1

Table of Contents

[DECLARATION 2](#_Toc407879026)

[CERTIFICATE 3](#_Toc407879027)

[ACKNOWLEDGEMENT 4](#_Toc407879028)

[SUMMARY 5](#_Toc407879029)

[LIST OF FIGURES 6](#_Toc407879030)

[LIST OF TABLES 7](#_Toc407879031)

[LIST OF SYMBOLS AND ACRONYMS 8](#_Toc407879032)

[Chapter 1. INTRODUCTION 9](#_Toc407879033)

[1.1 GENERAL INTRODUCTION 9](#_Toc407879034)

[1.2 CURRENT RELEVANT PROBLEMS 10](#_Toc407879035)

[1.3 PROBLEM STATEMENT 11](#_Toc407879036)

[1.4 OVERVIEW OF SOLUTION 12](#_Toc407879037)

[CHAPTER 2. BACKGROUND STUDY 13](#_Toc407879038)

[2.1 LITERATURE SURVEY 13](#_Toc407879039)

[2.1.1 SUMMARY OF PAPERS 13](#_Toc407879040)

[2.1.2 INTEGRATED SUMMARY OF LITERATURE 17](#_Toc407879041)

[2.1.3 COMPARISON OF EXISTING APPROACHES WITH PROBLEM STATEMENT. 18](#_Toc407879042)

[2.2 DETAILS OF EMPIRICAL STUDY 19](#_Toc407879043)

[2.2.1 Field Survey 19](#_Toc407879044)

[2.2.2 Experimental Studies 19](#_Toc407879045)

[2.2.3 New Tools/Technologies 19](#_Toc407879046)

## DECLARATION

We hereby declare that this submission is our own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: Noida Signature:

Date: Jan 02, 2015 Name: Mahima Goel

Enrollment No: 11103499

Signature:

Name: Shivam Tiwari

Enrollment No: 11104718

Signature:

Name: Siddharth Taneja

Enrollment No: 11103586

## CERTIFICATE

This is to certify that the work titled “**JSPHINX**” submitted by “**MAHIIMA GOEL, SHIVAM TIWARI AND SIDDHARTH TANEJA**” in partial fulfilment for the award of degree of **B.Tech** of Jaypee Institute of Information Technology University, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor ……………………..

Name of Supervisor ……………………..

Designation ……………………..

Date ……………………..

## ACKNOWLEDGEMENT

In performing our project, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. We would like to show our gratitude **Mr. Kishore Kumar Y., Assistant Professor, Jaypee Institute Of Information Technology**for giving us a good guideline for the project throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in writing this assignment.

Many people, especially our classmates and team members itself, have made valuable comment suggestions on this proposal which gave us an inspiration to improve our project. We thank all the people for their help directly and indirectly in the same.

Signature of the Student ……………………..

Name of Student Mahima Goel

Enrollment Number 11103499

Signature of the Student

Name of Student Shivam Tiwari

Enrollment Number 11104718

Signature of the Student ……………………..

Name of Student Siddharth Taneja

Enrollment Number 11103586

Date : Jan 02, 2015

## SUMMARY

JSphinx, in its final stage, will be an API wrapped over CMU Sphinx 4 API. It will provide the required functionality to tackle the problems mentioned in the problem statement while providing abstraction from CMU SPhinx API to the developers. All functions and features of CMU Sphinx 4 will be relayed across and additional features will be added.

Machine Learning and training algorithms for phonetics identification are already present in CMU Sphinx 4. Machine Learning and training algorithms for accent recognition, sharing training set across devices, etc will be in the JSphinx module.

We will, primarily, use Java for code development and Maven architecture. In its final stage we would convert it into a Sonatype OSS repository API.

## LIST OF FIGURES

## LIST OF TABLES

## LIST OF SYMBOLS AND ACRONYMS

# Chapter 1. INTRODUCTION

## 1.1 GENERAL INTRODUCTION

JSphinx is a development over CMU Sphinx which integrates phonetic analysis & speaker identity with already existing speech to text API. This is intended to solve two problems

1. **Provide a common speaker identity**  
   This is intended to share and sync the learning/training data on one device with all other devices of the same user.

1. **Improve speech to text performance across cultures and geographies**  
   People from various cultures and geographies have different accents. Any modern speech to text program can be gradually trained to cater to accent of the user. But it takes time to train and personalize the speech to text program. Keeping a track of distinctive phonetic signatures mapped to similar learning sets can help in faster training of such programs.

## 1.2 CURRENT RELEVANT PROBLEMS

A few current open problems that we were able to pin point are

1. Standardization of speech to text API and establishing a common universal speaker identity wouldn’t be easy as applications are vendor specific.
2. Creating a database to train and test JSphinx is a tedious task considering the variations in accents that are essential to this project.

## 1.3 PROBLEM STATEMENT

Speech to text applications are changing the way we perceive, control and interact with IT products around us. Using techniques like Machine Learning and AI to train the application and personalize it to user’s tone we have increased the usability of such applications. However, there are two main problems with such systems.

1. If a user uses speech to text applications on N devices, then he trains each one of them independently to suit his accent. There is no universal speaker identity which allows to share learning data across devices.
2. Each user trains his own device. Even though many users from same culture of geolocations have similar accents there is no way to use this fact to increase the performance of speech to text applications.

Through JSphinx we will try to solve these two problems.

## 1.4 OVERVIEW OF SOLUTION

CMU Sphinx already applies Machine Learning techniques in the speech to text application and trains itself according to a user’s accent. To solve the two problems mentioned above we will encapsulate CMU Sphinx in JSphinx. CMU Sphinx Engine (and database) will be accessed through the API and will run on the CMU architecture. JSphinx will interface with it as an inside module. It will have its own engine to extract signature phonetics and choose best fit text for a particular speech. It will use Machine learning and AI to achieve this. This data will be stored in JSphinx database and will run on our architecture.

In basic terms JSphinx uses CMU SPhinx as a module for speech to text processing and solves the two problems mentioned in the problem statement itself.

# CHAPTER 2. BACKGROUND STUDY

## 2.1 LITERATURE SURVEY

### 2.1.1 SUMMARY OF PAPERS

**Paper 1**

Title : Environmental Robustness in Automatic Speech Recognition

Authors : Alejandro Acero, Richard M. Stern

Year Of Publication : 1990

Publishing Details : Department of Electrical and Computer Engineering and

School of Computer Science, Carnegie Mellon University, Pittsburgh , Pennsylvania

Summary : Efforts made to increase the robustness of CMU to the environment, two algorithms were proposed. First, SDCN, which adds a correction vector depending on the instantaneous SNR response of the input.Second, CDCN,uses a maximum likelihood technique to estimate noise and spectral tilt in the context of an iterative algorithm.An alphanumeric database was tested on two microphones ,a close talking and a desk-top microphone.

Weblink : [Paper 1](https://drive.google.com/drive/#folders/0B_8cThvSzWuNa21zdjFveEt6d0E/0B_8cThvSzWuNMzNibW1fMkJLbTQ/0B_8cThvSzWuNN180LVpXQ0lIcDA)

**Paper 2**

Title : The 1999 CMU 10X Real Time Broadcast News Transcription

System

Authors : Mosur Ravishankar, Rita Shankar, Bhiksha Raj, Richard M.S.

Year Of Publication : 2000

Publishing Details : Department of Electrical and Computer Engineering and

School of Computer Science, Carnegie Mellon University, Pittsburgh , Pennsylvania

Summary : Describes the architecture of the CMU-SPHINX-III fast decoder and

the various components of the recognition system. The

two models,full bandwidth and narrow bandwidth,were

and the primary was tested. Various components of the CMU

system,namely,Signal processing,Segmentation,Acoustic

models,Language Models,were described and instantiated.

Web Link : [Paper 2](https://drive.google.com/file/d/0B_8cThvSzWuNanFXQXVEWTliOTg/view?usp=sharing)

**Paper 3**

Title : Speech Recognizer-Based Microphone Array Processing For

Robust Hands-Free Speech Recognition

Authors : Michael L.S. , Bhiksha Raj, Richard M.S.

Year Of Publication : 2002

Publishing Details : 1)Department of Electrical and Computer Engineering and

School of Computer Science, Carnegie Mellon University, Pittsburgh , Pennsylvania

2) Mitsubishi Electric Research Labs, Cambridge, USA

Summary : A new array processing algorithm for microphone array

speech recog.. The new algorithm uses an objective function

which utilizes information from the recognition system itself

to optimize the parameters of a filter-and-sum array processor

We see a 36% improvement of the new algorithm.

Web Link : [Paper 3](https://drive.google.com/file/d/0B_8cThvSzWuNWTNGU0xvQWlLQ1U/view?usp=sharing)

**Paper 4**

Title : HMM-Based Audio Keyword Generation

Authors : Min Xu, Ling-Yu Duan, Jianfei Cai, Liang-Tien Chia,

Changsheng Xu and Qi Tian

Year Of Publication : 2004

Publishing Details : 1) School of Computer Engineering, Nanyang Technological University, Singapore

2) Institute for Infocomm Research, 21 Heng Mui Keng

Terrace, Singapore

Summary : Proposed classification method based on Hidden Markov

Model (HMM) for audio keyword identification as an improved

work instead of using SVM classifier. Unlike the frame-

based SVM classification, HMM-based classifiers treat specific

sound as a continuous time series data and employ hidden

states transition to capture hidden information.

Web Link : [Paper 4](https://drive.google.com/file/d/0B8B571L8a-NCc2V0ZDM3VnVURDA/view?usp=sharing)

**Paper 5**

Title : Minimum Variance modulation filter for robust speech

Recognition

Authors : Yu-Hsiang Bosco Chiu and Richard M Stern

Year Of Publication : 2009

Publishing Details : Dept of Electrical and Computer Engineering and Language

Technologies Institute, Carnegie Mellon University, Pittsburgh

PA, USA

Summary : Designing a modulation filter by data driven analysis which

improves the performance of automatic speech recognition

systems that operate in real environments. Recognition accuracy is measured using CMU-SPHINX III and DARPA Resource Management and Wall Street Journal speech corpus for testing and training.

Weblink : [Paper 5](https://docs.google.com/file/d/0B_8cThvSzWuNNVlZaHd1VEp4RUU/edit)

**Paper 6**

Title : Automatic Generation Of Subword Units For Speech

Recognition Systems

Authors : Rita Singh, Bhiksha Raj, and Richard M.S.

Year Of Publication : 2002

Publishing Details : IEEE Transactions On Speech And Audio Processing,Vol No.

2. February 2002

Summary : A complete probabilistic formulation for the automatic design

of subword unit and dictionary,given only the acoustic data and

their transcriptions. The problem of automatically designing

the subword units and the dictionary given only a set of acoustic signals and their transcriptions,is met by forming a modeling perspective viz. identifying the sound class that best

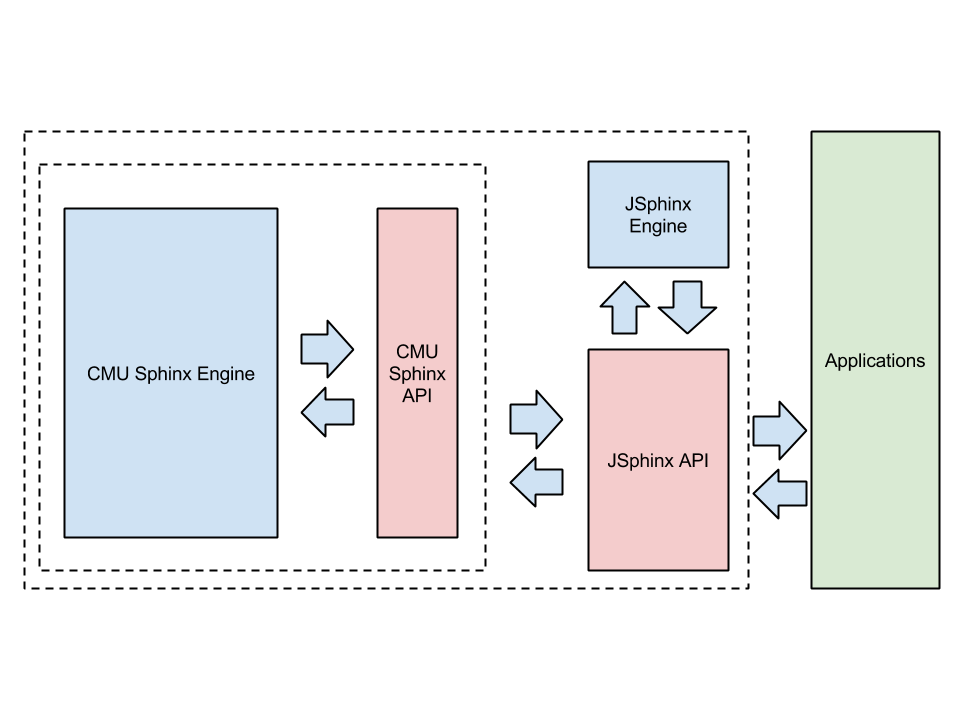
fit the training set.

Weblink : [Paper 6](https://drive.google.com/file/d/0B_8cThvSzWuNMFlSaUU1c1M0bEE/view?usp=sharing)

### 2.1.2 INTEGRATED SUMMARY OF LITERATURE

After scanning 25 years of literature of CMU Sphinx and looking into a couple of more projects we feel that speech to text applications have reached an appreciable development stage. It is time that we create a universal unique speaker identity for cross-domain applications for supporting portability and sharing the learning data across devices. So instead of having many stand-alone speech to text training instances we could just share all this data to make the process faster and more efficient.

To achieve this we plan to create a wrapper class around CMU Sphinx API that interfaces with a database that stores user’s phonetic signatures to mark his accent against group learning data for similar phonetics. It will also be mapped to his/her universal speaker identity.



### 2.1.3 COMPARISON OF EXISTING APPROACHES WITH PROBLEM STATEMENT.

SIDDHARTH

## 2.2 DETAILS OF EMPIRICAL STUDY

### 2.2.1 FIELD SURVEY

We asked around people for feedback on Siri, Okay Google and Windows Speech Recognition. We tried to get an idea of things they liked and disliked about their speech recognition apps.

### 2.2.2 EXPERIMENTAL STUDIES

No experimental studies have been conducted in this research yet.

### 2.2.3 NEW TOOLS AND TECHNOLOGIES

We have used the following new tools and technologies in this project so far

1. Maven  
   <http://maven.apache.org/>
2. CMU-Sphinx  
   <http://cmusphinx.sourceforge.net/>
3. Sonatype-OSS  
   <https://oss.sonatype.org/>

# CHAPTER 3. ANALYSIS, DESIGN AND MODELLING

## 3.1 REQUIREMENT SPECIFICATIONS

JSphinx, in its final stage, will be an API wrapped over CMU Sphinx 4 API. It will provide the required functionality to tackle the problems mentioned in the problem statement while providing abstraction from CMU SPhinx API to the developers. All functions and features of CMU Sphinx 4 will be relayed across and additional features will be added.

Machine Learning and training algorithms for phonetics identification are already present in CMU Sphinx 4. Machine Learning and training algorithms for accent recognition, sharing training set across devices, etc will be in the JSphinx module.

We will, primarily, use Java for code development and Maven architecture. In its final stage we would convert it into a Sonatype OSS repository API.

## 3.2 FUNCTIONAL AND NON FUNCTIONAL REQUIREMENTS

### 3.2.1 Non-functional Requirements

**Requirement 1**

Requirement ID : NFR001

Requirement Name : Apache Server-enabled backend server space

Requirement Description : To run the API we will require linux web hosting running APache Server on it because Maven architecture guidelines work best on Apache servers.

**Requirement 2**

Requirement ID : NFR002

Requirement Name : Web client hosted on web

Requirement Description : To run the application using a web browser we will need to create a website that acts as a web client and interfaces with the API running in NFR001.

**Requirement 3**

Requirement ID : NFR003

Requirement Name : Mobile applications running locally on devices

Requirement Description : To demonstrate the portability of this project across platforms we need to create mobile applications for major manufacturers viz. iOS, Android and Windows Phone.

**Requirement 4**

Requirement ID : NFR004

Requirement Name : Training data

Requirement Description : Training data from other similar projects need to be extracted and used to teach JSphinx before it is tested on real data set.

**Requirement 5**

Requirement ID : NFR005

Requirement Name : Audio Samples

Requirement Description : To test this project we would require Audio samples of different accents. We will need to take these samples across cultures and geo locations to test the program on as many distinctive accents as possible.

**Safety requirements**

System should keep the safety of user data in mind and should not cause any harm to it.

**Security requirements**

The user data (voice samples) should be kept secure from outside intervention. The user database should be properly scrutinized to prevent any sort of hacking issues that are common in such cases.

**Error handling**

The user data is properly expertised in handling user data errors. Both expected and non-expected errors can be prevented before the data is fed to the exception handler module.

**Reliability**

The data is already reliable as it has been taken from speaker.

#### 

**Correctness**

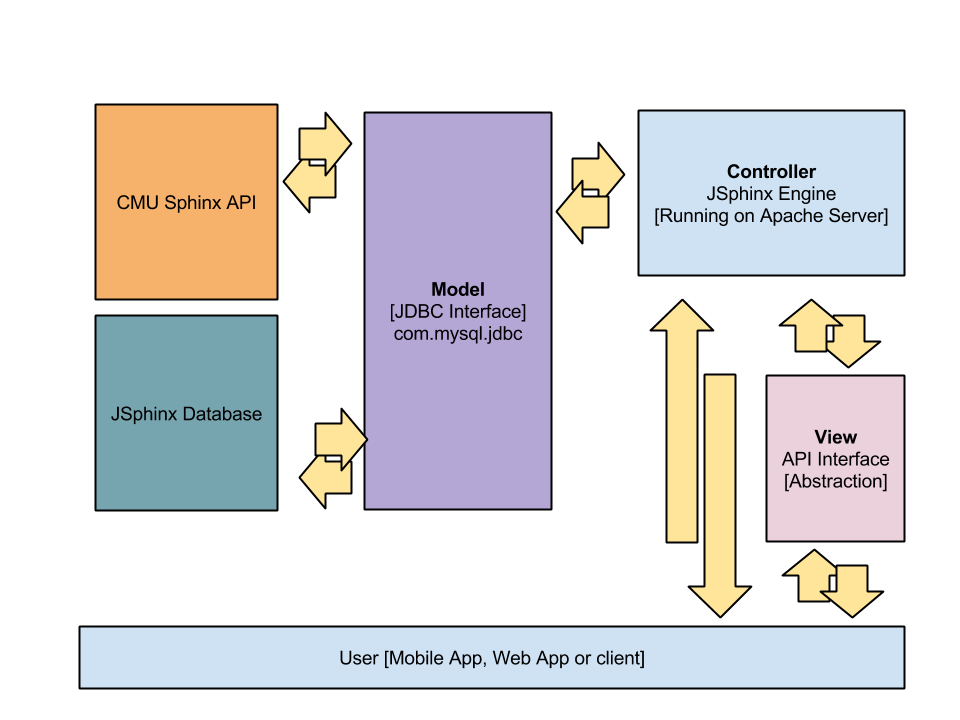
The results obtained from the training set are then matched with the documented results available. This ensures the relative correctness of the output.

### 3.2.2 Functional Requirements

SHIVAM

## 3.3 ARCHITECTURE

JSphinx uses a modified MVC architecture. The components of this architecture are shown in the diagram below.



## 3.4 DESIGN

### 3.4.1 USE CASE DIAGRAMS

### 3.4.2 CLASS DIAGRAM / CONTROL FLOW DIAGRAM

### 3.4.3 SEQUENCE / ACTIVITY DIAGRAM

### 3.4.4 DATA STRUCTURES AND ALGO / PROTOCOLS

SHIVAM

## 3.5 RISK ANALYSIS AND MITIGATION

SIDDHARTH

# CHAPTER 4. IMPLEMENTATION AND TESTING

## 4.1 IMPLEMENTATION DETAILS AND ISSUES

SHIVAM

## 4.2 TESTING

### 4.2.1 TESTING PLAN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | Type of test | Will Test be performed? | Explanation | Software Component |
| 1. | Requirements Testing | Yes | To test the successful implementation of all the requirements in the problem statement |  |
| 2. | Unit Testing | Yes | To check if each part of the code is working individually |  |
| 3. | Integration Testing | Yes | To check if the individual modules work together when integrated |  |
| 4. | Performance Testing | Yes | To test how the system performs in terms of responsiveness and stability under a particular workload |  |
| 5. | Stress Test | Yes | To test the stability under intense workload. |  |
| 6. | Compliance Testing | No | Testing done to check if the software complies with the company standard. Not required in our project |  |
| 7. | Security Testing | No | To reveal security flaws. Doesn’t apply to our system |  |
| 8. | Load Testing | No | To test the system under expected load. Already covered in stress testing |  |
| 9. | Volume Testing | No | To test the software under a certain amount of data. Already covered in stress testing |  |