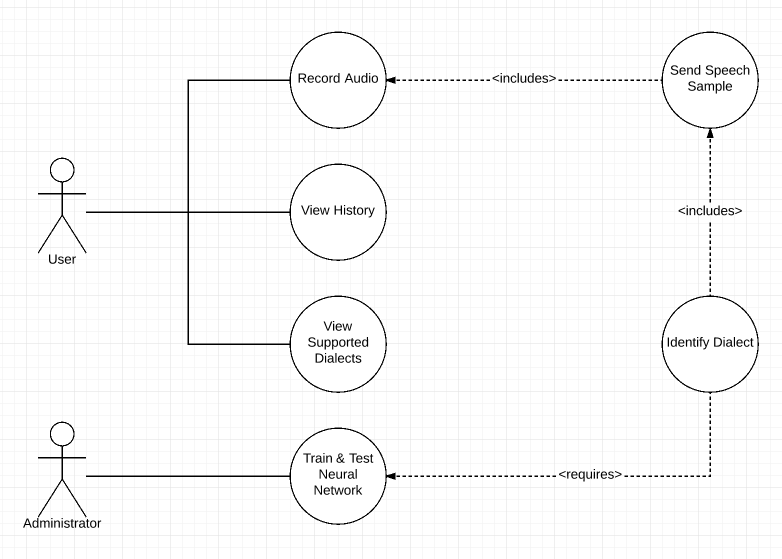


Figure 5 | Dialect Identification Use Case

**Activity Diagram:**

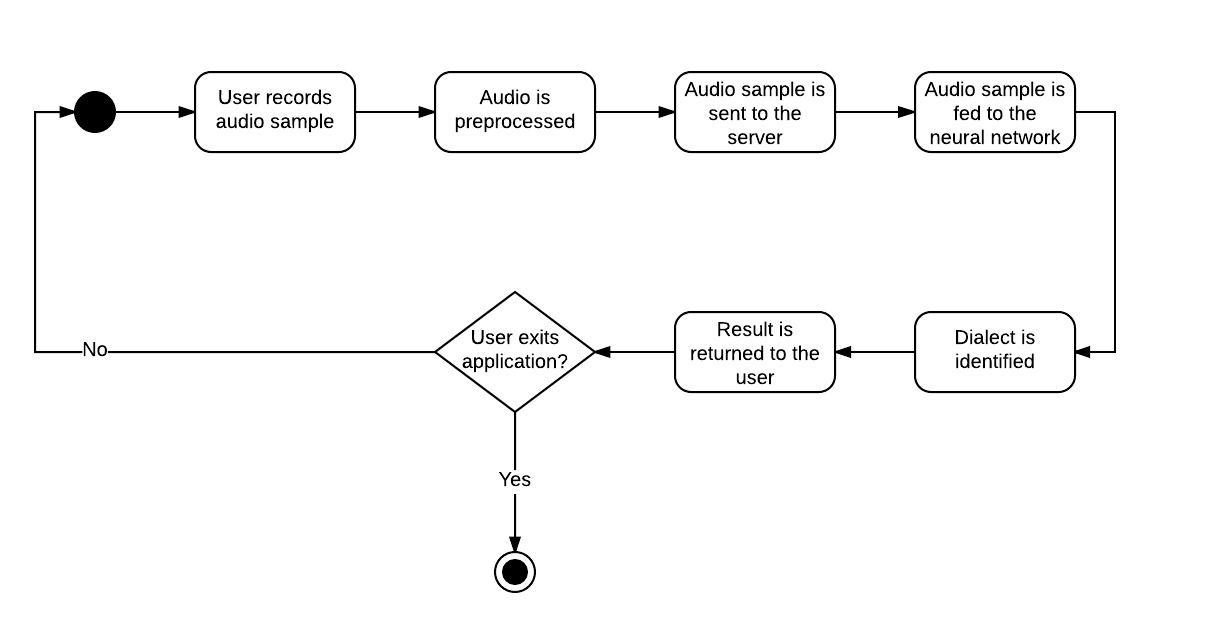
****

Figure 6 | Dialect Identification Activity Diagram