**Technical Design Document (TDD)**

**AI-Enabled recommendation chatbot**

|  |  |
| --- | --- |
| **Mentors*:*** | **Authors:** |
| *Prof. Sudhir Wadhwa*  *Jyoti Wadhwa*  **Mentor (Client):**  *Satish Pendurthi*  *Bharani Kumar*  *Sharat Manikonda* | *Himanshu Kesarvani*  *Chirag Sharma*  *Seshadri Deb Roy*  *Amit Barik* |

Table of Contents

[1. Introduction 4](#_Toc43633323)

[1.1 Purpose 4](#_Toc43633324)

[1.2 Scope 4](#_Toc43633325)

[1.3 Document Organization 4](#_Toc43633326)

[1.4 Audience 4](#_Toc43633327)

[2. System Overview 5](#_Toc43633328)

[2.1 Context 5](#_Toc43633329)

[2.2 Product Feature 5](#_Toc43633330)

[2.3 Technologies Used 5](#_Toc43633331)

[3. Application Architecture 5](#_Toc43633332)

[3.1 Architecture Design as per student perspective 5](#_Toc43633333)

[3.2 Architecture Design as per video processing 6](#_Toc43633334)

[3.2 System Operations 7](#_Toc43633335)

[4. Database Architecture (Entity Relationship Diagram) 8](#_Toc43633336)

[4.1 Database 8](#_Toc43633337)

[4.2 Tag Generation and Storing 8](#_Toc43633338)

[4.3 Models 8](#_Toc43633339)

[5. Approach 9](#_Toc43633340)

[5.1 Doc2Vec 9](#_Toc43633341)

[5.2 TF-IDF Vector 9](#_Toc43633342)

[5.3 SVM with SGD training 9](#_Toc43633343)

[6. Assumption and Constraints 10](#_Toc43633344)

[6.1 Students are registered 10](#_Toc43633345)

[6.2 Recommend Modules for registered courses 10](#_Toc43633346)

[6.3 Focus on model building 10](#_Toc43633347)

[6.4 Video with clean audio not multiple voice interference 10](#_Toc43633348)

[7. Scalability 10](#_Toc43633349)

[7.1 Multiple video processing 10](#_Toc43633350)

[7.2. Concurrency management 10](#_Toc43633351)

[8. Further Suggestion 10](#_Toc43633352)

[8.1. Student registration and course assignment 10](#_Toc43633353)

[8.2. Dictionary keywords (tags) 11](#_Toc43633354)

[9. References 11](#_Toc43633355)

**Revision History**

|  |  |  |
| --- | --- | --- |
| Version | Date | Description |
| 1.0 | 19/04/2020 | Initial Draft |
| *2.0* | 19/06/2020 | Final Draft |

# Introduction

## 1.1 Purpose

The purpose of this document is to outline the high-level design of the ***AI enabled recommendation system*** and provide an overview for the tool implementation.

Its main purpose is to –

* Recommend relevant subscribed course module for student query

## Scope

***AI enabled recommendation system*** is an AI chatbot which receive questions from the students, tries to understand the question and provide appropriate recommendation links from students subscribed course modules. If it doesn’t find any answer in user subscribed module then it provides the link of courses which user can buy and clarify his query.

It does this by extracting the relevant key words from the query using NLP, then look up in the database for key words and extract necessary information to recommend module links.

The main objective is to develop an NLP, ML based model which will take courses video input and extract relevant keywords and store in the database so that if student query he should able to find relevant module to resolve his doubt.

## 1.3 Document Organization

This document is organized into the following section:

|  |  |
| --- | --- |
| Introduction | Provides information related to the document |
| System Overview | Describes the approach, architectural goals and constraints, Guiding principles |
| Application Architecture | Describe the application architecture in terms of different layers of application. Description of the presentation layer, business layer, data access layer and resource layer and their relationship to each other. |
| Database Architecture | Describes the overall Data model and entity relationship diagram |
| Assumptions and Constraints | Details various assumptions made during design and development of the Online Screening tool |

## Audience

The intended audiences for this document are: -

* Innodatatics Inc.
* The project development team
* Mentors

# System Overview

## 2.1 Context

The evolving tech industry serves the business needs. The first key area for a business to grow is to identify the need or demands of the user. The latest cut edge technology such as a smart AI chat bot is the need of the hour. The chat bot will provide a platform where the user can interact with the business and give their requirements. After receiving the inputs from the user, the chat bot may provide solutions which will serve their purpose.

    Enrolled students are looking to resolve their queries using chatbot. So, we would be recommending relevant modules to students pertaining to their questions related to enrolled courses. In today's education sector there's a huge scope of improving the business. A smart AI enabled chat bot is a solution for the business to grow.

## 2.2 Product Feature

The major feature of the ***AI enabled recommendation system*** will be the following: -

* + **REST API - A** REST API call will include a question in the form of a query string URL parameter and the service will reply in JSON
  + **Natural Language Processing –** The question will take query process it extracts keywords from query string using NLP
  + **Information Extraction –** There will be a database containing all the information needed, populated using information extraction technique

## Technologies Used

The REST API will be developed in Python using Flask framework, Model will be implemented in Python. The frontend web application will be implemented using HTML, CSS, JavaScript.

The system will have Machine Learning libraries too.

**Front-end –** HTML, CSS, JavaScript, AJAX

**Middleware (REST API) –** Python, Flask framework

**Backend –** Mongo dB/Postgres SQL

# Application Architecture

## 3.1 Architecture Design as per student perspective

**Front-end Application –** Web app with chatbot UI which receive questions from user in text and speech format.

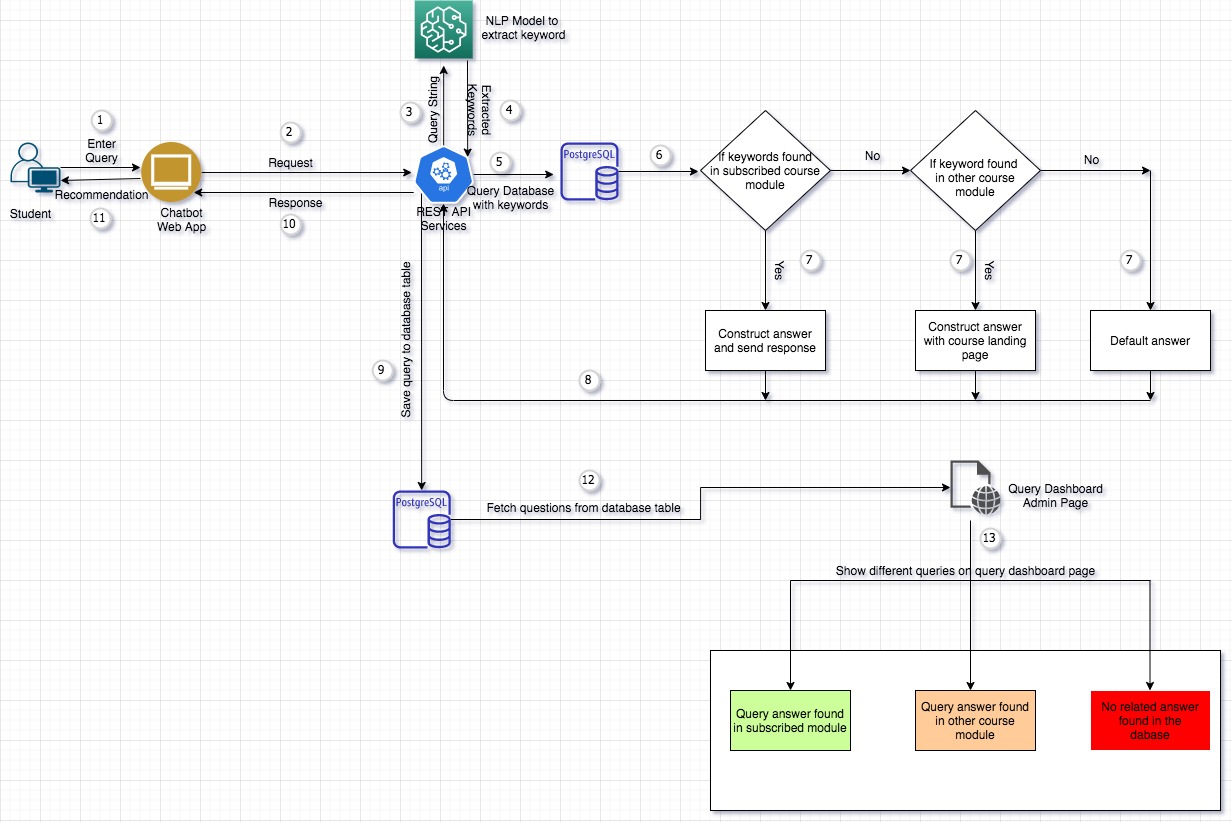
**REST API Services –** Receive HTTP GET request containing user query and forward them to ML based model

**ML Model –** Model will receive the string query and process to extract the key words/labels. This model will also be responsible to extract key words from the video files too.

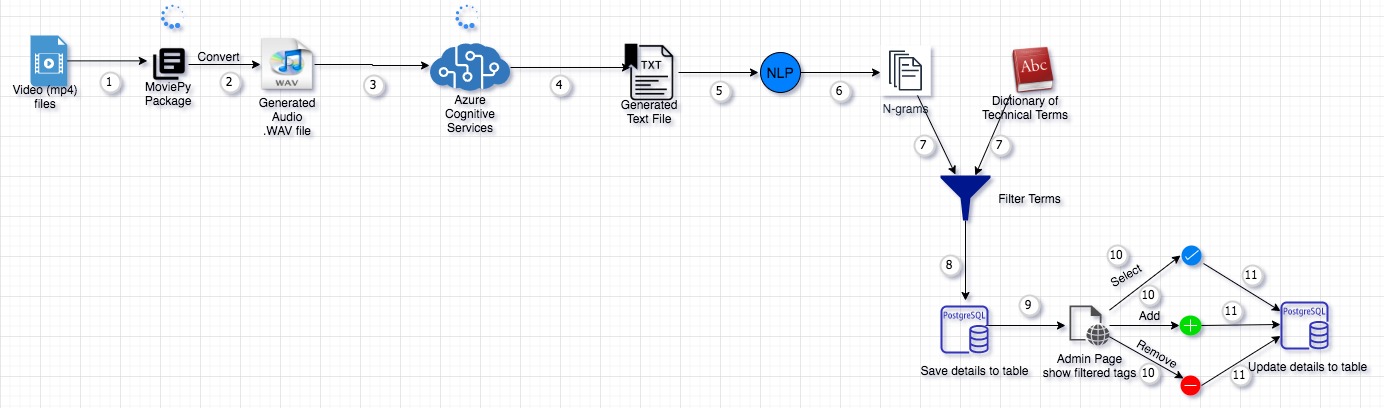
**Database –** Store key words/labels in the database with module info

**Recommendation –** Query database and fetch resources matching with the key words extracted from string query. There are 3 scenarios here which explained below:

1. ***Match found in subscribed modules: -*** If match found in the student subscribed course it will return the module video link as recommendation.
2. ***Match found in other than subscribed modules: -*** If query match found in other modules which aren’t subscribed by student then it will return link of course landing page.
3. ***No match found: -*** If nothing matches then it will return default message.

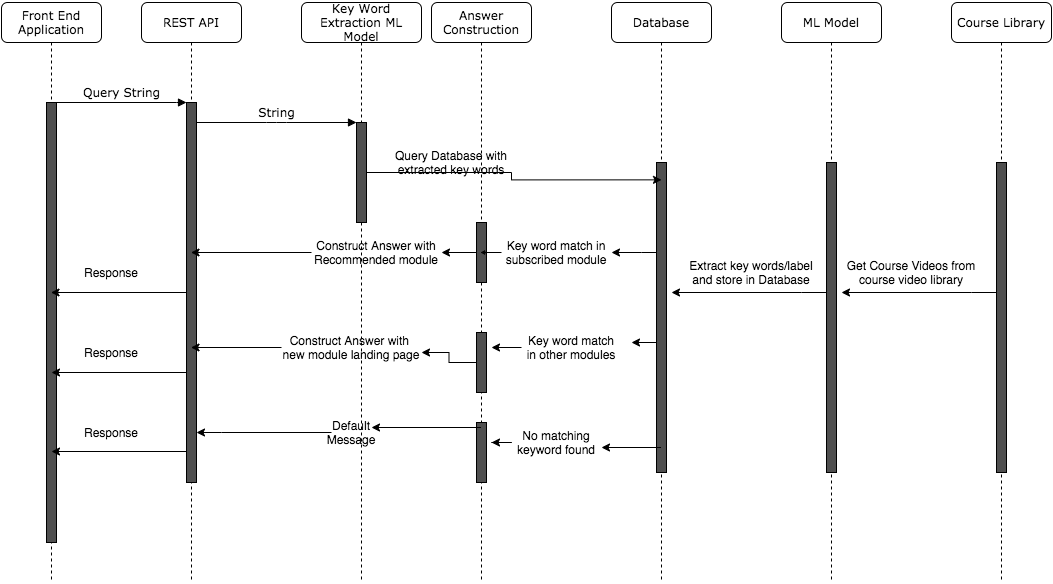


## 3.2 Architecture Design as per video processing



## 3.2 System Operations

Below figure is illustration of interaction between system subsystem in during use of application



# Database Architecture (Entity Relationship Diagram)

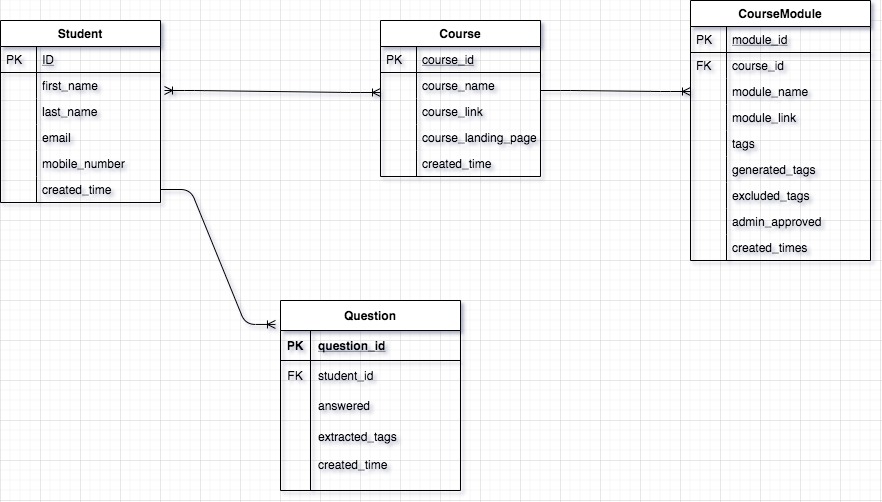
## 4.1 Database

Chatbot uses database to store information regarding student, course, and course module. The database essentially serves as an all-encompassing cluster that provides the data needed for all recommendation query.

## 4.2 Tag Generation and Storing

Tag data will be generated from existing course videos and store in the database. The tag generation will be done by our ML model.

Entity Relation diagram of database shown below: -



## Models

Started with 3 model which will be used in the solution (Student, Course, Course Module). These are the initial understanding of models and fields which will be changed if require in future development.

**Student –** This model will be used to store student information with below fields

* Name
* Email
* Mobile
* Password
* Course Subscription = List of course ID

**Course –** This model will be used to store different course information with below fields

* ID
* Name
* Course Link
* Course Landing Page Link
* Created Time

**Course Modules –** This model will be used to store all modules related to course with below fields

* Module ID
* Course ID (Foreign Key)
* Module Link
* Tags/Label = List of tags
* Created Time
* Excluded Tags = List of tags
* Generated Tags = List of tags
* Admin Approved

**Question –** This model will be used to store all question which student has asked

* Question ID
* Student ID (Foreign Key)
* Answered (Yes, No, Found in Another module)
* Question
* Created Time
* Extracted Tags

**Relationship-** There is many to many relations between student and course model, and one to many between course and course modules

# Approach

We have tried ML approached to solve the issue which are listed below: -

## 5.1 Doc2Vec

Using ***genism*** library with doc2vec, and news group data created label/tags for each news group

## 5.2 TF-IDF Vector

Using ***NLTK*** library create TF-IDF vector removed stop words and generated tags/label for news group

## SVM with SGD training

Using ***sklearn*** library using linear classifier in SVM with Stochastic Gradient Descent technique created tags/labels

We are exploring more on Deep Learning and other text mining technique to get better accuracy.

# Assumption and Constraints

## 6.1 Students are registered

Our first assumption here is that the user asking question in the chatbot is already registered with the portal. There won’t be any chatbot for un-registered users

## 6.2 Recommend Modules for registered courses

Recommendation modules will be shown only from those courses which user has subscription. If user ask question which matches with other module which user hasn’t subscribe we need to show landing page of course matching with query.

## Focus on model building

Team main focus to develop better model to generate tags which will be used to recommend for course. The end to end application will be minimal to test and present model accuracy.

## Video with clean audio not multiple voice interference

While building model extraction tags from video we found out that some of the video where multiple person are speaking it’s not able to convert video to text because of noise. So, we have taken those videos in consideration which has less noise.

# Scalability

The below points will explain how can we further scale the solution:

## 7.1 Multiple video processing

As discussed with the team they have multiple video to process daily basis, to solve this issue we can parallelize the video processing pipeline while deploying to any cloud platform (or server)

## 7.2. Concurrency management

If chatbot started getting access by multiple user and frequency of query per second increases in this scenario we have to make few changes to solution which is like production-ready web servers like *Nginx*, and let your app be handled by WSGI application server like *Gunicorn*.

# Further Suggestion

## 8.1. Student registration and course assignment

To achieve this, we can create two pages:

1. ***Student Registration Page****: -* Student can register by providing his details (e.g. Name, email, mobile number) and we can store the information to database
2. ***Admin page for course assignment****: -* Create an admin page where course can be assigned to registered user

## 8.2. Dictionary keywords (tags)

We have 3 solution for this which discussed as below:

1. ***Analyze un-answered questions****: -* We can analyse the un-answered question by pulling to excel, csv format and check what frequent keywords has been asked from students and we can add to our dictionary database

1. ***Notify only frequent keywords:*** *-* As we are storing each query and keyword related to it we can add a threshold mark above which will be useful if same type of tag related query has been asked multiple time you can flag those question and show on dashboard and periodically admin can review and insert into database by admin portal

1. ***Complete automation:*** *-* As mentioned in the (b) part of solution we can create a threshold of question. If same type of question has been asked from multiple user. We can create a job which will run periodically and find those keywords and insert into database without any human intervention.

All the above 3 methods will work without issue and next time a student will ask same type of question chatbot will able to answer it.

# References

<https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/rest-speech-to-text>

<https://medium.com/analytics-vidhya/building-a-simple-chatbot-in-python-using-nltk-7c8c8215ac6e>

<https://towardsdatascience.com/build-your-first-chatbot-using-python-nltk-5d07b027e727>

<https://aiiseasy.com/2019/06/09/text-classification-svm-naive-bayes-python/>

<https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/>